ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ НАУКИ
ИНСТИТУТ ПРОБЛЕМ УПРАВЛЕНИЯ им. В.А. ТРАПЕЗНИКОВА РАН

На правах рукописи

КОВАЛЁВ Сергей Протасович

ТЕОРЕТИКО-КАТЕГОРНЫЕ МОДЕЛИ И МЕТОДЫ ПРОЕКТИРОВАНИЯ БОЛЬШИХ ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ

Специальность: 05.13.17 – Теоретические основы информатики

Диссертация на соискание ученой степени
доктора физико-математических наук

Научный консультант: академик РАН, д.ф.-м.н. Васильев Станислав Николаевич

Москва 2013

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность работы. Глобализация экономики и коммуникаций приводит к росту масштаба объектов, требующих охвата единым процессом управления или тесно согласованным комплексом процессов. Например, укрупняются до транснационального и межгосударственного уровня цепи поставки продукции, транспортные инфраструктуры высокой доступности, программы развития отраслей. Давно известно, что к числу основных способов повышения эффективности управления большими объектами относится комплексная автоматизация ([94] и др.). Поэтому объекты оснащаются цифровыми измерительными приборами и исполнительными механизмами, которые подключаются к программным средствам управления – информационно-управляющим системам, достигающим не только большого масштаба (large scale), но и сверхбольшого (ultra-large scale systems, ULS) [258]. По сравнению с традиционными системами, они обладают очень большими значениями показателей размера – количества данных, элементов, взаимосвязей, процессов, нормативов, пользователей и др. Примером служит Smart Grid («умная» сеть) – система сквозной автоматизации крупной электроэнергетической сети [142].

Традиционные подходы к созданию информационных систем не были рассчитаны на поддержку больших значений размера. Кроме того, рост масштаба приводит к проявлению принципиально новых проблем, незаметных при малых размерах и затрудняющих соблюдение классических принципов разработки АСУ, сформулированных полвека назад [28]. Например, принцип первого руководителя требует создавать систему под непосредственным руководством лица, способного выступить главным заказчиком (приобретателем) системы и непререкаемым арбитром в разрешении конфликтов между ожиданиями групп пользователей. Однако на большом объекте такое лицо может отсутствовать, если каждый руководитель имеет недостаточный уровень полномочий и/или управляет только частью объекта. К тому же, большой объект нестабилен: почти все время в нем присутствуют участки, находящиеся в процессе существенного изменения, способного повлиять на потребности пользователей. В результате не удается выдать разработчикам полный непротиворечивый набор требований к информационно-управляющей системе, и она входит в непрерывный процесс развития и адаптации. Возникают трудности в применении принципа типовости: типовые решения, предназначенные для автоматизации заранее заданных задач, требуют огромных затрат на адаптацию к априори неизвестным и постоянно меняющимся условиям их использования в системе. Ключевую роль приобретает трассирование компонентов к задачам – одна из самых трудоемких операций в инженерии информационных систем [205].

В связи с этим большой интерес вызывают новые технологии разработки систем, направленные на уменьшение затрат труда путем построения широкого набора моделей, заполняющих «когнитивную дистанцию» между автоматизируемой предметной областью и программным кодом [227]. Такие технологии включают инструменты для быстрого пошагового преобразования моделей, выражающих разные точки зрения на задачи, в программы и структуры данных, с высокой степенью верифицируемости и трассируемости. К таким технологиям относятся инженерия предметной области (domain engineering), разработка, управляемая моделями (model-driven engineering, MDE), организация распределенных вычислений (distributed computing), аспектно-ориентированный подход (aspect-oriented software development, AOSD).

Применение таких технологий в проектировании больших систем требует масштабировать их – приспособить к гибкому манипулированию многочисленными, сложными, разнородными моделями [211]. Здесь необходима глубокая автоматизация, поэтому технологии жизненного цикла должны иметь единую формальную теоретическую базу, позволяющую кратко описать механизмы масштабирования, сформулировать и доказать их основные свойства, не «потонув» в деталях структуры частных моделей. Однако большинство формальных методов современной инженерии базируется на разнородных «тяжеловесных» математических средствах, подогнанных под разнообразные частные парадигмы программирования и вследствие этого плохо совместимых друг с другом. Технологии широкого назначения, подобные перечисленным выше, способные порождать рациональные типовые решения, развиваются в основном *ad hoc*, не опираясь на математические методы [169].

Таким образом, формирование теоретической основы технологий проектирования больших информационно-управляющих систем, свободной от указанных недостатков, является важной научной проблемой. В качестве математического аппарата, пригодного для ее решения, целесообразно привлечь теорию категорий. Эта теория позволяет явно и компактно выразить основные положения системной инженерии [181]. Артефактам[1] технологий можно сопоставить объекты подходящей категории (формальные модели), а технологическим процессам – морфизмы, перерабатывающие объекты-области (входы) в объекты-кообласти (выходы) [174]. Переходы между технологиями, сохраняющие структуру процессов, могут быть представлены функторами. Процедурам синтеза сложных моделей отвечают диаграммы в таких категориях («мегамодели» [203]), так что их анализ позволяет выявлять рациональные типовые решения, в том числе с привлечением автоматизированных инструментов [252, 207]. Для этого требуется построить теоретико-категорные конструкции, наиболее точно отражающие ключевые проце-

[1] Напомним, что артефактом в инженерии программного обеспечения традиционно называется результат любой деятельности, выполняемой в рамках жизненного цикла, от лат. artefactum – искусственно сделанное [19].

дуры жизненного цикла, и доказать их основные свойства, важные с прикладной точки зрения. Общие конструкции такого рода редко встречаются в литературе – чаще изучаются частные категории, описывающие частные формальные методы.

Цель работы – повышение эффективности жизненного цикла больших информационно-управляющих систем путем создания единой формальной базы технологий их проектирования.

Достижение этой цели требует решения следующих **задач**:

- проведение системного анализа жизненного цикла больших информационно-управляющих систем;
- построение аппарата для формального анализа и синтеза технологий проектирования систем на основе теории категорий;
- построение формальной технологии проектирования распределенных вычислительных систем;
- построение формальных технологий совместного аспектно-ориентированного моделирования данных и процессов;
- применение построенного формального аппарата для рационального проектирования прикладных информационно-управляющих систем.

Эти задачи решены в работе с использованием **методов** теории категорий, теории моделей, инженерии информационных систем.

Научная новизна работы состоит в том, что впервые построен и теоретически обоснован аппарат для математического (формального) анализа и синтеза технологий проектирования программных систем на основе теории категорий, позволяющий находить рациональные типовые решения проблем масштабируемости (scalability), трассируемости (traceability), разделения ответственности (separation of concerns). Путем применения этого аппарата впервые построены математические (формальные) технологии, способные служить теоретической основой для широкого класса методов проектирования информационно-управляющих систем.

Практическая значимость работы заключается в применимости результатов для повышения эффективности жизненного цикла больших информационно-управляющих систем, в том числе для решения следующих практических задач:

- масштабирование технологий моделирования в целях обеспечения их применимости в жизненном цикле большой системы;
- отображение вычислительных алгоритмов на архитектуру гетерогенной распределенной вычислительной среды;
- трассирование артефактов жизненного цикла к классам задач, решаемых системой;

- расширение модульных технологий проектирования аспектно-ориентированными приемами и инструментами;
- совместное моделирование данных и процессов.

Положения, выносимые на защиту. На защиту выносятся:

- аппарат для математического (формального) анализа и синтеза технологий проектирования систем на основе теории категорий;
- алгебраические методы отображения алгоритмов на архитектуру распределенной вычислительной среды;
- теоретико-категорная семантика расширения модульных технологий проектирования систем аспектно-ориентированными приемами с обеспечением трассируемости;
- теоретико-категорные модели процедур идентификации, связывания и модуляризации аспектов;
- теоретико-категорные методы совместного моделирования данных и процессов.

Достоверность и обоснованность результатов. Корректность теоретических результатов, изложенных в диссертации, обоснована рядом теорем, снабженных подробными доказательствами. В подтверждение достоверности практических результатов автором получено 4 акта внедрения научных и практических результатов исследований (Приложение 1), 5 свидетельств о регистрации программ для ЭВМ, 1 патент РФ на изобретение.

Внедрение результатов работы. При помощи подходов, предложенных в диссертации, рационально спроектированы и реализованы модели данных, процессов, вычислений и др. в следующих больших системах управления объектами топливно-энергетического комплекса:

- автоматизированная информационно-измерительная система коммерческого учета электроэнергии ООО «Транснефтьсервис-С» для ОАО «АК «Транснефть» (АИИС КУЭ ТНС, 2005-2007);
- автоматизированная информационно-измерительная система коммерческого учета электроэнергии ОАО «Томусинское энергоуправление» (АИИС КУЭ ТЭУ, 2008);
- автоматизированная система диспетчерского управления энергохозяйством ООО «Газпром энерго» (АСДУ ГПЭ, 2008-2009);
- единая интегрированная автоматизированная информационная система мониторинга и управления эффективностью энергосбережения на объектах города Москвы (ЕИАИС ЭЭ, 2010-2012).

Апробация работы. Результаты работы докладывались на следующих международных конференциях: 7th Joint European Networking Conference JENC7 (Budapest, Hungary, 1996); 8th

Joint European Networking Conference JENC8 (Edinburg, Scotland, 1997); 3 Смирновские чтения (Москва, 2001); Международная конференция по вычислительной математике МКВМ-2004 (Новосибирск, 2004); Международная конференция «Алгебра, логика и кибернетика-2004» (Иркутск, 2004); Всероссийская научная конференция «Научный сервис в сети Интернет» (Новороссийск, 2004); IX рабочее совещание по электронным публикациям El-Pub2004 (Новосибирск, 2004); 2nd IASTED International Conference on Automation, Control and Information Technology ACIT-2005 (Новосибирск, 2005); Международная конференция «Диалог'2005» (Звенигород, 2005); XI International Conference “Knowledge-Dialogue-Solution” (Varna, Bulgaria, 2005); Международная конференция «Вычислительные и информационные технологии для наук об окружающей среде» CITES-2005 (Новосибирск, 2005); 9th Asian Logic Conference (Новосибирск, 2005); International Conference “Molecular spectroscopy and atmospheric radiative processes” (Томск, 2005); X Российская конференция с участием иностранных ученых «Распределенные информационно-вычислительные ресурсы» DICR-2005 (Новосибирск, 2005); II Международная научно-техническая конференция «Новые информационные технологии в нефтегазовой отрасли и образовании» (Тюмень, 2006); Международная конференция «Вычислительные и информационные технологии в науке, технике и образовании» (Павлодар, 2006); Международная конференция «Алгебра и ее приложения» (Красноярск, 2007); VII Международная научно-практическая конференция «Исследование, разработка и применение высоких технологий в промышленности» (Санкт-Петербург, 2009); 9th Workshop on Foundations of Aspect-Oriented Languages (Rennes, France, 2010); 3rd IASTED International Conference on Automation, Control and Information Technology ACIT-2010 (Новосибирск, 2010); XIII Российская конференция «Распределенные информационные и вычислительные ресурсы» DICR-2010 (Новосибирск, 2010); XI Международная научно-практическая конференция «Фундаментальные и прикладные исследования, разработка и применение высоких технологий в промышленности» (Санкт-Петербург, 2011); Научная сессия НИЯУ МИФИ-2012 (Москва, 2012); XIV Международная конференция «Проблемы управления и моделирования в сложных системах» ПУМСС-2012 (Самара, 2012); VI Международная конференция «Управление развитием крупномасштабных систем» MLSD'2012 (Москва, 2012); Международная конференция «Алгебра и логика: теория и приложения» (Красноярск, 2013).

Также работа была представлена на научных семинарах Института проблем управления им. В.А. Трапезникова РАН, Вычислительного центра им. А.А. Дородницына РАН, Института вычислительных технологий СО РАН, Института математики СО РАН.

Публикации. По результатам выполненных исследований опубликовано 60 работ, в том числе: 17 в 10 журналах, ведущих рецензируемых изданиях, рекомендованных в действующем

перечне ВАК для публикации результатов докторских диссертаций (из них 11 без соавторов), 1 учебное пособие, 1 патент, 5 свидетельств о регистрации программ для ЭВМ.

Личный вклад автора. Все теоретические результаты, вынесенные на защиту, получены автором лично и опубликованы в работах без соавторов. Работы, опубликованные в соавторстве, посвящены практической апробации результатов в ходе создания прикладных систем. В них автору принадлежат концептуальные модели и проектные решения.

Структура и объем работы. Диссертация состоит из введения, 5 глав, заключения, библиографии и приложения. Общий объем работы – 281 страниц, библиография содержит 264 наименований.

В главе 1 изложены результаты системного анализа проблем снижения затрат на жизненный цикл больших информационно-управляющих систем при соблюдении ограничений качества. Построена типовая модель качества больших систем согласно стандарту ISO/IEC 9126. Описаны принципы рационального проектирования больших информационно-управляющих систем с применением аспектно-ориентированного подхода, путем трассируемого отображения задач на типовую архитектуру. Представлен подход к организации жизненного цикла систем, соответствующий стандарту ISO/IEC 12207-2008, на основе технологий инженерии предметной области и разработки, управляемой моделями. Показано, что для достижения масштабируемости этот подход необходимо проводить в рамках единой формальной базы, математический аппарат для построения и анализа которой способна предоставить теория категорий.

Глава 2 посвящена теоретико-категорному подходу к формализации процессов жизненного цикла, который позволяет единообразно описать многие известные технологии инженерии систем. В качестве отправной точки для выработки подхода использована известная конструкция математической (формальной) технологии проектирования. Выделен ряд классов формальных технологий, отражающих сложившуюся практику разработки систем. Введены свойства структурируемости, скординированности, трансформационной однородности и кооднородности технологий. На языке теории категорий сформулированы и исследованы практически значимые задачи комплексирования систем: применение шаблонов комплексирования, выявление рациональной архитектуры, выбор интеграционных интерфейсов, распараллеливание, покомпонентная трансформация систем, разработка специализированных технологий комплексирования систем. Построена формальная технология разработки аксиоматических спецификаций.

В главе 3 построена и теоретически обоснована формальная технология проектирования вычислительных систем путем отображения алгоритмов на архитектуру вычислительной среды. Предложена процедура частичной интерпретации, предназначенная для редукции алгебраических спецификаций алгоритмов на конечные множества доступных ресурсов с контролем корректности. Построена категория конечных алгебр, конструкции в которой описывают комплек-

сирование вычислительных систем, в том числе распределенных с динамическим развертыванием типа Grid или облака. Доказано, что обогащение спецификаций операциями условного вычисления и контроля переполнения представляет собой рефлектор в этой категории, порождающий полупримальные алгебры. Доказано, что его применение к кольцу $\mathbf{Z}/n\mathbf{Z}$, на котором основана реализация арифметики в большинстве современных компьютеров, порождает логическую матрицу Лукасевича. Найдены обогащения множества кольцевых операций, образующие базисы в матрице Лукасевича.

В главе 4 построена и теоретически обоснована универсальная теоретико-категорная семантика расширения технологий модульного проектирования систем приемами аспектно-ориентированного подхода. Расширение основано на семантике трассирования. Оно описано как обогащение моделей модулей разметкой их интерфейсов классами задач, образующими ее аспектную структуру. Аспектное связывание описано универсальной конструкцией в категории таких обогащенных моделей. Предложен новый способ разделения ответственности путем экспликации аспектной структуры – «подъема» разметки на уровень модулей. Введены свойства формальных технологий, характеризующие их аспектно-ориентированные расширения с точки зрения тривиальности, универсальности, полноты. Доказано, что переход от формальных технологий конфигурирования к технологиям специфицирования, состоящий в оснащении артефактов процессов проектирования систем интеграционными интерфейсами, представляет собой частный случай перехода от модульного проектирования к аспектно-ориентированному. Построена формальная технология проектирования технологий проектирования.

Глава 5 посвящена технологиям формального моделирования больших систем. Предложена процедура синтеза технологий, обеспечивающая хорошую трассируемость, и как следствие, широкие возможности аспектно-ориентированного расширения. Технологии, полученные при помощи этой процедуры, названы трансформационными. Для описания хорошо трассируемых трансформаций средствами теории категорий введено понятие M-инициального морфизма. В качестве приложений построены формальные технологии, позволяющие совместно моделировать данные помеченными множествами и сценарии исполнения процессов помеченными частично упорядоченными множествами. Сформулированы и доказаны критерии существования аспектного связывания и разделения ответственности в моделях данных и процессов. Предложен общий способ формального преобразования моделей динамических систем в помеченные структуры событий, образующие процессную модель архитектуры систем. Описаны результаты практической апробации подходов, предложенных в работе, в ходе проектирования больших информационно-управляющих систем для объектов топливно-энергетического комплекса.

Всего в работе построено и исследовано 5 нетривиальных формальных технологий, описывающих проектирование ключевых составляющих информационно-управляющих систем. Из них 3 оказались аспектно нетривиальными (и в работе подробно рассмотрены их аспектно-ориентированные расширения). Основные свойства исследованных технологий перечислены в следующей таблице (Таблица 1).

Таблица 1

Назначение формальной технологии	Аксиоматизация спецификаций	Проектирование вычислительных систем	Моделирование данных	Моделирование сценариев исполнения процессов	Синтез технологий конфигурирования
Обозначение	TS_σ	CSD, CSD$_*$	*DM*	*SM*	***SCONF***
Раздел работы	2.3	3.4	5.2	2.3, 5.3	2.6
Структурируема (определение 2.5)	Нет	Да	Да	Да	Нет
Скоординирована (определение 2.7)	Да	Да	Да	Нет	Нет
Поддерживает параллелизм (определение 2.8)	Да	Да	Да	Да	Да
(Ко)однородна (определения 2.11, 2.12)	Нет	Однородна	Кооднородна	Кооднородна	Кооднородна
Поддерживает трассирование (определение 4.1)	Да	Нет	Да	Да	Да
Элементарна (определение 4.2)	Да	Да	Да	Да	Да
Аспектно тривиальна (определение 4.6)	Да	Да	Нет	Нет	Нет
Аспектно универсальна (определение 4.14)	Да	Да	Да	Нет	Нет
Аспектно полна (определение 4.17)	Нет	Да	Да	Да	Нет
L-**трансформационна** для некоторого L (определение 5.2)	Нет	Нет	Да	Да	Нет

Глава 1. СИСТЕМНЫЙ АНАЛИЗ ЖИЗНЕННОГО ЦИКЛА БОЛЬШИХ ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ

1.1. Проблемы автоматизации управления большими объектами

Проведем анализ проблем, связанных с созданием большой системы, согласно работе [76]. Начнем с уточнения потребностей заказчика, приводящих к росту размеров системы. Будем рассматривать любой показатель размера системы как количество сущностей подходящего рода, употребляя понятие «сущность» в самом общем смысле – как денотат некоторого знака (простого или составного), пригодного к размещению в памяти компьютера. Очевидно, что значение показателя выводится из требований к системе, состоящих во включении в нее некоторого набора сущностей. Если такое требование содержит явное перечисление элементов набора, то оно не «опасно», поскольку объем перечисления не может быть очень большим. Резкий рост масштаба вызывается требованиями полноты (замкнутости) набора относительно тех или иных отношений между сущностями. Отношения могут классифицироваться по типам связей, которые порождают различные масштабные факторы – группы близких по природе показателей размера. Например, замыкание топологических связей, отражающих взаимную близость сущностей, размещенных в пространстве, приводит к высокой степени распределенности системы. Если подразумевается физическое пространство, то система должна охватывать все объекты, расположенные на некотором участке местности. Но часто выбираются другие классы пространств, такие как графы сетей распространения потоков ресурсов (материальных объектов, энергии, информации и т.д.). Близость вершин графа определяется длиной соединяющего их пути, так что топологическое замыкание сводится к обходу графа.

Размер системы может значительно возрасти также благодаря каузальным связям, если в ней требуется регистрировать причины (и/или следствия) сущностей, возникающих во времени. Для составных сущностей важно проследить их мереологические связи вида «часть–целое» – представить в системе все их части (и/или сущности более высокого порядка, в которые они входят в качестве частей). Способность сущностей описывать другие сущности (выступать в роли «метаданных») порождает дескриптивные связи, ответственные за рост сложности информационной структуры системы. Сущности, обладающие функциональностью, вступают в телеологические связи вида «цель–средство», замыкание которых (например, включение обеспечивающих процессов в контур автоматизации наряду с основными) приводит к росту количества функций системы и появлению конфликтующих целей.

Перечисленные типы связей, вместе с масштабными факторами и примерами массивов данных, приобретающих «большой» размер вследствие замыкания, приведены в таблице (Таблица 2).

Таблица 2

Тип связи	**Отношение**	**Масштабный фактор**	**Пример большого массива данных**
Топологическая	Близость	Распределенность	Реестр абонентов
Каузальная	Причина–следствие	Историчность	Журнал документооборота
Мереологическая	Часть–целое	Иерархичность	Организационная структура
Дескриптивная	Абстрактное–конкретное	Сложность	Классификатор видов объектов
Телеологическая	Цель–средство	Многофункциональность	Реестр оборудования

Чтобы быть большой, система не обязана обладать большими значениями всех масштабных факторов. Например, исторически первой сверхбольшой системой является глобальная сеть Интернет. Она предназначена для обмена данными в квазиреальном времени между узлами – аппаратными единицами, подключенными к каналам связи пропускной способностью 10^3–10^9 бит/с, охватывающим большинство мест присутствия человека и техники. Каналы образуют иерархию сегментов – подсетей (subnets), выражающуюся в позиционной структуре сетевого адреса узла [53]. В 2012 г. Интернет насчитывал почти 10^9 узлов [198] и 5×10^5 подсетей верхнего уровня [164]. Таким образом, сеть Интернет обладает большими значениями распределенности и иерархичности. Значения остальных факторов невелики, поскольку история, структура и функциональность узлов не имеет значения с точки зрения доступности сети. Тем не менее, даже в региональном масштабе создание и эксплуатация сегментов сети Интернет ставит ряд нетривиальных технических задач. В их решении на территории Сибирского региона принимал участие автор [149, 150, 151].

Высокая доступность сети предрасполагает к созданию глобальных хранилищ информации, снабженной большим объемом сложно организованных метаданных для повышения удобства просмотра, поиска и анализа. Такие хранилища представляют собой информационные системы (в узком смысле этого слова). В отличие от сети Интернет, они обладают большими значениями масштабного фактора сложности и в то же время практически не имеют иерархической организации (иерархия сетевого уровня не имеет для них значения). Примерами служат поисковые системы, социальные сети, крупные научные базы данных и др.

Для увеличения полезного эффекта от информационных систем их нагружают поддержкой разнообразных процессов сбора и обработки информации. Процессом в теории систем называется целое, части которого образуют непрерывные причинно-следственные цепочки и сами являются процессами [102, с.151]. Если видов процессов немного и они достаточно коротки (так что общее количество функций системы исчисляется десятками), то они привносят в

большую информационную систему только один дополнительный масштабный фактор – историчность, вызванную необходимостью хранить результаты и журналы исполнения процессов. Такие системы называются транзакционными (transaction processing systems), в отечественной литературе их также называют автоматизированными системами обработки данных (АСОД) [111]. К ним относятся международные платежные системы и магазины, электронные средства массовой информации, системы мониторинга, среды распределенных вычислений и др. На концептуальном уровне мы предлагаем рассматривать их как совокупности каналов сбора и распространения информации, образующих виртуальную сеть [86]. Такой подход развивает концепцию измерительного канала, зафиксированную в метрологических стандартах [35]: здесь каналы выполняют не только передачу данных, но и (де)мультиплексирование, перевод на разные языки и в разные форматы, журналирование, архивирование, резервирование системных ресурсов. Каждому процессу сопоставляется некоторая конфигурация таких функционально богатых информационных каналов. При участии автора был разработан ряд транзакционных систем такого типа в области автоматизации научно-исследовательской деятельности: портал математических ресурсов MathTree [16], электронный атлас мониторинга окружающей среды «Атмосферные аэрозоли Сибири» [36, 39, 212, 215], система коллективной разработки онтологий предметных областей ONTOGRID [45, 46, 47, 48], единый каталог сотрудников научных организаций [104].

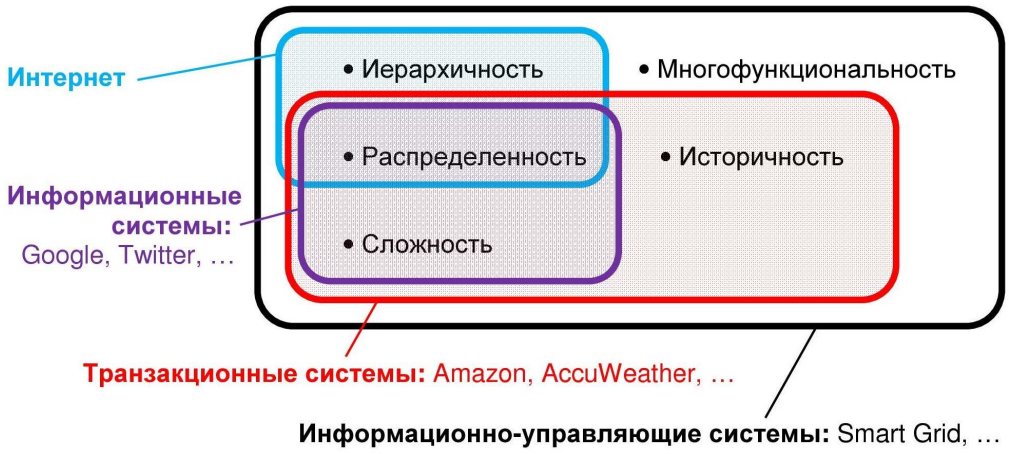
Чтобы превратить транзакционную систему в полноценную информационно-управляющую, в контур автоматизации необходимо включить все процессы, происходящие на объекте: основные, обеспечивающие и управляющие (а также обеспечивающие для обеспечивающих и т.д.). Если объект достаточно велик, то они характеризуются многочисленностью, разнонаправленностью, глубокой иерархической/сетевой организацией, поэтому автоматизированная система приобретает большие значения всех пяти масштабных факторов. К числу таких объектов относятся протяженные разветвленные сети производства, распространения, хранения, переработки, потребления потоков ресурсов: энергии, сырья, грузов. В качестве субъектов управления выступают энергоснабжающие организации, крупные предприятия нефтегазового сектора, металлургические холдинги полного цикла, железнодорожные монополии, муниципалитеты больших городов и др.

Основными критериями эффективности управления сетевым объектом являются максимизация объема полезного отпуска ресурса, надежности и качества ресурсоснабжения. Однако для отдельных участников процессов на первое место могут выходить совсем другие критерии: например некоторые потребители стремятся экономить – получать как можно меньше ресурса. Кроме того, большой размер объекта вынуждает разделять его на сегменты, передаваемые в управление разным организационным единицам. Границы сегментов обычно проводятся исходя

из распределения собственности, истории взаимоотношений и т.п., поэтому часто не совпадают с естественными границами технологических участков. Возникает концептуальный разрыв между уровнем управления технологическими процессами («уровень АСУТП») и уровнем корпоративного управления («уровень АСУП»), усугубляющий разноречивость требований участников процессов. Часто не удается выработать даже единый язык общения между ними: реестр технологических узлов объекта значительно отличается от бухгалтерского реестра основных средств. Тем не менее, критерии эффективности необходимо балансировать, поскольку автоматизируемые процессы объединяют участников, несмотря на различия их функциональной и административной подчиненности, и имеют сквозной характер [43, с.26].

Количественный «портрет» типичной информационно-управляющей системы для большого сетевого объекта выглядит следующим образом. Фактор распределенности задается составом объекта и насчитывает 10^3–10^7 единиц. Глубина хранения истории обычно составляет 10^3–10^5 записей на единицу, в зависимости от периода измерения показателей состояния объекта. Иерархия и структурная сложность насчитывают по 10^1–10^2 уровней (эти значения можно считать «большими», поскольку человеческий мозг очень плохо справляется с навигацией по уровням целостности и абстрактности [101]). Количество функций составляет 10^3–10^4 штук. Характерным примером системы такого масштаба служит Smart Grid («умная» сеть) – система сквозной автоматизации крупной электроэнергетической сети, направленная на балансирование экономических интересов и технических возможностей поставщиков электроэнергии с разноречивыми настроениями потребителей [142].

Таким образом, получается одна из возможных классификаций систем, основанная на номенклатуре (природе) масштабных факторов, имеющих большие значения. Эта классификация изображена на следующей схеме, где наименования факторов соответствуют третьей колонке вышеприведенной таблицы (см. Таблица 2).



Эта схема помогает увидеть основную концептуальную проблему, связанную с поддержкой больших значений масштаба. Дело в том, что артефакты процессов инженерии информационных систем могут вступать друг с другом только в отношения «часть–целое» и «абстрактное–конкретное». Действительно, все действия, выполняемые в процессе создания системы, сводятся к (де)композиции составных артефактов (например, детализация требований, сборка приложения из объектных модулей) и трансформации (refinement) абстрактных артефактов в конкретные (например, реализация спецификации на языке программирования) [244]. В то же время сущности, составляющие объект управления, связаны отношениями всех пяти типов. Поэтому разработчикам информационно-управляющей системы приходится выражать все возможные связи сущностей через мереологические и дескриптивные связи отвечающих им системных единиц.

Для топологических и каузальных связей эту проблему можно решить путем сохранения их в базе данных: они отображаются на мереологические связи структур данных. Эффективность их хранения и обработки в большом масштабе достигается путем экстенсивного наращивания количества и мощности вычислительных узлов, параллельно выполняющих навигацию по ним согласно потребностям автоматизированных функций. Разнообразные средства разработки таких узлов предоставляются технологиями параллельных и распределенных вычислений [26]. Однако с телеологическими связями поступить подобным образом в общем случае не удается, поскольку с ростом масштаба они неизбежно приобретают конфликтность, запутанность и изменчивость. В свою очередь, схемы данных рассчитаны на относительно редкие изменения, поэтому оформление изменчивых отношений «цель–средство» явными ссылками между информационными объектами приводит к необходимости постоянно производить ресурсоемкие, длительные, чреватые ошибками операции перестройки хранилища данных. Не менее трудно выразить крупномасштабные телеологические связи через иерархию сборки модулей в систему, поскольку модуль создается для автоматизации заранее заданных задач, имеет заранее заданный интерфейс доступа и поставляется в виде цельного фрагмента исполняемого кода. При заранее фиксированных и медленно меняющихся требованиях к системе разделение на модули приводит к уменьшению сложности и ускорению создания системы [95]. Однако чтобы получать полезную отдачу от модулей в априори не известных и постоянно меняющихся условиях их использования, приходится проникать в их внутреннюю структуру в обход интерфейса и даже принудительно «рассеивать» их по системе.

1.2. Ограничения качества больших информационно-управляющих систем

Несмотря на трудности выявления требований в условиях роста масштаба, для любой сколь угодно большой системы должны быть заданы свойства, определяющие ее функциональное назначение и качество [258]. Они характеризуют ее способность удовлетворять потребности пользователей – готовность к разнообразным воздействиям, которым она подвергается со стороны внешнего окружения в течение эксплуатации. По отношению к информационно-управляющей системе, как и к любому автомату, важнейшим воздействием служит запрос на выполнение той или иной функции. Однако ничуть не меньшее значение имеют другие, «нефункциональные», воздействия: инсталляция, изучение, модификация, наложение ограничений по ресурсам и даже нанесение вреда [78]. Иллюстрацией их важности служит печально известный пример системы автоматизации работы лондонской службы «скорой помощи». Эта система вела себя в полном соответствии с функциональными требованиями, но оказалась непригодной к практическому использованию из-за несоответствия пользовательского интерфейса привычкам диспетчеров и постоянных потерь данных в ненадежной среде городских беспроводных коммуникаций [157].

При формулировке нефункциональных требований пользователи обычно испытывают значительные затруднения. Это связано с отсутствием достаточного опыта работы с информационно-управляющими системами, который позволял бы на интуитивном уровне чувствовать баланс между его возможностями и ограничениями. Например, заказчик веб-портала требует от него способности одновременно обслуживать 1 млн запросов с задержкой не более 1 мс. При этом он выделяет более чем скромный сервер и удивляется, что разработчики не могут его «как следует настроить». В отношении материальной продукции дело обстоит иначе, поскольку необходимый опыт накоплен в течение многих тысячелетий жизни в физическом мире. Так что нефункциональные требования даже при «небольшом» масштабе плохо поддаются точному описанию и проверке, часто не допускают количественной оценки, вступают в противоречие с функциональными и друг с другом. Чтобы формулировать, обеспечивать и проверять требования к программной продукции, нужны специальные подходы, учитывающие ее специфику по сравнению с материальными изделиями. Наиболее широко распространенные из таких подходов становятся основой стандартов качества программного обеспечения (software quality).

Рассмотрим международный стандарт ISO/IEC 9126 «Information Technology – Software Product Evaluation – Software Quality Characteristics and Guidelines for their use» (в России он принят под названием ГОСТ Р ИСО 9126-93 «Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению» [32]). Он предпи-

сывает задавать требования качества (quality requirements) как ограничения на значения показателей (attributes), подлежащих выявлению и измерению. Обычно требование имеет вид рейтинга (rating level), когда диапазон значений делится на поддиапазоны, задающие степень «качественности» изделия. Процедура оценки качества, которую проходит законченное изделие либо версия, сводится к измерению и ранжированию всего набора показателей, по результатам которого составляется заключение. В стандарте ISO/IEC 9126 показатели сгруппированы в шесть характеристик (characteristics), соответствующих готовности к типовым видам воздействия на программное обеспечение, выявленным из накопленного опыта его эксплуатации: функциональные возможности, надежность, практичность (удобство), эффективность, сопровождаемость, мобильность (переносимость). В качестве приложения к стандарту приведены рекомендации по выделению 21 подхарактеристики (subcharacteristics), к которым относятся конкретные показатели.

Систематическое применение стандартов качества особенно важно в процессе создания больших систем, в который вовлекается большое количество разнородных разработчиков. Здесь выявление состава и диапазонов допустимых значений показателей качества осложняется ввиду проблем, описанных в разд.1.1. Из-за них многие показатели попадают на «метауровень», характеризуя не обеспечение конкретных потребностей, а возможность удовлетворить широкий класс разноречивых требований [237]. Достижение высоких значений таких показателей придает системе сходство с живым организмом: в ней появляются механизмы ассимиляции, самовосстановления, целенаправленного поведения.

Начнем с рассмотрения функциональных характеристик. Конечно, для большой системы, как и для «небольшой», задается ряд функций, отражающих ее назначение и возможности. Однако потребительский эффект от их реализации не будет достигнут, если не удастся организовать потоки выполнения функций «вдоль» изменчивых сквозных процессов. Чисто функциональная организация системы не позволяет обеспечить управляемость большим объектом, поскольку вне системы практически невозможно проверять факты выполнения функций многочисленными исполнителями, относящимся к разным организационным единицам. Необходимо встроить в систему средства компоновки схем исполнения и контроля сквозных процессов, причем должна быть предусмотрена возможность изменять ход выполнения уже идущего процесса (в рамках, не нарушающих его корректность) [167]. Конкретные функциональные требования следует формулировать в привязке к шагам процессов, их входам и выходам.

С точки зрения надежности система, состоящая из большого количества аппаратных и программных узлов, отличается тем, что в любой момент времени некоторая часть их находится в состоянии отказа. Система должна сохранять способность функционировать в таких условиях и содержать в себе механизмы самовосстановления, такие как нахождение резервных уз-

лов и назначение им задач, выполнявшихся отказавшими узлами. Система, обладающая подобными способностями, называется живучей, поскольку они присущи живым организмам и экосистемам. Активно разрабатываются подходы к имитации заложенных в биосистемах механизмов обеспечения высокой доступности [258].

Уровень практичности большой системы определяется возможностью предоставить информацию каждому пользователю в каждый момент времени в формате и объеме, не превосходящем порог постижимости и в то же время достаточном для выполнения его действий. Система должна защищать пользователя от «лавины данных» (data deluge) [195], постоянно поступающих из ее многочисленных компонентов-«сенсоров». Для этого необходимо вести текущий контекст действий пользователя и в соответствии с ним фильтровать терминологию и наполнение информационных видеокадров. Вместе с тем, следует постоянно показывать пользователю индикаторы и напоминания, фокусирующие его внимание на самых важных данных. Также целесообразно визуализировать мнемонические схемы объектов и процессов с переменной степенью детализации (zoom) – от очень грубого эскиза объекта управления в целом «с высоты птичьего полета» до детального изображения той части, которая в данный момент интересует пользователя [112].

Интенсивность «лавины данных» также оказывает критическое влияние на эффективность больших систем. Длительность циклов обработки данных, затрагивающих систему в целом, как правило, достаточно велика – от часа до месяца (10^3–10^6 с), поскольку более оперативные циклы отрабатываются внутри компонентов низкого уровня [160]. Однако большое количество источников данных (до 10^7 ед.) приводит к необходимости параллельно выполнять соответствующее количество циклов. Такой параллелизм в сочетании с пространственной распределенностью источников требует распараллеливания функций хранения и обработки данных.

Потребность в высоком уровне сопровождаемости большой системы диктуется неопределенностью начальных требований, вследствие которой основной объем работ по достижению удовлетворенности пользователей ложится на процесс сопровождения. Ввиду большого размера системы, он становится непрерывным – не удается разделить его на дискретные шаги, состоящие в поставке очередных версий системы, прошедших лабораторные тесты на предмет удовлетворения фиксированным наборам требований. Как указывалось в разд.1.1, разноречивость требований вынуждает привлекать в ходе сопровождения для развития системы приемы комплексирования, выходящие за рамки подключения модулей с фиксированным интерфейсом. Такие приемы напоминают механизмы ассимиляции в биологических системах.

Показатели мобильности (переносимости) большой системы задаются необходимостью включать в ее состав компоненты, разработанные при помощи разнородных парадигм программирования. Разнородность компонентов диктуется многообразием задач, выставляемых раз-

личными классами пользователей – от регистрации состояния объекта в реальном времени до многошагового электронного документооборота. Для программной поддержки ряда классов задач имеются специализированные проблемно-ориентированные подходы, включающие специализированные нотации моделирования, инструменты проектирования, языки программирования. Большая система должна предоставлять компонентам, созданным при помощи таких разнородных подходов, универсальную среду для развертывания, интеграции и взаимозаменяемости.

Таким образом, получается следующая «порождающая» абстрактная типовая модель качества больших информационно-управляющих систем (Таблица 3), где для каждой характеристики определен ключевой «метапоказатель», путем уточнения которого для частного объекта управления порождаются показатели качества частной системы.

Таблица 3

Характеристика качества по ISO/IEC 9126	**Порождающий показатель**
Функциональные возможности (functionality)	Поддержка изменчивых сквозных процессов
Надежность (reliability)	Способность функционировать в условиях постоянных частичных отказов
Практичность (usability)	Динамическая подстройка визуализации информации под текущий контекст пользователя
Эффективность (efficiency)	Параллельное выполнение циклов сбора и обработки данных от множества распределенных источников
Сопровождаемость (maintainability)	Способность к непрерывной ассимиляции компонентов
Мобильность (portability)	Совместное функционирование компонентов, разработанных в рамках разнородных парадигм

В настоящее время Международная организация стандартов ISO разрабатывает обновленную целостную модель описания и оценки качества, известную под названием ISO 25000 SQuaRE («Systems and software Quality Requirements and Evaluation» – «Требования качества к программному обеспечению и их оценка»). В частности, в 2011 г. стандарт ISO/IEC 9126 был заменен стандартом ISO/IEC 25010:2011 «Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models». В нем характеристика «Функциональные возможности» расщеплена на три: функциональная пригодность (functional suitability), совместимость (compatibility), защищенность (security). В стандарте ISO/IEC 9126 они выступали в качестве подхарактеристик характеристики «Функциональные возможности». Соответственно, порождающий показатель этой характеристики можно расщепить на три: один характеризует качество средств автоматизированной поддержки отдельных шагов процессов, другой – возможности сборки этих средств в комплексы автоматизации сквозных процессов, третий – разграничение прав участников процессов в точном соответствии с их функциями в процессе. Другие отличия стандарта ISO/IEC 25010:2011 от ISO/IEC 9126,

такие как уточнение наименований некоторых характеристик и добавление/удаление подхарактеристик, не требуют существенной переработки нашей типовой модели качества.

Значения показателей качества должны постоянно измеряться и проверяться на предмет нахождения в допустимом диапазоне в течение всего жизненного цикла системы. Процедуры измерения, проверки и внесения корректировок при нарушениях становятся все более трудоемкими по мере роста масштаба системы, поэтому нуждаются в автоматизации. Чтобы она была возможной, необходимо построить формальную модель качества (quality model) – описать структуру показателей на формальном языке, не допускающем многозначных толкований. Для записи показателей можно использовать те или иные формальные методы инженерии программного обеспечения [73], однако наибольший интерес представляют специализированные языки, совместимые со стандартами качества. Среди таких языков мы выделим NoFun [156], основанный на стандарте ISO/IEC 9126. Он предоставляет выразительные средства для структурированного типизированного описания показателей качества, а также для формулировки требований в виде ограничений на значения (метрики) этих показателей. Нотация напоминает декларативный фрагмент современного языка программирования. Система типов (domains) включает примитивные типы (Boolean, Integer, Real, String), перечисления (enumeration), структуры (tuple), множества (set) и даже функции (function). Можно задавать подтипы (например, положительные числа) путем наложения ограничений. Характеристики качества изделий объединяются в модули (modules), допускающие иерархическую организацию путем наследования. Для описания правил задания значений можно использовать логические, арифметические, строковые и теоретико-множественные операции.

Чтобы упростить реализацию средств машинной обработки моделей качества, нами разработана структурная нотация представления конструкций языка NoFun, основанная на языке XML [216]. Здесь для записи арифметических и теоретико-множественных конструкций, с помощью которых задаются метрики и ограничения, привлекается стандартная XML-нотация MathML [223]. Рассмотрим его применение для описания показателей надежности информационно-управляющей системы. Например, время восстановления, измеряемое в часах, задается числовой функцией, определенной на иерархически организованном множестве классов (видов, типов, экземпляров) компонентов. Предположим, что для класса «измерительный комплекс» оно составляет 1 час. «Зная» это значение, система может автоматически по факту прекращения поступления результатов измерений составить расписание оповещений эксплуатирующего персонала и корректирующих действий, таких как переход на резервные приборы. Приведем для этого случая (частичные) декларации трех модулей: тип компонентов, показатели качества, требования надежности. В них элементы нотации xNoFun набраны жирным шрифтом.

```
<domainModule name="D_SYSTEM_COMPONENTS">
 <explanation>Classes of system components</explanation>
 <domain name="D_System_components">
  <defined>
   <declare xmlns="http://www.w3.org/1998/Math/MathML">
    <ci type="set">D_System_components</ci>
    <set>
     <ci>Data_metering_complex</ci>
     ...
    </set>
   </declare>
  </defined>
 </domain>
</domainModule>

<attributeModule name="A_QUALITY_INDEXES">
 <attribute name="A_Recovery_time">
  <declared>
   <declare xmlns="http://www.w3.org/1998/Math/MathML">
    <ci type="function">A_Recovery_time</ci>
    <apply><eq/>
     <apply><domain/><ci>A_Recovery_time</ci></apply>
     <ci>D_System_components</ci>
    </apply>
    <apply><eq/>
     <apply><codomain/><ci>A_Recovery_time</ci></apply>
     <integers/>
    </apply>
   </declare>
  </declared>
 </attribute>
 ...
</attributeModule>

<requirementModule name="R_RELIABILITY_REQUIREMENTS">
 <requirement name="req-rely-comp-3" category="important">
  <defined>
   <apply><leq/>
    <apply>
     <fn><ci>A_Recovery_time</ci></fn>
     <ci>Data_metering_complex</ci>
    </apply>
    <cn>1</cn>
   </apply>
  </defined>
 </requirement>
 ...
</requirementModule>
```

Полное формальное описание модели качества в таком ключе можно рассматривать как машинно-читаемую спецификацию целей системы. Поэтому ее наличие открывает богатые возможности для привлечения агентных технологий и платформ. Для этого модель качества должна быть дополнена набором терминов и знаний о предметной области объекта управления, достаточным, чтобы агенты могли эффективно взаимодействовать и принимать решения о достижении целей [237]. Такие наборы называются онтологиями (ontology) [185]. Формирование

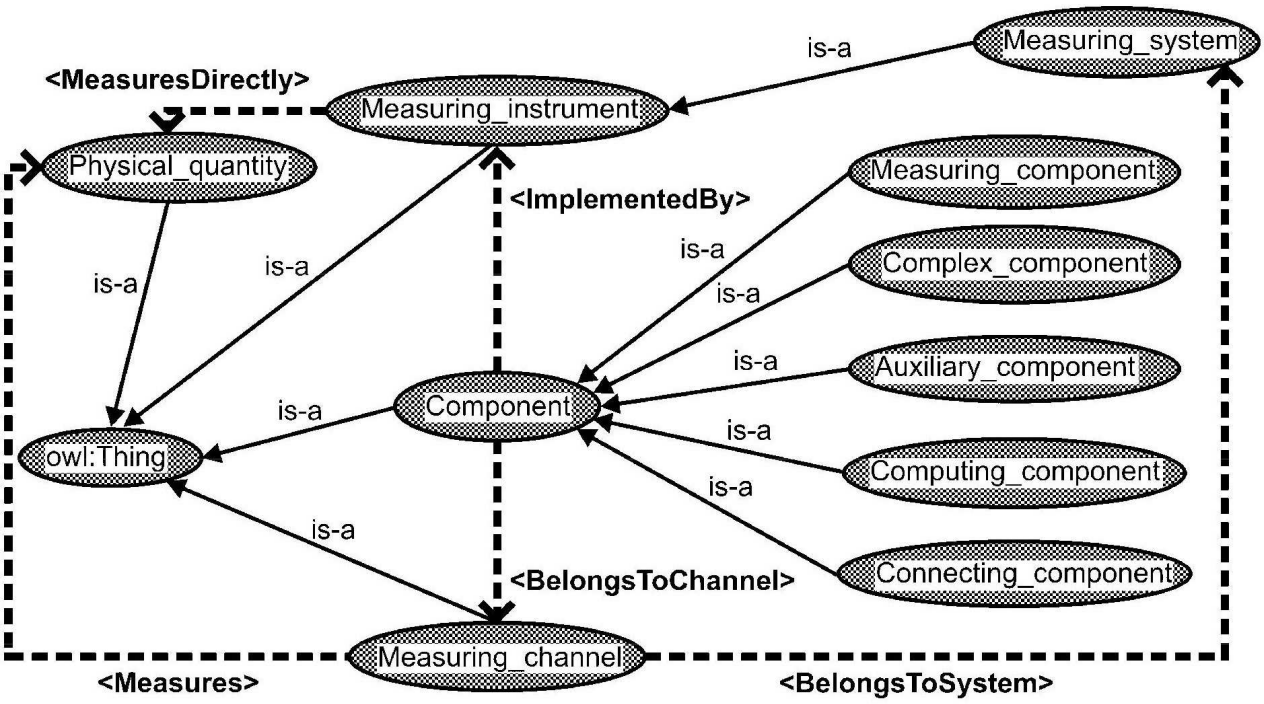
онтологии является чрезвычайно сложной и ресурсоемкой процедурой, которую невозможно провести в рамках цикла разработки отдельной системы, даже большой. Для этого организуются специальные научные исследования, результаты которых фиксируются в международных стандартах, содержащих готовые формальные описания онтологий. В настоящее время такие исследования активно ведутся в областях, обладающих собственными формализованными языками: математика, химия и т.д. Предлагаются нотации представления онтологий в виде, удобном для применения в информационных технологиях, среди которых на роль стандарта претендует структурный язык Web Ontology Language (OWL) [231], основанный на языке XML. Онтологии, составленные на языке OWL, легко расширяются новыми фрагментами при помощи ссылочных связей (принцип «открытого мира»).

Существуют средства автоматизации процесса разработки онтологии, например графический редактор OWL-документов Protégé [217], предоставляющий визуальную среду для просмотра и изменения онтологии. Однако он изначально не был приспособлен для коллективной работы распределенных групп экспертов, которые могут одновременно редактировать разные фрагменты онтологии, рецензировать результаты других экспертов, и даже конфликтовать за один и тот же фрагмент. Только в результате такой коллективной работы могут создаваться онтологии, заслуживающие доверия. Ясно, что ее эффективное выполнение невозможно без полномасштабной автоматизации: здесь требуется распределенная транзакционная система типа электронного документооборота, оперирующая с предложениями на изменение фрагментов онтологии и периодически консолидирующая взаимно согласованные наборы реализованных предложений в стабильные версии готовой онтологии. Чтобы обеспечить высокую динамику вовлечения групп, работающих в разных странах и в разнородном системном окружении, автор и его коллеги реализовали такую систему на базе технологий Grid, оформив ее компоненты в виде грид-сервисов [45, 46, 47, 48]. Отметим, что в настоящее время инструмент для коллективной разработки онтологий предлагают и создатели редактора Protégé [257].

В качестве примера рассмотрим разработанный нами фрагмент онтологии предметной области «управление большим объектом», описывающий измерение физических величин – одну из ключевых базовых функций системы управления [72]. Источником терминов для нее естественным образом служит метрология – наука о выполнении измерений. Ее положения стандартизованы и изложены в нормативных документах, в частности ключевые термины и определения зафиксированы в стандарте ГОСТ Р 8.596-2002 «Метрологическое обеспечение измерительных систем. Основные положения» [35]. Он определяет измерительную систему (`Measuring_system`) как совокупность компонентов (`Component`): измерительных (`Measuring_component`), вычислительных (`Computing_component`), связующих (`Connecting_component`), вспомогательных (`Auxiliary_component`), комплексных

(Complex_component), образующих измерительные каналы. Измерительный канал (Measuring_channel) – это часть измерительной системы, выполняющая законченную функцию от восприятия измеряемой физической величины (Physical_quantity) до получения результата ее измерения (эта функция обозначается отношением (<Measures>). Точка измерения на объекте управления – это место подключения измерительного канала-источника первичных данных, так что измерительные каналы выступают в роли контейнеров метаданных результатов измерения. Каждый измерительный канал входит в систему (<BelongsToSystem>), а компонент – в один или несколько каналов (<BelongsToChannel>). Физически компоненты измерительных каналов реализуются (<ImplementedBy>) средствами измерений (Measuring_instrument), непосредственно измеряющих величины (<MeasuresDirectly>), причем измерительная система как целое также является сложным средством измерений. Каждая из вышеназванных сущностей является вещью (Thing) – базовым понятием языка OWL.

Общая схема получившейся базовой онтологии представлена на рисунке: понятия обозначены овалами, классификационные отношения «частное–общее» (*is-a*) – сплошными линиями, прочие отношения – пунктирными линиями.



Отношения между понятиями используются в качестве «строительного материала» для формирования правил вывода предметных знаний. Логическое исчисление, позволяющее записывать правила на языке OWL, называется дескриптивной логикой (descriptive logic). Например, условие физической реализуемости измерительного канала задается следующей импликативной конструкцией: если канал измеряет некоторую физическую величину, то он содержит

компонент, реализованный средством измерения, непосредственно измеряющим ее [85]. Символически это правило выглядит так:

```
∀M, Q: Measures(M, Q) ⇒
       ∃C, I: BelongsToChannel(C, M) &
                ImplementedBy(C, I)    &
              MeasuresDirectly(I, Q)
```

Чтобы иметь возможность описать все классы задач и свойств информационно-управляющей системы, к базовой онтологии подключаются другие [72]. Например, понятие физической величины конкретизируется понятиями, зафиксированными в онтологии физических величин и единиц измерения согласно Международной системе единиц. Общее понятие средства измерений конкретизируется видами и техническими характеристиками приборов. Для задач управления энергообеспечением большой набор онтологий разработан в рамках модели CIM [143]. Чтобы можно было описывать задачи уровня АСУП, добавляются онтологии понятий организационного и процессного управления, основанные на международных стандартах электронного обмена деловой информацией типа ebXML [210]. Благодаря им, в частности, появляется возможность задавать организационную принадлежность измерительных каналов и процессы управления их жизненным циклом, наделяя систему функциями класса CALS (Continuous Acquisition and Lifecycle Support [105]).

1.3. Принципы рационального проектирования больших систем

Решения о наиболее рациональных способах организации системы, обеспечивающих заданные значения показателей качества, принимаются в ходе ее проектирования. Цель проектирования состоит в формировании архитектуры системы. Понятие архитектуры применительно к программным системам определено в стандарте ISO/IEC/IEEE 42010:2011 «Systems and software engineering – Architecture description» следующим образом: архитектура – это «фундаментальные концепции и свойства системы, находящейся в своем окружении, воплощенные в ее элементах, отношениях и принципах ее конструирования и развития» (fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution) [199, п.3.2]. Архитектура фиксируется в разнообразных описаниях – «осязаемых» результатах работ проектировщиков. Выбор нотации для каждого описания обусловлен стилем проектирования. В [130, с.80] перечислены следующие стили:

- стиль взаимодействия, выражающий взаимодействие в виде диаграмм вариантов использования, схем процессов и т.п.;

- алгоритмический стиль, задающий алгоритмы вычислений при помощи псевдокодов;
- информационно-ориентированный стиль, описывающий модели данных в виде диаграмм «сущность–связь»;
- потоковый стиль, фиксирующий потоковую структуру системы в диаграммах потоков данных.

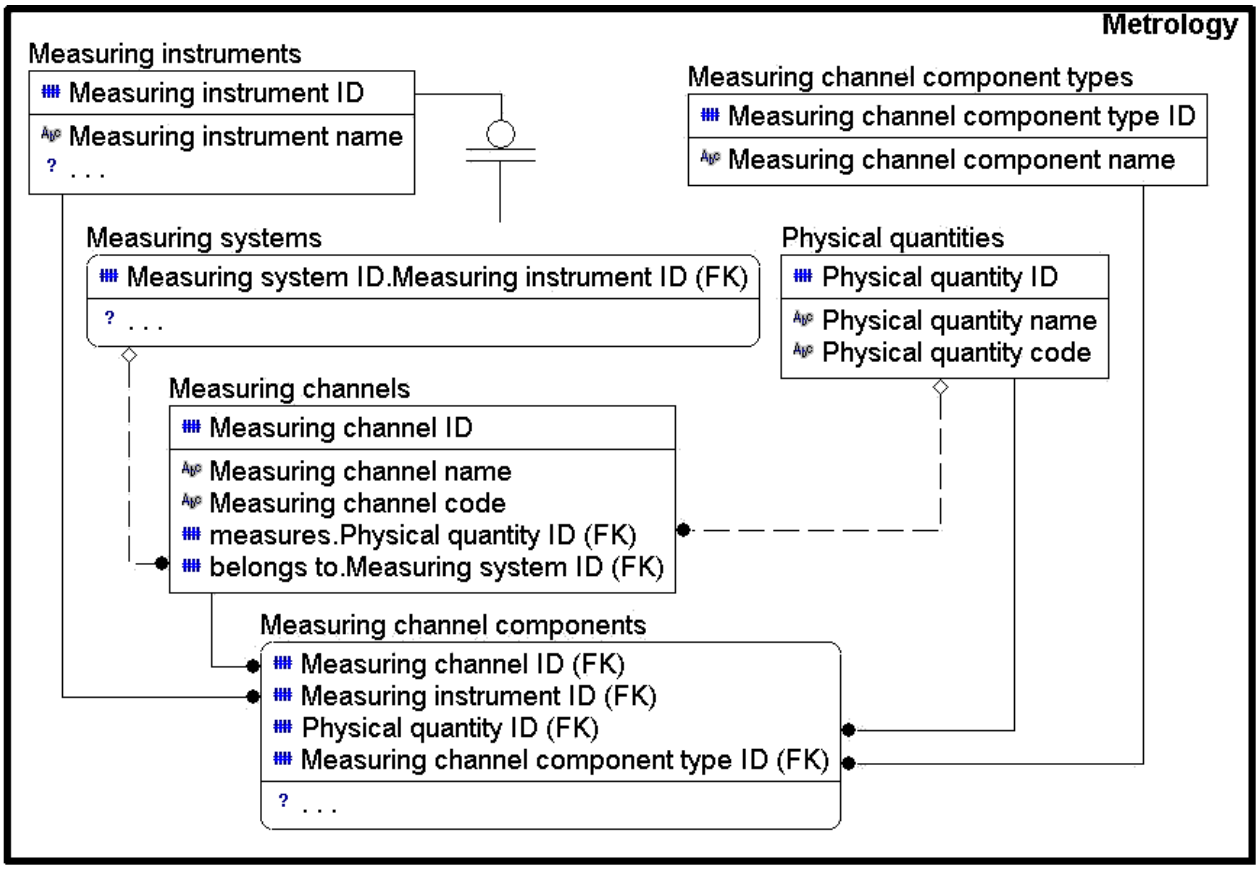
Каждый стиль отвечает одному из четырех способов работы с информацией, доступных программам: ввод/вывод, аналитическая обработка, хранение, передача [102]. Многие технологии программирования специально ориентированы на поддержку того или иного стиля. Они предоставляют средства переработки моделей, выполненных в этом стиле, в компоненты различных видов: сервисы, агенты, драйверы, портлеты, компоненты EJB (Enterprise Java Beans), библиотеки, хранилища данных и др. Компоненты снабжаются интерфейсами – специально подготовленными частями, предназначенными для интеграции с другими компонентами, в том числе разработанными при помощи других технологий. Система как целое комплексируется из компонентов, обращающихся друг к другу через интерфейсы [129].

Стиль взаимодействия применяется для проектирования компонентов большой информационно-управляющей системы, которые обслуживают три вида ее контрагентов: измерительные приборы и регуляторы, пользователей, смежные автоматизированные системы управления. Контрагенты находятся вне границ системы, поэтому каждый сеанс взаимодействия контролируется компонентами защиты информации. Компоненты взаимодействия с приборами предназначены для сбора информации о физическом окружении системы и телеуправления исполнительными устройствами. Они относятся к уровню АСУТП, поэтому реализуются на базе средств критического (dependable) характера, таких как системы класса SCADA. Компоненты взаимодействия с пользователями отвечают за требования эргономики (удобства), среди которых важную роль играет доступность, поэтому в качестве среды для их развертывания выбирается портал на базе web-технологий. Взаимодействие со смежными системами обычно выполняется на уровне АСУП, поэтому для его реализации привлекаются интеграционные платформы на базе сервисно-ориентированной архитектуры, организующие обмен структурированными электронными документами. При помощи таких платформ целесообразно также организовывать взаимодействие между компонентами системы, принадлежащими разным крупным архитектурным единицам (подсистемам). Источником корректных терминов для наименований элементов протоколов взаимодействия, форм пользовательского интерфейса и структурных единиц документов способна выступать онтология предметной области. Например, в XML-формате 80020, предназначенном для обмена данными о потреблении электрической энергии

[1], тег, к которому привязываются значения количества энергии, называется `measuringchannel`.

Алгоритмические компоненты в основном предназначены для расчета и анализа данных в целях формирования достоверной картины состояния объекта и выработки управляющих воздействий на него. Основной цикл расчета воспроизводит подход системного анализа «в количественном выражении»: вычисляются показатели эффективности автоматизированных процессов, выявляются недопустимые расхождения между их фактическими и возможными значениями, рассчитываются наиболее рациональные корректирующие воздействия [106]. Дополнительно решаются вспомогательные расчетные и аналитические задачи. Здесь привлекаются разнообразные вычислительные средства, даже простое перечисление которых выходит далеко за рамки настоящей работы. Чтобы обеспечить эффективность, целесообразно развертывать их в динамической гетерогенной среде распределенных вычислений типа Grid или облака (cloud). Здесь возникают трудоемкие задачи отображения алгоритмов на архитектуру вычислительной среды, включающие редукцию на конечные множества доступных ресурсов памяти, подбор рациональных средств управления потоком вычислений, комплексирование вычислительных компонентов в целостные системы, оценку и минимизацию сложности вычислений [60].

Основным информационным компонентом системы является информационная модель объекта управления (ИМОУ) – всеобъемлющее хранилище метаданных автоматизированных процессов. Она может быть получена из онтологии предметной области, путем трансформации в диаграмму «сущность–связь», реализуемую средствами реляционной СУБД. В состав модели объекта входят реестры организационной структуры, оборудования и приборов, каналов измерения и управления, процессов, расчетных алгоритмов [9], поэтому ее также называют паспортом объекта. Для нормирования записей реестров заводятся всевозможные справочники и классификаторы. Например, базовая онтология, приведенная в разд.1.2, порождает реестр измерительных каналов, диаграмма «сущность–связь» которого представлена ниже [72].



Сравнение этой диаграммы со схемой онтологии позволяет увидеть «смещение центра тяжести», происходящее при переходе от анализа к проектированию. Если онтология нацелена на представление предметной области в наиболее понятном нормированном виде, то структура базы данных должна в первую очередь обеспечивать эффективность выполнения запросов на выборку и модификацию данных. В соответствии с этим, разбиение онтологического класса «Компонент измерительного канала» (`Component`) на подклассы реализовано посредством справочника типов компонентов. Кроме того, онтологические отношения «Компонент реализуется средством измерения» (`<ImplementedBy>`) и «Средство измерения измеряет физическую величину» (`<MeasuresDirectly>`) свернуты путем композиции в одну реляционную ссылку, направленную из реестра компонентов в справочник физических величин. В результате компонентный состав измерительных каналов определяется четырехместным отношением

$$\mathrm{MCC} \subseteq \mathrm{MC} \times \mathrm{MI} \times \mathrm{PQ} \times \mathrm{MCCT},$$

где MCC (`Measuring channel components`) – множество компонентов; MC (`Measuring channels`) – множество измерительных каналов; MI (`Measuring instruments`) – множество средств измерения; PQ (`Physical quantities`) – множество физических величин; MCCT (`Measuring channel component types`) – множество типов компонентов.

Для выборки информации из модели целесообразно предусмотреть универсальный программный интерфейс – профиль выборки данных [64]. Он представляет собой реляционную структуру, описывающую совокупность извлекаемых сущностей путем явного перечисления и/или по ограничительным критериям (например «все трансформаторы, установленные на заданной подстанции»). Каждый профиль имеет наименование и целевую функцию (класс задач, для которых требуется выборка). Посредством профилей можно задавать контрольные списки доступа субъектов к информации (access control lists), наборы технических средств для группового управления, предпочтения пользователей, правила заполнения генерируемых отчетов, входные данные для алгоритмов расчета и анализа, шаблоны обработки событий. Чтобы составлять сложные профили, реализуются теоретико-множественные операции над ними. Компонент извлечения данных, удовлетворяющих заданному профилю, представляет собой, по существу, решатель задач в ограничениях (constraint solver), реализованный средствами реляционной СУБД. Он может быть достаточно ресурсоемким, поскольку выполняет, в частности, навигацию по связям, ответственным за большие значения масштабных факторов.

При извлечении данных из хранилища большой системы возникает проблема обеспечения их взаимной согласованности. Она вызвана нестабильностью большого объекта управления: для достаточной широкой выборки часто выполняется соотношение

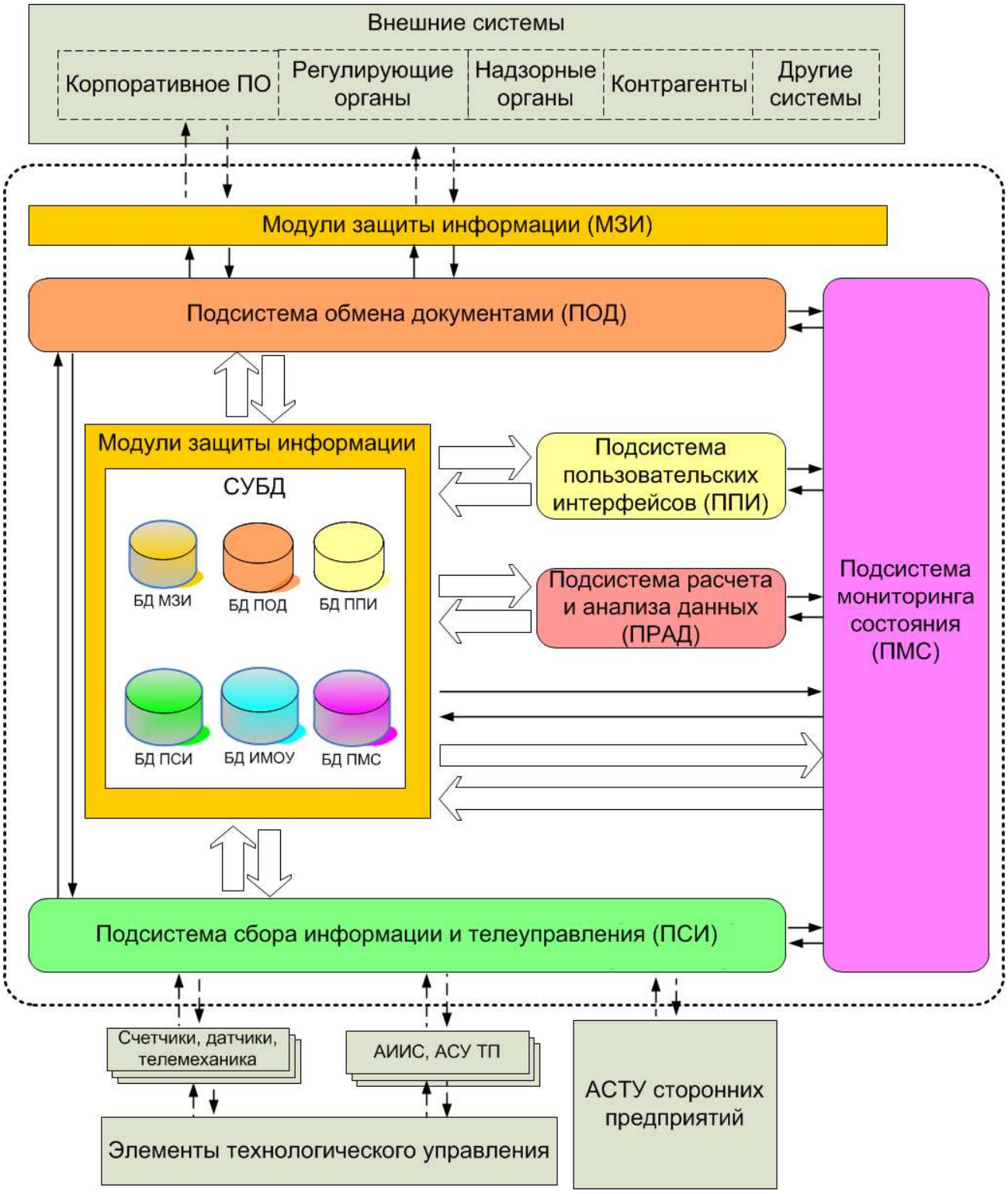
$$T_{стаб} < T_{выб} < T_{изм} ,$$

где $T_{стаб}$ – характерный период времени, в течение которого ни одна из выбранных сущностей не изменяет свое состояние, $T_{выб}$ – характерное время жизни выборки, $T_{изм}$ – характерная длительность полного цикла изменения участка объекта. Иными словами, выборка строится и обрабатывается настолько медленно, что нельзя пренебречь изменением состояния объекта в течение времени ее жизни, и вместе с тем настолько быстро, что нельзя считать изменения атомарными и целостными. Сходная проблема встречается при проектировании средств синхронизации в больших распределенных системах: невозможно установить глобальное время, единое для всех компонентов [122]. Решатель профилей выборки данных должен обнаруживать и игнорировать частичные, незавершенные изменения, формируя самосогласованный условно-постоянный образ запрашиваемой части объекта, пригодный в качестве полной информационной базы для выполнения задачи выборки [7, 141]. Конечно, это может привести к снижению уровня актуальности решения задачи (например, прогноз потребления энергоустановки может не учесть аварию, вызвавшую ее полное отключение), однако позволяет избежать трудноуловимых ошибок, вызванных несовместимостью состояний различных элементов выборки.

Рассмотрим применение потокового стиля при проектировании большой системы. В ходе выполнения автоматизированных процессов ее компоненты взаимодействуют между собой, обмениваясь данными и командами. Существует два основных потока обмена: «восхождение»

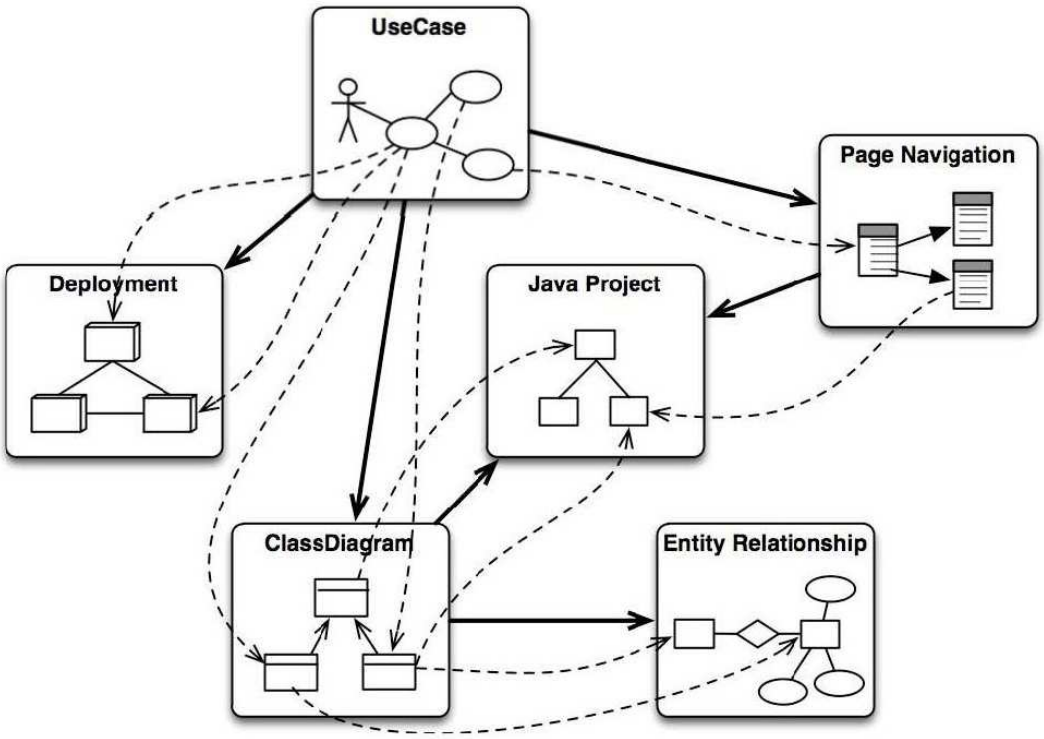
параметров состояния объекта управления с уровня АСУТП на уровень АСУП, и «нисхождение» управляющих воздействий в обратном порядке [111]. По ходу потоков информация форматируется и нормируется, сохраняется в базах данных, агрегируется и детализируется, фильтруется в соответствие с политикой безопасности, визуализируется в разнообразных представлениях для пользователей. Потоки можно представить виртуальными каналами сбора и обработки данных, как в транзакционных системах, только необходимо предусмотреть мощные средства динамической перестройки топологии и функциональности каналов. Для этого применяется событийная модель, согласно которой система снабжается компонентами мониторинга состояния и реагирования на события. Они обеспечивают ведение единого всеобъемлющего журнала событий, отражающих изменения состояния системы, и оперативное оповещение компонентов обработки событий о фактах их возникновения [90, 91]. Предметом мониторинга являются не только узлы объекта управления, но и компоненты самой системы, в отношении которых проверяется работоспособность, время отклика и другие показатели качества. Богатый набор правил оповещения и средств их обработки позволяет придать системе высокий уровень автономности – способности автоматически оптимизировать свою конфигурацию при изменениях во внешнем окружении, например переходить на резервную аппаратную единицу в случае отказа основной. Для реализации подобных функций можно применять технологии автономных вычислений [228].

Компоненты и потоки данных образуют следующую «порождающую» типовую крупноблочную функциональную архитектуру большой информационно-управляющей системы [6].



Проектирование средств автоматизации конкретного процесса состоит в его отображении на типовую архитектуру, так что она служит своего рода «системой координат» для проектировщиков. В результате отображения реализация задач, составляющих процесс, рассредоточивается по компонентам, выполненным в разных стилях. Например, автоматизация технического обслуживания и ремонта оборудования требует и поддержки взаимодействия с исполнителем работ (обмена юридическими и техническими документами), и расчета оптимального

графика ремонтов, и расширения информационной модели объекта. Ни один стиль не может быть выбран в качестве доминирующего, поэтому проектное решение рассеивается по равноправным разнородным составляющим. В больших системах составляющие обычно многочисленны и сложно связаны друг с другом, поэтому для обеспечения полноты автоматизированной поддержки процесса необходимо тщательно трассировать требования – явно прослеживать их воплощение в архитектурных единицах. Трассирование является важнейшим способом обеспечения результативности процесса реализации задач, поскольку оно устанавливает сквозную двунаправленную взаимосвязь между разнородными артефактами жизненного цикла системы – характеристиками качества, спецификациями, моделями архитектуры, программным кодом, массивами данных, тестами, электронными документами, экранными формами [205]. Эта взаимосвязь в общем случае носит характер отношения «многие-ко-многим»: не только реализация одной задачи входит в несколько артефактов, но и один артефакт реализует много задач. Поэтому в качестве источников и назначений трасс лучше всего использовать не цельные артефакты, а их фрагменты, каждый из которых реализует одну задачу. Разбиение артефактов на такие фрагменты способно радикально улучшить показатели сопровождаемости системы, поскольку именно на фрагменты направлены типовые воздействия, выполняемые при сопровождении (согласно стандарту ISO/IEC 9126, к ним относятся анализ функционирования системы, модификация функций, локализация изменений, тестирование [32]). Если удается выделить фрагменты компонента в самостоятельные единицы сборки системы, то возможно эффективно проводить ассимиляцию компонента, размещая каждый фрагмент по месту потребности в нем. В [203] изображен следующий показательный пример схемы трассировочных связей между фрагментами разнородных артефактов.



Однако на практике выявление фрагментов, каждый из которых относится к одной задаче, и сохранение разбиения на них при модификации системы оказывается очень трудоемким. Разноречивость целей приводит к тому, что задачи перемешиваются (tangle) друг с другом – не всегда можно строго провести границы между ними. Хуже того, реализация задачи может рассеиваться (scatter) по системе, пересекая (crosscut) границы единиц модульной архитектуры (процедур, классов, таблиц базы данных) и глубоко «погружаясь» в контекст своего исполнения. Качество автоматизации такой задачи зависит не только от качества средства ее реализации, но и от полноты его проникновения во все части системы. Такие рассеянные задачи порождаются как функциональными возможностями (например, ведение информационной модели объекта, верификация данных), так и нефункциональными требованиями (ведение журналов, защита информации и т.д.) [235]. Границу такой задачи невозможно записать при помощи традиционных нотаций проектирования и инструментов (CASE-средств), поскольку они трактуют каждую модульную единицу как элементарную (неделимую). Кроме того, CASE-средства, специализированные для частных технологий проектирования, плохо транслируют трассировочную информацию на артефакты других технологий, поэтому не удается автоматически прослеживать трассы. В результате обеспечение полноты трассирования с достаточно мелкой гранулярностью требует значительных затрат ручного труда [205].

Для снижения затрат мы предлагаем трассировать только наиболее важные требования к реализации задач, причем на достаточно грубом уровне. В этом мы следуем подходу, известному как трассируемость, основанная на значимости (value-based requirements traceability) [171]. Поскольку сопровождаемость большой системы критическим образом зависит от ассимилиру-

емости компонентов, мы предлагаем трассировать в первую очередь интеграционные интерфейсы артефактов, описывающие возможности встраивания в систему [76]. Такое трассирование порождает разметку интерфейсов классами задач, которую целесообразно оформлять как дополнительную структуру артефактов (такое оформление трассировочной информации принято в большинстве современных автоматизированных инструментов поддержки трассирования [138]). Например, журнал событий системы оснащается разметкой классами задач, выполнение которых приводит к их возникновению [6]. В терминах разд.1.1 можно сказать, что крупно гранулированные телеологические связи задач погружаются в дескриптивные связи артефактов, путем повышения уровня структурной сложности последних. Технологии проектирования должны расширяться механизмами преобразования дополнительной структуры в ходе разработки артефактов компонентов и комплексирования в системы (и одновременно сужаться путем исключения действий, разрушающих дополнительную структуру). Эти механизмы должны поддерживать рассеяние задач и работать однотипно в разнообразных технологиях.

Выработка таких механизмов входит в число целей аспектно-ориентированного программирования (АОП) [208] – парадигмы, предложенной в конце 1990-х годов для поддержки реализации рассеянных задач. Изначальная мотивация его создателей состояла в отсутствии в традиционных языках программирования конструкций, позволяющих выполнять разделение ответственности (separation of concerns) – разделять перемешанный исходный код по классам задач. Технологии аспектно-ориентированной разработки программ (aspect-oriented software development, AOSD) отличаются способами разделения, но сходятся в стремлении обеспечить сквозную трассируемость. Они имеют вид расширений традиционных технологий (в первую очередь, объектно-ориентированных): сначала «локальные» (нерассеянные) задачи реализуются обычными модулями, а затем в них вставляются аспекты – самостоятельные артефакты, реализующие рассеянные задачи. Вставка аспектов выходит за рамки традиционной компоновки модулей (linking), не позволяющей модифицировать поток исполнения программы, поэтому она называется связыванием (weaving). Связывание может выполняться как на этапе компиляции, путем генерации перемешанного кода, так и в процессе исполнения, путем вызова предварительно скомпилированных аспектов.

В качестве примеров эффективного применения АОП в публикациях обычно рассматриваются программно-технические задачи: журналирование, кэширование, защита информации и т.п. [144]. Обращения к ним перемешиваются с исполнением основных функций в любой системе. Их источником обычно служат такие нефункциональные требования качества, удовлетворить которые не удается без принятия специальных мер почти на каждом шаге исполнения программы: анализируемость, производительность и т.п. Их пользователями являются специалисты по техническому обслуживанию программного изделия, зачастую рассматриваемые как

менее значимые по сравнению с исполнителями основных автоматизированных процессов. Распространена практика обращаться к этим задачам по окончании реализации основных функций, поэтому возможность реализовать их отдельными артефактами и затем автоматически вставить в готовые функциональные модули приводит к значительному уменьшению затрат (по сравнению с ручной модификацией модулей). Вставлять аспекты действительно можно автоматически, если удается задать места обращения к ним на синтаксическом уровне – как точки выполнения операторов определенного вида (присваивание, вызов метода и т.п.). Приведем наглядный пример на языке AspectJ (это аспектно-ориентированное расширение языка Java [166]) – распечатка всех вызовов методов, наименования которых начинаются со слова `init` (инициализировать). Ручная реализация такой задачи включает поиск этого слова в текстах всех программ системы, умозрительное отделение вызовов методов (от наименований переменных, комментариев и др.) и вписывание обращения к функции печати после них. Компилятор AspectJ выполняет всю эту работу автоматически, получив на вход аспект следующего вида:

```
public aspect methodCallLogging {

    // Регулярное выражение, задающее точки вставки
    //     кода аспекта в исходную программу

    pointcut methodCalled(): execution(public * init*(..));

    // Действие, вставляемое после каждой точки

    after(): methodCalled() {
        System.out.println("Method call: " +
                           thisJoinPoint.getSignature());
    }
}
```

Примеры такого рода свидетельствуют, что рассеяние присуще задачам обеспечивающих процессов, включенных в контур автоматизации. Если ассортимент этих процессов исчерпывается техническим обслуживанием системы, так что их связь с основными можно выразить в программно-технических терминах, то эффект применения классических технологий АОП очевиден. В связи с этим даже предлагалось формально сопоставлять аспекты всем нефункциональным требованиям качества программных изделий [263]. Однако в больших информационно-управляющих системах автоматизируемые обеспечивающие процессы выходят далеко за рамки программно-технических: они обладают значительным разнообразием, оказывают заметное влияние на показатели результативности основных процессов почти на каждом шаге и в то же время плохо совмещаются с ними на понятийном уровне. Потребность в технологии типа АОП обостряется, поскольку привязка средств их автоматизации к основным в условиях применения традиционных «модульных» подходов сопряжена со значительными затратами ручно-

го труда [24]. В качестве примера рассмотрим задачу ведения информационной модели (паспорта) объекта управления. Информация, подлежащая занесению в паспорт, формируется по ходу многих процессов, например:

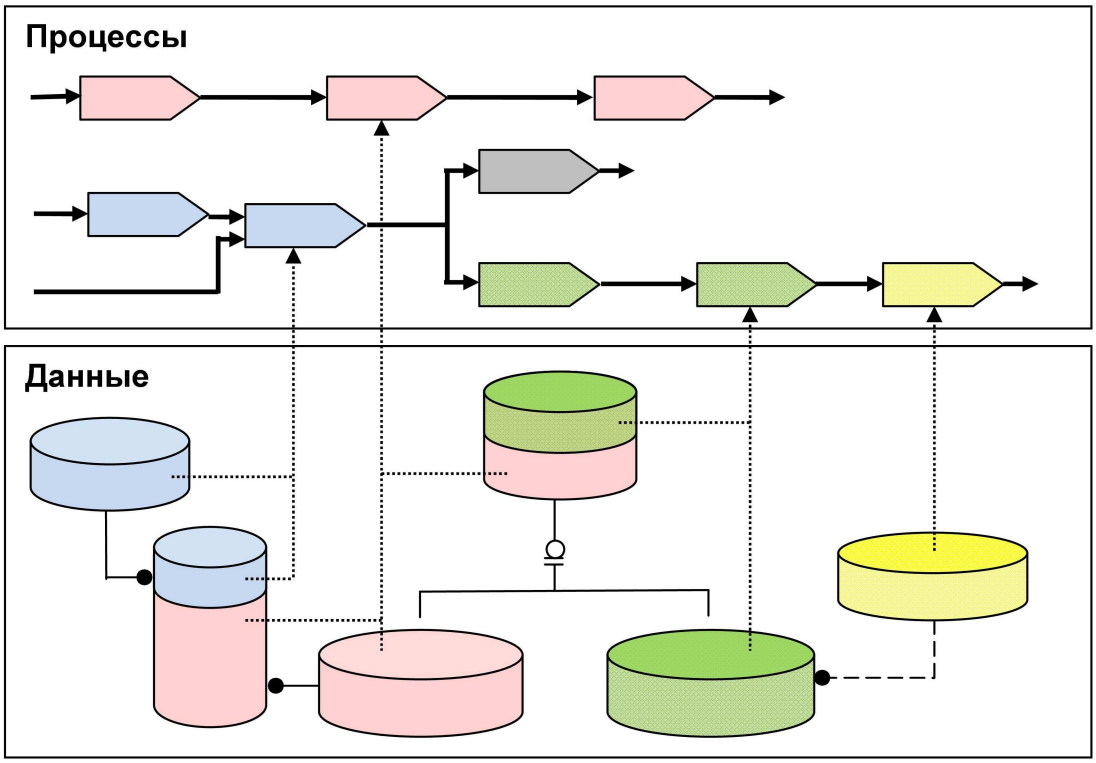
- сведения о собственности задаются в рамках процедур корпоративного управления;
- технические характеристики оборудования устанавливаются в процессах пусконаладки и технического обслуживания;
- разнообразные коды назначаются в ходе процессов кодирования и инвентаризации;
- энергетические показатели выявляются по результатам энергетических обследований;
- и т.д.

В традиционных средствах ведения информационных моделей каждому элементу сопоставляется монолитный набор атрибутов («карточка»), отражающий его участие во всех процессах без разделения по ним. Именно такое проектное решение диктуется «модульными» подходами к проектированию информационных систем, в том числе объектно-ориентированным подходом. Информацию об элементе можно ввести/получить только путем полного заполнения/чтения всей карточки, причем часто эти действия атомарны – не разделимы между многими пользователями. Но на достаточно крупном объекте нет и не может быть одного пользователя, владеющего информацией из всех процессов, поэтому приходится формировать карточки вне системы (например, рассылая по электронной почте файлы формата Excel) и далее передавать их оператору системы для однократного ввода. Значения многих атрибутов выяснить таким путем, в отрыве от контекста процесса-источника, не удается, и они заполняются «мусором». В итоге степень достоверности паспорта в целом неуклонно снижается, в конечном счете делая его непригодным к использованию. Нет возможности управлять качеством паспорта, заполняя карточки частично – в объеме, достаточном для поддержки выполнения заданного подмножества процессов.

Таким образом, обеспечение актуальности и достоверности паспорта требует разметки его атрибутов процессами-источниками значений, а задача ведения паспорта должна рассеиваться по артефактам, реализующим соответствующие шаги процессов. Самая мощная система ведения паспорта, оснащенная высокоинтеллектуальными алгоритмами анализа и корректировки данных, не принесет никакой пользы, если информация не будет поступать непрерывно по мере изменения объекта. Возникает естественное поле применения аспектно-ориентированного подхода, только не на этапе программирования, а раньше – при проектировании задачи ведения паспорта. Оформить в виде аспекта нужно модель этой задачи в подходящей нотации и автома-

тически вставить ее в модели процессов. Инструменты для такого аспектно-ориентированного моделирования существуют (например, инструмент WEAVR, позволяющий изобразить аспекты в виде действий и связать их с диаграммами состояний языка UML [264]), но в целом такое моделирование развито гораздо слабее, чем АОП. Дело в том, что современные аспектно-ориентированные технологии не имеют единой семантической базы, которая позволила бы обеспечить их совместимость друг с другом, прямой переход к реализации средствами АОП, формальную проверку корректности [238]. Как отмечается в [247], эти технологии «имеют разное происхождение и преследуют различные цели при поддержании характерных возможностей аспектно-ориентированного подхода». Поэтому для получения значимого эффекта от применения АОП в ходе проектирования необходимо создать его универсальную семантическую модель.

В настоящей работе (гл.4) предложена модель, основанная на описанном выше подходе к трассированию путем разметки интеграционных интерфейсов артефактов классами задач. Здесь аспект представляет собой не что иное, как средство реализации одного класса задач – артефакт, интерфейс которого размечен единственной меткой. Связывание описывается как присоединение аспектов, выполняемое по одной и той же специальной структурной схеме на уровне «модульных» основ и разметок связываемых артефактов. Правила построения этой схемы не зависят от выбора технологии проектирования. В большой информационно-управляющей системе целесообразно реализовать такую схему на базе событийной модели, оформляя каждый аспект как обработчик некоторой совокупности событий, вызываемый динамически по факту регистрации событий [6]. Таким образом, аспектно-ориентированный подход позволяет обеспечить достижение одного из ключевых функциональных показателей качества системы – способности динамически подстраивать ход исполнения процессов [193]. Как показано в разд.5.3, для задачи ведения паспорта объекта таким путем удается спроектировать рассеяние с сохранением разметки элементов задачами-источниками. Получается совместная модель данных и процессов, размеченных в результате трассирования, которая изображена «с высоты птичьего полета» на следующей концептуальной схеме. Здесь разметка обозначена цветовой раскраской: различным задачам соответствуют разные цвета, которыми окрашены как шаги процессов выполнения задач, так и данные, модифицируемые в ходе выполнения задач. Каждый элемент данных снабжается ссылкой на шаг процесса, в ходе которого он был изменен (поэтому в одном массиве данных могут встречаться элементы, размеченные разными задачами).



К числу рассеянных функциональных задач, возникающих в обеспечивающих процессах, относятся также оповещение участников процессов о ходе их выполнения, оперативная оценка эффективности процессов, проверка правильности действий пользователей и компонентов системы, перевод информации на разные языки и в разные форматы, и т.д. Но аспектно-ориентированный подход способствует снижению затрат на ассимиляцию компонентов не только в случае, когда требуется рассеивать их по системе. Дело в том, что аспектное связывание позволяет модифицировать модуль путем вставки аспектов внешним образом, в обход штатного интерфейса управления его структурой и поведением, на этапе сборки системы и даже «на лету» во время исполнения. Так можно реализовать модули, предназначенные для многократного использования в конфигурациях, различающихся наличием/отсутствием нескольких изменчивых задач, оформляемых как аспекты [226]. Еще одной областью такого применения связывания является интеграция унаследованных модулей (legacy), не доступных в исходных кодах и потому не допускающих модификацию путем редактирования [137].

В заключение раздела отметим, что в традиционной инженерии материальных систем раздельное решение рассеянных задач (separation of concerns) является давно устоявшейся практикой. Рассмотрим в качестве примера проектирование зданий – область, из которой системные инженеры заимствовали много идей. Модульную архитектуру здания составляют подъезды, этажи, комнаты, а аспектную – интенсивно пересекающие их системы освещения, водоснабжения, отопления и др. [158] Проект каждого из таких аспектов документируется в форме отдельного плана и отдается на реализацию отдельной бригаде специалистов. Связывание аспектов по

ходу строительства, включая разрешение всевозможных технологических и процедурных конфликтов, входит в число рутинных задач прораба.

1.4. Организация жизненного цикла больших информационно-управляющих систем

Подход к проектированию во многом предопределяет характер всего жизненного цикла любой системы. Жизненным циклом (life cycle) называется развитие системы начиная с разработки концептуального решения о ее создании и заканчивая прекращением ее применения. Базовые концепции, связанные с жизненным циклом, зафиксированы в международном стандарте ISO/IEC 12207-2008 «Systems and software engineering – Software life cycle processes» (в России принят под названием ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств» [33]). В этом стандарте жизненный цикл представлен в виде совокупности взаимосвязанных процессов, участниками которых являются заказчики системы, поставщики, разработчики, операторы, сопровожденцы, менеджеры и пользователи [33, разд.1.2]. Для каждого процесса указывается его наименование, цель, выходы, действия и задачи (как ни странно, входы процессов не указаны). Применение стандарта для организации жизненного цикла частной системы, как правило, требует его адаптации, которая состоит в исключении ненужных стандартных процессов, результатов, задач, и в уточнении особенностей выполнения оставшихся процессов [33, прил.А]. Процессы разделены на следующие группы:

- процессы соглашения (agreement processes);
- процессы организационного обеспечения проекта (organizational project-enabling processes);
- процессы проекта (project processes);
- технические процессы (technical processes);
- процессы реализации программных средств (software implementation processes);
- процессы поддержки программных средств (software support processes);
- процессы повторного применения программных средств (software reuse processes).

При наличии модели качества, указанные в ней диапазоны допустимых значений показателей естественным образом применяются в процессах жизненного цикла в качестве ограничений. Заказчикам они позволяют точнее сформулировать свои ожидания от системы, поставщикам – обосновать пригодность своей продукции, разработчикам и специалистам по сопровождению – принять правильные проектные решения по созданию и развитию системы, операторам и пользователям – сформировать корректную модель поведения при работе с системой, ме-

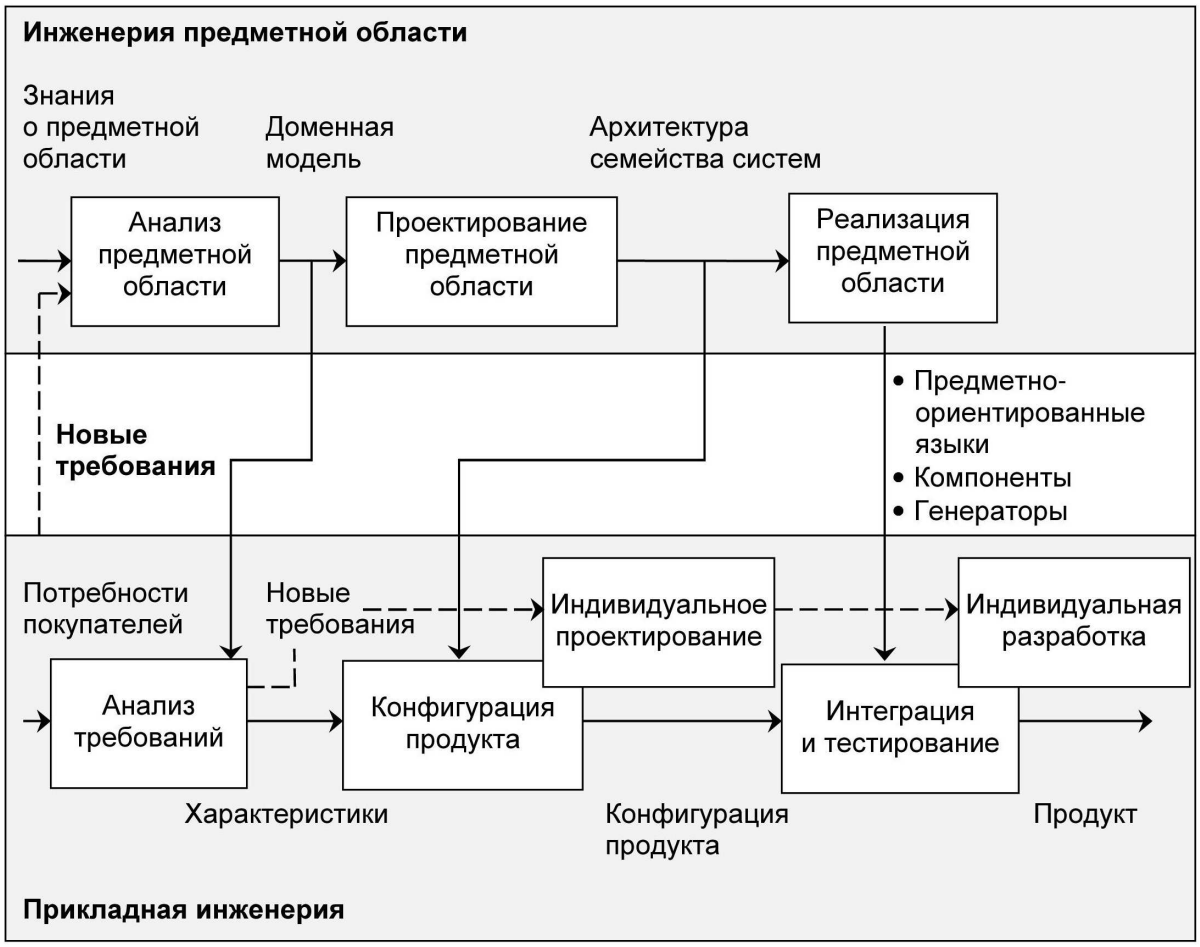
неджерам – найти основания для выполнения разнообразных управляющих воздействий. Критерии эффективности управления процессами жизненного цикла в ограничениях качества, как правило, имеют экономический характер – они состоят в минимизации затрат финансовых средств, труда и времени. По такому принципу организуются тендеры и аукционы, направленные на отбор наиболее компетентных исполнителей тех или иных процессов. (К сожалению, на практике этот принцип часто доводится до абсурда, но это – тема для отдельного рассмотрения вне рамок настоящей работы.)

В целях снижения затрат в инженерии информационных систем, как и в других областях техники, были выработаны разнообразные технологии – шаблоны (модели) выполнения процессов жизненного цикла, наиболее рационального в определенных условиях, таких как заданная область применения изделия, значения масштабных факторов, парадигма программирования и др. Технологии регламентируют состав и порядок работ процессов, вид и формат результатов, структуру коммуникаций между исполнителями, применение CASE-средств. Многие технологии способствуют выполнению задач, регламентированных стандартом ISO/IEC 12207-2008. Хорошо проработанные и многократно апробированные технологии приобретают статус продукции промышленного масштаба, потребителями которой являются исполнители процессов жизненного цикла. Например, широко применяются технология объектно-ориентированной разработки Rational Unified Process (RUP) [134] и технология быстрого создания малых изделий eXtreme Programming (XP) [13].

Однако, как указывалось в разд.1.1, в жизненном цикле большой системы непосредственно использовать традиционные технологии не удается, поскольку они предъявляют высокие требования к ясности и определенности постановки входных задач. В условиях, когда невозможно подать технологиям на вход исчерпывающий перечень автоматизируемых функций, применяется предметный подход: ставится задача реализовать в системе сущности и взаимосвязи, наиболее характерные и наиболее существенные для автоматизируемой предметной области [55]. К ним привязываются «инвариантные» высокоуровневые пользовательские цели (goals), способы достижения которых подбираются и оптимизируются по мере функционирования и развития системы [237]. Чтобы выявить и разделить сущности на небольшие технологические «кусты», для реализации которых пригодны традиционные методы, необходимо провести полномасштабную системно-инженерную проработку предметной области. Ее результатами являются модели и средства, предназначенные для многократного применения в «прикладных» процессах создания и модификации отдельных компонентов (процессы реализации программных средств) и системы в целом (технические процессы) [76]. Согласно стандарту ISO/IEC 12207-2008 такая проработка проводится в рамках процесса инженерии предметной области (domain engineering, в ГОСТ Р ИСО/МЭК 12207-2010 – «проектирование доменов»).

Изначально он был разработан для создания линеек продуктов, поэтому включен в группу процессов повторного применения программных средств. Выдвижение его во главу угла лежит в основе адаптации стандарта для организации жизненного цикла большой системы.

Процесс инженерии предметной области изображен на следующей схеме [130].



Согласно этой схеме, процесс уровня предметной области состоит из этапов, присутствующих почти во всех технологиях создания программных изделий: анализ, проектирование, реализация. Специфика состоит в том, что их результаты часто находятся на метауровне по отношению к традиционным технологиям: это разнообразные «порождающие» модели, языки описания частей предметной области (domain-specific languages, DSL), генераторы программ. На этапе анализа сначала проводится выявление (identification) предметной области: указываются ее границы, выявляется круг представляющих ее лиц и их целей, выделяются смежные предметные области, устанавливается набор первоочередных источников предметных знаний. По-видимому, здесь самой трудной задачей, критическим образом влияющей на технико-экономические показатели всего жизненного цикла, является ответ на вопрос, что *не нужно* включать в рассмотрение при создании задуманной системы. По результатам выявления пред-

метной области проводится ее моделирование, включающее сбор, систематизацию и формализацию информации о ней, поступающей из разных источников. Здесь существует ряд технологий, нацеленных на построение доменных моделей (domain models) различных видов. Так, FODA (Feature-Oriented Domain Analysis) и родственные методы позволяют представить предметную область в виде диаграммы характеристик (feature model) – элементарных свойств, допустимые сочетания которых образуют конфигурации средств автоматизации [130]. Метод RSEB (Reuse-driven Software Engineering Business) [201] задает правила объектно-ориентированной декомпозиции предметной области с фиксацией результатов на языке UML. В контексте подхода, изложенного в разд.1.2, особый интерес представляют методы, в основе которых лежит онтология предметной области, например ODE (Ontology Domain Engineering) [173]. Онтология служит источником терминов и условий корректности для модели качества большой системы и детализирующих понятийных моделей (процессов, организационной структуры и т.д.).

В совокупности все артефакты анализа предметной области формируют пространство задач (problem space) системы. Оно служит источником входных данных для этапов проектирования и реализации, формирующих пространство решений (solution space). В результате проектирования, как мы видели в разд.1.3, формируется архитектура системы в целом и ее функциональных подсистем, выполненных в разных стилях. В качестве основных входов работ по построению архитектуры подсистем выступают части пространства задач. Соответствие частей подсистемам приведено в следующей таблице (Таблица 4) вместе с примерами задач, относящихся к процессу учета энергоресурсов.

Таблица 4

Часть пространства задач	**Функциональная подсистема**	**Стиль**	**Пример задачи**
Онтология	Информационная модель объекта управления (ИМОУ)	Информационный	Хранение реестра измерительных каналов
Модель процессов	Подсистема мониторинга состояния (ПМС)	Потоковый	Управление режимом опроса измерительных приборов
Организационная структура	Модули защиты информации (МЗИ)	Взаимодействия	Разграничение доступа к результатам измерений по месту пользователя в оргструктуре
Каталог форм документов	Подсистема обмена документами (ПОД)	Взаимодействия	Генерация справки о расходе энергоресурсов за месяц
Эскизы пользовательского интерфейса	Подсистема пользовательских интерфейсов (ППИ)	Взаимодействия	Визуализация графиков энергопотребления
Протоколы взаимодействия с устройствами	Подсистема сбора информации и телеуправления (ПСИ)	Взаимодействия	Сбор показаний с измерительных приборов
Каталог расчетных алгоритмов	Подсистема расчета и анализа данных (ПРАД)	Алгоритмический	Расчет суммарного объема потребления энергоресурсов по объекту

В соответствие с моделями архитектуры реализуются средства, многократно используемые в непрерывных процессах уровня прикладной инженерии, по мере развития большой системы. Они бывают как предметно-ориентированными, автоматизирующими частные задачи (например компонент расчета потерь электроэнергии в линиях электропередачи), так и программно-инфраструктурными, реализующими общие концепции автоматизированного управления (например компонент ведения журнала событий). Большой вклад в уменьшение трудозатрат прикладного уровня вносят «родовые» компоненты – базы для создания предметно-ориентированных компонентов решения определенных классов задач путем метапрограммирования (например абстрактный драйвер опроса произвольного измерительного прибора, способный интерпретировать предметно-ориентированный язык описания протоколов опроса). От компонентов требуется максимальная сочетаемость и минимальная избыточность. Построение любой конфигурации системы сводится к конструированию моделей всех охватываемых ею процессов и объектов, комплексированию и адаптации набора компонентов, реализующих эти модели.

Таким образом, критическим фактором снижения затрат на технические процессы жизненного цикла является тщательная проработка моделей, создаваемых в процессе инженерии предметной области. В то же время по мере роста масштаба системы возрастает и трудоемкость процедур их создания, модификации, проверки, трансформации в программный код, поэтому здесь требуется автоматизация. Как мы видели в разд.1.2 на примере модели качества, для этого необходимо представление моделей в формальных языках, допускающее машинную обработку. В арсенале инженерии информационных систем накоплено большое количество формальных языков представления разных точек зрения на предмет моделирования и методов составления и применения формальных моделей.

Многие из этих методов разрабатывались с прицелом на возможность математически строго доказывать соответствие программ требованиям. Некоторые из них предписывают моделировать спецификации наборами предложений того или иного формального исчисления, а реализации – алгебраическими системами, в которых эти предложения выполняются [50]. Здесь активно применяются разнообразные конструктивные варианты традиционной логики [22]. Для отражения динамики функционирования программ в дополнение к ним применяются модальные и динамические логики [29], денотационные модели [251], алгебры процессов [127], сети Петри [135], эволюционирующие алгебры [187] и др. Для проектирования информационных моделей и онтологий, наоборот, достаточно аппарата теории множеств. Основными операциями здесь являются декартово произведение и переход к подмножествам (в этом можно убедиться на примере конструкции реестра измерительных каналов из разд.1.3), поэтому достаточно мощные формальные методы информационного моделирования получаются на базе теории ро-

дов структур Н. Бурбаки [102]. К формализованным можно отнести также многочисленные языки и методы диаграммного моделирования, такие как язык объектно-ориентированного проектирования UML и разнообразные методы моделирования процессов. Они основаны на теории графов и вследствие этого обладают сравнительно невысокой доказательной силой, но зато более просты в изучении.

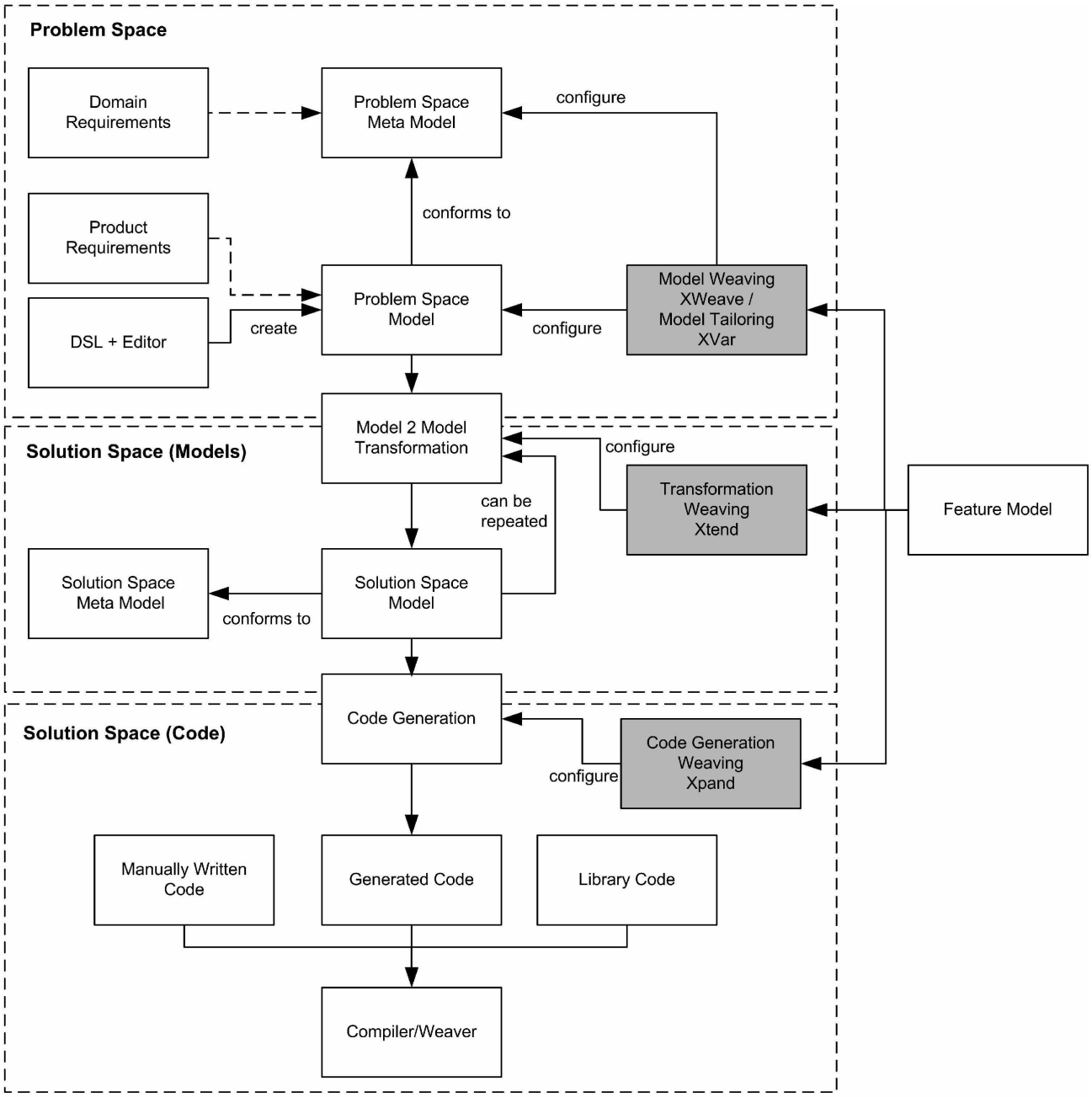
Все упомянутые методы глубоко проработаны с теоретической точки зрения, однако на практике они часто оказываются неудобными в применении по следующим причинам [78]:

- формальные конструкции (как текстовые, так и графические) не всегда адекватно воспроизводят интуитивные образы моделируемых понятий;
- конструкции разных языков плохо переводятся друг в друга при попытке построить единую многоплановую модель;
- многие формальные языки не поддерживают разбиение моделей на небольшие фрагменты, а процесса их модификации – на небольшие быстрые шаги;
- высокий уровень абстрактности не позволяет учесть нефункциональные требования качества.

Для преодоления этих затруднений в начале 2000-х годов был предложен новый технологический подход к моделированию в жизненном цикле систем – разработка, управляемая моделями (model-driven engineering, MDE) [248]. Этот подход предписывает создавать удобные предметно-ориентированные языки моделирования (domain-specific languages, DSL), специализируя языки общего назначения на уровне метамоделей. Предоставляется инструментарий для создания средств автоматического анализа, синтеза и трансформации (transformation) моделей. Большое внимание уделяется генерации программного кода из модели, причем предусматривается адаптация к особенностям разнообразных сред исполнения приложений (платформ) в целях повышения значений «машинно-зависимых» нефункциональных показателей. Предусматривается применение моделей и во время исполнения кода, как для верификации, так и для адаптации к изменяющимся условиям внешней среды [259] (к моделям такого рода относится пример декларации на языке xNoFun из разд.1.2). Все эти возможности стимулируют применение технологий разработки, управляемой моделями, в инженерии предметной области. Здесь фактически формируются полноценные предметно-ориентированные технологии автоматизированного проектирования, призванные радикально снизить затраты на преодоление концептуального разрыва между пространством задач и пространством решений [168].

Автоматизация разработки, управляемой моделями, открывает широкие перспективы для поддержки трассирования [138]. Действительно, инструмент автоматической генерации модели (в том числе программного кода) способен вести подробный журнал переработки моделей, поданных ему на вход, в котором можно найти мелко гранулированную трассировочную

информацию. Ее семантику можно задать в форме метамодели трассирования, позволяющей строить, прослеживать и анализировать типизированные трассы (typed trace links) [232]. В этих условиях целесообразно привлекать аспектно-ориентированный подход, интегрируя рассеянные и изменчивые задачи в модели путем связывания. Такой технологический процесс инженерии предметной области, предложенный в [184], приведен на следующей схеме.



Несмотря на достоинства, технологии разработки, управляемой моделями, в настоящее время редко применяются в жизненном цикле больших систем [225]. Причиной этого служит низкая масштабируемость формальных методов вообще и средств автоматизированного моделирования в частности [211]. Большинство формальных методов и средств разработано в пред-

положении, что модель всегда используется и модифицируется как единое целое. Поэтому ведение моделей, состоящих из десятков тысяч элементов, оказывается чрезвычайно трудоемким: слабо проработаны механизмы разделения моделей на части, с которыми можно работать по отдельности [194]. Кроме того, предполагается, что предметно-ориентированные метамодели изменяются относительно медленно, поэтому процедуры их модификации громоздки и длительны [168]. Фактически, в условиях разработки, управляемой моделями, центр затрат труда, требуемого для проектирования и реализации системы, переносится на разработку предметно-ориентированных средств моделирования [189]. При этом требуется «научить» машину принимать множество решений: от гармоничного размещения изображений элементов модели на экране до «переноса центра тяжести» с предметной области на особенности архитектуры компьютеров по ходу трансформации, пример которого был приведен в разд.1.3 при переходе от онтологии к схеме данных. Это относительно новая область программирования, где пока не выработаны ни типовые приемы, ни эффективные методики обучения специалистов. Поэтому предметно-ориентированные модели, созданные с использованием средств от разных поставщиков, плохо интегрируются друг с другом [194]. Для моделей до сих пор не сформировалось общепринятое понятие интеграционного интерфейса, способное лечь в основу эффективных приемов комплексирования в рамках MDE. Напрашивается вывод, что разработка, управляемая моделями, сама нуждается в метамодели – единой семантической базе, позволяющей выявить, изучить и масштабировать общую структуру механизмов создания и использования моделей, независимо от их смысла, нотации и размера.

Математический аппарат, пригодный для построения такой семантической базы, должен быть достаточно абстрактным, чтобы можно было кратко описать главные понятия и доказать их основные свойства, не «потонув» в деталях описания состава и структуры частных моделей. В качестве такого аппарата целесообразно привлечь теорию категорий [98]. Математики разрабатывают и применяют ее начиная с 1940-х годов для решения сходных задач – выявления структурного единства внешне различных математических конструкций без потери строгости, и проведения однократных доказательств утверждений о них [93]. Подобный подход был разработан и в математической теории систем, где рассуждения «поднимаются» на уровень теории множеств и даже логики, а затем их результаты интерпретируются в теориях частных классов систем [96]. Категория – это коллекция абстрактных объектов, связанных морфизмами (стрелками), обозначающими допустимые преобразования одних объектов в другие. Ее точное определение занимает буквально несколько строк: категория C состоит из совокупности объектов $\mathrm{Ob}\ C$ и совокупности морфизмов $\mathrm{Mor}\ C$, на которых заданы следующие операции. Во-первых, каждому морфизму f сопоставляется два объекта: область $\mathrm{dom}\ f$ и кообласть $\mathrm{codom}\ f$ (соотношения вида $\mathrm{dom}\ f = A$ и $\mathrm{codom}\ f = B$ наглядно записываются в форме стрелки $f : A \to B$). Во-

вторых, для любой пары морфизмов f, g, удовлетворяющей условию $\mathrm{codom}\, f = \mathrm{dom}\, g$, определена композиция («цепочка») – морфизм $g \circ f : \mathrm{dom}\, f \to \mathrm{codom}\, g$, причем она ассоциативна: для любой тройки морфизмов f, g, h, удовлетворяющей условиям $\mathrm{codom}\, f = \mathrm{dom}\, g$ и $\mathrm{codom}\, g = \mathrm{dom}\, h$, выполняется соотношение $h \circ (g \circ f) = (h \circ g) \circ f$. В-третьих, любой объект A обладает тождественным морфизмом $1_A : A \to A$ («ничегонеделание») таким, что для любого морфизма $f : A \to B$ выполняется соотношение $f \circ 1_A = 1_B \circ f = f$.

Широко распространенным способом задания объекта в категории является требование универсальности – существование и единственность связи с объектами, аналогичными в некотором явно указанном смысле. Универсальные объекты задаются однозначно с точностью до изоморфизма – формального представления несущественного различия между объектами. Напомним, что морфизм $f : A \to B$ называется изоморфизмом, если существует (обязательно единственный) морфизм $f^{-1} : B \to A$ такой, что $f^{-1} \circ f = 1_A$ и $f \circ f^{-1} = 1_B$. Примером универсального объекта служит терминальный объект – такой, в который из любого другого объекта входит единственный морфизм. Покажем, что терминальный объект, если он существует, определяется однозначно с точностью до изоморфизма: единственным его морфизмом в себя является тождественный морфизм, так что если $\mathbf{1}$ и $\mathbf{1}'$ – терминальные объекты, то по определению существуют (единственный) морфизм $f : \mathbf{1} \to \mathbf{1}'$ и (единственный) морфизм $f' : \mathbf{1}' \to \mathbf{1}$, причем они обязаны удовлетворять соотношениям $f' \circ f = 1_{\mathbf{1}}$ и $f \circ f' = 1_{\mathbf{1}'}$.

Вместе с категорией вводится понятие функтора – отображения категорий, сохраняющего структуру преобразований. Функтор fun, действующий из категории C в D – это пара одноименных отображений $fun : \mathrm{Ob}\, C \to \mathrm{Ob}\, D$, $fun : \mathrm{Mor}\, C \to \mathrm{Mor}\, D$, удовлетворяющая следующим условиям (для произвольных морфизмов f, g и объекта A в категории C): (i) $fun(\mathrm{dom}\, f) = \mathrm{dom}\, fun(f)$, $fun(\mathrm{codom}\, f) = \mathrm{codom}\, fun(f)$; (ii) $fun(g \circ f) = fun(g) \circ fun(f)$, если композиция $g \circ f$ определена; (iii) $fun(1_A) = 1_{fun(A)}$. Обратим внимание, что совокупность всех категорий и всех функторов (со стандартным законом композиции отображений) удовлетворяет всем аксиомам категории.

Характерным примером категории служит **Set**, состоящая из всех множеств и всех их отображений (со стандартным законом композиции отображений). Изоморфизмами в **Set** служат в точности все биекции, поэтому рассмотрение множеств с точностью до изоморфизма приводит к классическому понятию кардинального числа. Терминальным объектом служит одноэлементное множество. Категорией также является совокупность всех алгебраических систем фиксированной сигнатуры и всех их гомоморфизмов. Переход от алгебраической системы к ее основному множеству представляет собой функтор из такой категории в **Set**, «забывающий» сигнатуру.

Поскольку множества традиционно используются в теоретическом программировании в качестве формальных моделей типов данных, были предложены категорные модели программ, в которых объекты отвечают типам, а морфизмы – вычислениям. Было показано, что определенные универсальные категорные конструкции (конечные произведения и экспоненты) адекватно описывают построение типов высших порядков, так что основным объектом исследования оказались категории, в которых они существуют [218]. Такие категории называются декартово замкнутыми и образуют модели лямбда-исчисления (а значит, и других известных концепций алгоритма ввиду тезиса Черча–Тьюринга). На них основана теория областей (domain theory) [116] – категорная формализация денотационной семантики алгоритмических языков программирования. Позднее она была распространена на объектно-ориентированные языки [262]. Отметим, что частным случаем конструкции экспоненты является объект-степень (категорный аналог множества всех подмножеств заданного множества) [30], поэтому подходящие подкатегории в декартово замкнутых категориях способны служить моделями теории родов структур.

Однако инженерия информационных систем нацелена, в первую очередь, не на разработку вычислительных и информационных моделей, а на повышение эффективности жизненного цикла систем. Поэтому ее математические методы требуют категорий другого рода, в которых объекты отвечают артефактам процессов жизненного цикла, обозначаемых морфизмами [174]. Такие категории позволяют придать строгий смысл разнообразным диаграммам процессов, состоящим из «квадратиков и стрелочек» (в том числе приведенным в настоящей работе). Разнообразные универсальные категорные конструкции позволяют формализовать выбор процессов, доставляющих рациональные решения заданных проблем (таких как синтез артефакта с заданными свойствами) [180]. Таким образом, теория категорий позволяет наиболее адекватно, полно и явно выразить основные положения системного анализа в приложении к проблемам управления жизненным циклом систем [181].

Категории такого рода, описывающие частные формальные методы, встречаются в литературе начиная с 1970-х годов. Наиболее просто они строятся для алгебраических методов и их расширений, путем заимствования «готовых» категорий из универсальной алгебры: морфизмами аксиоматических спецификаций служат сигнатурные морфизмы, а реализующих их алгебраических систем – как правило, гомоморфизмы. Соответствие реализаций спецификациям описывается метаязыковыми функторами, обеспечивающими согласованность отношения выводимости формул с их выполнимостью. Конструкция такого рода называется институцией (institution) [182]. На алгебраическом подходе также основаны категорные модели баз данных [110] и онтологий предметных областей [192]. Для синтеза сложных спецификаций и онтологий путем вычисления действий сигнатурных морфизмов было разработано CASE-средство SPECWARE [252]. Из алгебр процессов и родственных им структур были построены категории,

пригодные для формального моделирования распределенных систем, и были установлены естественные взаимоотношения между моделями разных видов (например, модели детерминированных систем образуют рефлективные подкатегории в категориях моделей произвольных систем) [245]. Позднее для моделирования динамики программ был привлечен аппарат коалгебр [200]. Существуют и категории, описывающие мультиагентные системы [233].

Список примеров можно продолжить, однако это не входит в рамки работы, поскольку примеры не способны составить базу для формулирования и доказательства утверждений, справедливых для *всех* технологий разработки информационных систем. Получается, что цель привлечения теории категорий достигнута не в полной мере: общеинженерные проблемы программирования, такие как масштабирование, трассирование, разделение ответственности, попрежнему имеют множество разнородных частных трактовок, специфичных для частных технологий. В направлении улучшения этой ситуации исследователями сделаны только первые шаги: например категорная формализация понятий метамодели и трансформации, ключевых для разработки, управляемой моделями, предложена в [146] и развита в [169]. Для формального описания произвольных технологий проектирования систем предложена многокомпонентная конструкция, состоящая из категорий и функторов (architecture school) [175].

Отталкиваясь от этих результатов, в [65] мы построили категорию, объектами в которой выступают конструкции такого рода, а морфизмы отвечают «метапроцессам» преобразования технологий, в том числе создания предметно-ориентированных языков и средств моделирования. Систематическому изложению этого подхода посвящена гл.2 настоящей работы. Опираясь на него, мы снабжаем единообразной теоретико-категорной семантикой, предусматривающей масштабирование, ключевые технологические приемы жизненного цикла больших информационно-управляющих систем: организацию распределенных вычислений (гл.3), трассирование и аспектно-ориентированный подход (гл.4), совместное моделирование данных и процессов (гл.5). Таким образом, сформирован фундаментальный задел для значительного снижения затрат на жизненный цикл больших систем путем сквозной автоматизации.

Этот задел применяется при выработке и реализации проектных решений, трудоемкость которых возрастает по мере роста масштаба системы, следующим образом. Строятся теоретико-категорные конструкции, формально описывающие решения на абстрактном концептуальном уровне. Путем вычислений в категориях оцениваются свойства этих решений, выбирается наиболее адекватное из альтернативных решений. Соответствующая ему абстрактная конструкция интерпретируется в понятиях подходящей технологии, и выбираются либо создаются инструменты для автоматизации работ по его реализации. Пример применения такого подхода, рассматриваемый на протяжении настоящей работы, изображен на следующей схеме.


Расчет режимных показателей большого энергетического объекта
Генерация программного кода расчета показателей по информационной модели (паспорту) объекта
Обеспечение корректности и эффективности сгенерированного кода
Поддержание актуальности паспорта
Вычислительное расширение алгебры расчетов (разд.3.5)
Категорная схема связывания аспекта ведения паспорта (разд.4.3)
Генератор программного кода расчета показателей в распределенной вычислительной среде (разд.5.5)

Глава 2. ТЕОРЕТИКО-КАТЕГОРНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ

2.1. Приемы комплексирования систем

Комплексированием называется сборка сложных систем из отдельных составных частей (компонентов). Акты комплексирования наглядно описываются диаграммами – схемами, состоящими из обозначений составных частей и действий по их соединению. В традиционной технике для этого применяются сборочные чертежи, смысл которых однозначно определяется физической природой систем. Однако для инженерии информационных систем характерно многообразие подходов к выделению частей (декомпозиции), отражающих разные точки зрения на автоматизируемую предметную область. Они определяют различную семантику единиц комплексирования и правил взаимодействия между ними в рамках систем. Назовем следующие классические подходы, образующие целостные парадигмы разработки программного обеспечения:

- структурный подход, предписывающий собирать программы из процедур, взаимодействующих посредством вызовов [40];
- объектно-ориентированный подход, в рамках которого в предметной области выделяются автономные иерархически организованные сущности (объекты), взаимодействующие путем обмена сообщениями [20];
- аспектно-ориентированный подход, когда система собирается из программных реализаций классов задач (concerns), самостоятельно определяющих условия и содержание взаимодействия со своим окружением [208].

Как указывалось в разд.1.3, при создании больших систем целесообразно использовать разные подходы для синтеза разных подсистем. Возникает потребность в методологии сборки систем из таких разнородных составляющих, пригодной к применению в том числе в рамках разработки, управляемой моделями (MDE) [203]. Здесь требуется описывать акты комплексирования и анализировать свойства их результатов в абстрактных терминах теории систем, не зависящих от конкретной парадигмы декомпозиции.

Единица декомпозиции программного приложения традиционно называется модулем. В классическом структурном программировании модуль представляет собой набор процедур, а его интеграция в систему сводится к их вызову из других модулей, т.е. к подстановке их программных реализаций вместо имен в ходе исполнения. Чтобы интегрировать модуль, нужно изменить систему (добавить вызов), оставляя его неизменным. В ходе развития программирования концепции процедуры и модуля претерпевали существенные изменения, однако вызов процедуры остается основным механизмом межмодульного взаимодействия. Современным

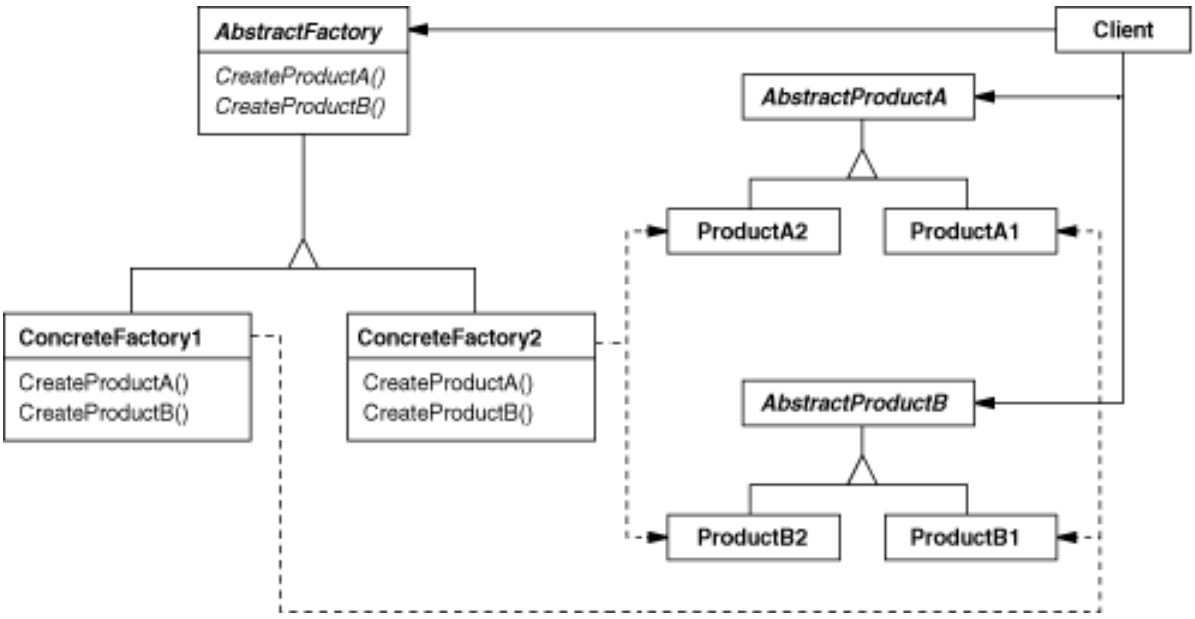
подходом к модуляризации процедур, в т.ч. удаленных от точек вызова, считается сервисно-ориентированная архитектура (service-oriented architecture, SOA).

Полноценные системы, в дополнение к программным модулям, содержат массивы данных, которые читаются, порождаются и модифицируются в ходе исполнения. Чтобы интегрировать в систему массив, его необходимо загрузить – преобразовать к виду, доступному для модулей. При этом модули менять не нужно. Например, для упрощения загрузки данных из реляционных баз в объектные приложения разрабатываются средства объектно-реляционного отображения (object-relational mapping, ORM).

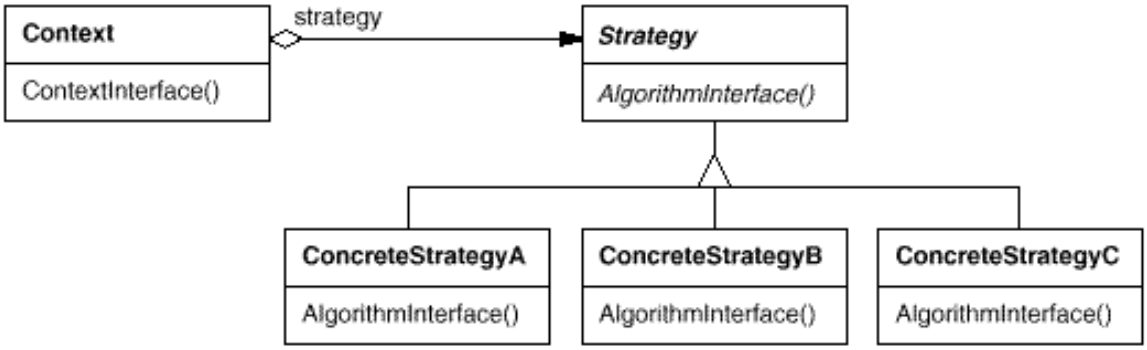
При создании крупных систем в качестве компонентов применяются не только модули, но и готовые автономные приложения. Они недоступны для изменения, поэтому для их интеграции требуется создавать дополнительные промежуточные компоненты, способные взаимодействовать с каждым из них, посылая им и принимая от них сообщения в подходящем формате. Такие компоненты выступают в роли «клея» (glue), соединяющего приложения в систему. Программные реализации «клея», совместимые с распространенными технологиями программирования, выпускаются в виде промежуточного программного обеспечения (middleware). Примером соединения также является связывание (weaving) – основной способ комплексирования в аспектно-ориентированном подходе, состоящий в присоединении аспектов к базовым программным модулям без изменения тех и других [234].

Таким образом, имеются три приема комплексирования: загрузка, подстановка и соединение [65]. Ясно, что любой акт сборки системы комбинируется из них, поскольку для включения части в целое можно либо изменить часть, либо изменить целое, либо добавить «клей». В структурном программировании комплексирование ограничивается тем обстоятельством, что каждый прием может быть применен только к единице соответствующего вида. Чтобы снять это ограничение, была предложена концепция объекта – автономной сущности с неизменным интерфейсом, содержащей процедуры (методы), доступные для внешнего вызова, и данные (атрибуты). Однако обнаружилось, что такая универсальность объектов порождает слишком большую степень произвола: архитекторам трудно предугадать, какой режим совместной работы объектов будет наиболее эффективным для выполнения той или иной конкретной задачи. Решение этой проблемы (заимствованное из практики проектирования зданий) заключается в выявлении шаблонов (patterns). Шаблон – это правило комплексирования объектов, обеспечивающее оптимальное решение явно описанного класса задач проектирования. Оказывается, шаблоны естественно классифицируются по приемам комплексирования – в основополагающей книге [27] все шаблоны разделены на три класса (причем без объяснения причины):

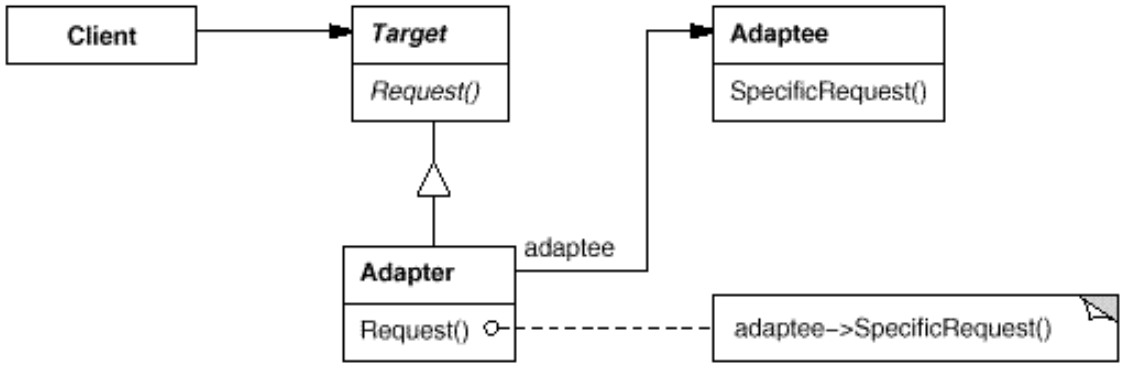
- создающие шаблоны, порождающие объекты из исходных данных, например *абстрактная фабрика* (AbstractFactory);



- поведенческие шаблоны, управляющие поведением объектов путем подстановки разных реализаций одних и тех же методов (перегрузки), например *стратегия* (Strategy);



- структурные шаблоны, «склеивающие» неизменяемые объекты, например *адаптер* (Adapter).



В краткой форме соображения, изложенные в настоящем разделе, собраны в следующей таблице (Таблица 5) [65].

Таблица 5

Прием комплексирования	**Способ**	**Схема**	**Механизм интеграции**	**Единица комплексирования**	**Пример технологии**	**Класс шаблонов ООП**
Загрузка	Изменение части	$P \to S$	Загрузка данных	Массив данных	ORM	Создающие шаблоны
Подстановка	Изменение целого	$P \to S' \leftarrow S$	Вызов процедуры	Модуль	SOA	Поведенческие шаблоны
Соединение	Добавление «клея»	$P \leftarrow G \to S$	Обмен сообщениями	Приложение	Middleware	Структурные шаблоны

2.2. Категории диаграмм и оптимизация архитектуры

Мнемоническим схемам приемов комплексирования, приведенным в третьей колонке таблицы выше (см. Таблица 5), можно придать точный смысл, если привлечь аппарат теории категорий – математической дисциплины, которая «начинается с наблюдения, что многие свойства математических систем можно представить просто и единообразно посредством диаграмм» [93, с.12]. Любому компоненту или системе сопоставляется объект подходящей категории (формальная модель), а любому действию по интеграции компонента в систему – морфизм, отображающий объект-область (компонент – вход действия) в объект-кообласть (систему – выход). Подчеркнем, что для компонента *P* и системы *S*, как правило, указывается не только возможность (или невозможность) интеграции *P* в *S*, но и совокупность всех способов интеграции (вставка, разрешение ссылок, перекомпоновка и т.д.), порождающая множество морфизмов Mor(*P*, *S*). В этом выразительные возможности теории категорий превосходят мереологию – дисциплину, в рамках которой формально анализируется отношение «часть-целое» [219]. Композиция морфизмов отвечает многошаговым действиям (процессам), причем их результат не зависит от порядка прослеживания шагов (свойство ассоциативности). Имеются тождественные морфизмы, означающие «ничегонеделание». Таким образом, получается категория, которую мы далее будем обозначать через *c-DESC*. Подчеркнем, что мы не накладываем на нее никаких априорных ограничений, поскольку не имеем оснований исключать, что любая категория описывает некоторую (может быть, еще не придуманную) технологию комплексирования систем.

Наборы компонентов, из которых строятся системы, и взаимосвязи между ними представляются диаграммами в этой категории (формальными мегамоделями). Акту сборки системы из набора Δ отвечает построение копредела – коконуса с основанием Δ, обладающего свойством универсальности (определения этих понятий, ввиду их важности, мы приводим ниже). Вершина копредела обозначает систему, а ребра копредельного коконуса – вхождения компонентов в нее. Примером из области классического программирования служит компоновка приложений из модулей (linking): с точки зрения компоновщика, модуль представляет собой мно-

жество имен внешних процедур, а его интеграция – привязку к вызывающим их модулям путем отождествления процедур с совпадающими именами. В простейшем случае, когда компоненты никак не взаимосвязаны (например при объединении модулей в библиотеку), система представляет собой их прямую сумму, явным образом сохраняющую идентичность каждого компонента и не добавляющую ничего лишнего. Даже пустая диаграмма может иметь копредел (его вершина называется инициальным объектом): его можно создать, вообще не имея компонентов. Примером служит инициализационный массив данных, загружаемый в любой модуль единственным образом.

Схемы приемов комплексирования представляют собой схемы диаграмм. Любая диаграмма загрузки имеет копредел с вершиной *S*, а подстановки – с вершиной *S'*. Копредел диаграммы соединения называется кодекартовым квадратом, он существует не всегда. Поэтому большое практическое значение имеют технологии комплексирования систем посредством соединения, гарантирующие получение адекватных результатов. Они активно разрабатываются в рамках компонентного подхода к проектированию [139].

Диаграммы категории *c-DESC*, копределы которых действительно отвечают актам сборки систем, называются конфигурациями и образуют класс, обозначаемый как *Conf*. В [175] класс конфигураций задается в форме отображения, сопоставляющего каждому набору объектов *CD* множество диаграмм *Conf*(*CD*), семейство объектов каждой из которых совпадает с *CD*. Наш класс *Conf* представляет собой объединение всех множеств вида *Conf*(*CD*).

Чтобы формализовать многошаговые процессы синтеза систем, в ходе которых одни диаграммы преобразуются в другие, класс конфигураций сам рассматривается как категория (ковариантная категория «сверхзапятой» [93]). В дальнейшем мы будем даже строить диаграммы, состоящие из диаграмм – они позволяют формализовать процессы синтеза мегамоделей. Несложным примером приложения этих конструкций служит задача выявления архитектуры (architecture mining) системы *S* – нахождение ее конфигурации, оптимальной с точки зрения некоторого метода архитектурной декомпозиции. Если метод формализуется правилом построения некоторого подкласса в *Conf*, то оптимальной («минимальной»/«максимальной») конфигурацией будет универсальный (инициальный/терминальный) объект в категории, состоящей из всех диаграмм, содержащихся в этом подклассе и имеющих копредел с вершиной *S*. Действительно, инициальная конфигурация, единственным образом входящая в любую другую, представляет собой апробацию (proof of concept) метода, а терминальная, единственным образом поглощающая в себя любую другую – наилучшее приближение (best approximation), получаемое с его помощью. Такой критерий оптимизируемости распространяется на методы декомпозиции более общего вида, которые формализуются классами коконусов, не обязательно представляющих копределы конфигураций.

Перейдем к точным определениям. Мы используем (элементарные) теоретико-категорные понятия, определения которых можно найти в книгах [93, 136]. В частности, в [136] объекты (морфизмы) категории C кратко называются C-объектами (C-морфизмами), а функтор вида $\Delta : X \to C$ – C-диаграммой со схемой X. Диаграмма часто изображается как граф категории X, вершины которого помечены C-объектами, а ребра – C-морфизмами. Мы будем предполагать, что X – малая категория (т.е. ее класс объектов представляет собой множество). Категория всех малых категорий и всех их функторов обозначается через **Cat**, она является полной подкатегорией в категории **CAT** всех категорий и всех функторов. (На самом деле **CAT** настолько «велика», что в книге [136] она названа квазикатегорией, но для целей настоящей работы это не существенно). Диаграмма называется конечной, если таковой является ее схема.

В настоящей работе ключевую роль играет конструкция категории запятой (comma category [93, с.58]). «Строительным материалом» для нее служит любая пара функторов с общей кообластью:

$$fc : D \to E \leftarrow C : gc.$$

Объектами категории запятой $fc \downarrow gc$ являются все тройки $\langle X, Y, f\rangle$, где $X \in \mathrm{Ob}\, D$, $Y \in \mathrm{Ob}\, C$, $f : fc(X) \to gc(Y) \in \mathrm{Mor}\, E$. Морфизмом $(fc \downarrow gc)$-объекта $\langle X_1, Y_1, f_1\rangle$ в $\langle X_2, Y_2, f_2\rangle$ является любая пара $\langle p : X_1 \to X_2 \in \mathrm{Mor}\, D,\ q : Y_1 \to Y_2 \in \mathrm{Mor}\, C\rangle$ такая, что $f_2 \circ fc(p) = gc(q) \circ f_1$, с покомпонентным законом композиции.

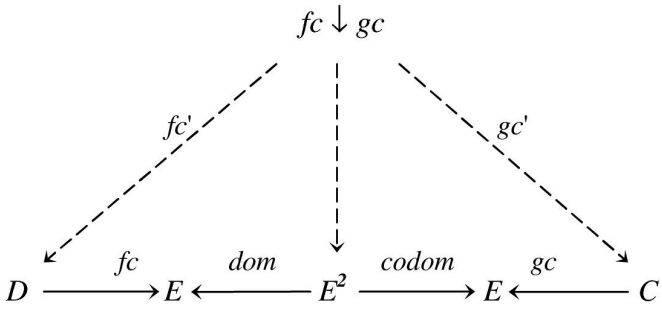
$$\begin{array}{ccccc}
\langle X_1, & Y_1, & fc(X_1) & \xrightarrow{f_1} & gc(Y_1)\rangle \\
\downarrow p & \downarrow q & \downarrow fc(p) & & \downarrow gc(q) \\
\langle X_2, & Y_2, & fc(X_2) & \xrightarrow{f_2} & gc(Y_2)\rangle
\end{array}$$

Имеются канонические «забывающие» функторы

$$fc' : (fc \downarrow gc) \to D : \langle X, Y, f\rangle \mapsto X, \langle p, q\rangle \mapsto p,$$

$$gc' : (fc \downarrow gc) \to C : \langle X, Y, f\rangle \mapsto Y, \langle p, q\rangle \mapsto q.$$

Категория запятой вида $1_E \downarrow 1_E$, порожденная двумя тождественными функторами, представляет собой категорию стрелок над E (поскольку класс всех ее объектов совпадает с $\mathrm{Mor}\, E$), обозначаемую также через E^2. Соответствующие ей забывающие функторы обозначаются (по очевидной причине) через *dom* и *codom*, соответственно. Произвольную категорию запятой $fc \downarrow gc$ можно построить при помощи категории стрелок как универсальную конструкцию в **CAT** – вершину предела следующего вида:



Ребра этого предела, направленные в D и в C, представляют собой канонические забывающие функторы, а в E^2 – еще один забывающий функтор, переводящий тройку $\langle X, Y, f\rangle$ в f.

Класс всех C-диаграмм служит классом объектов категории, конструкция которой обобщает конструкцию категории запятой. За основу возьмем категорию запятой вида $D \downarrow S$, где D – подкатегория в некоторой категории E, S – произвольный E-объект (т.е. в роли fc выступает вложение в E подкатегории D, а в роли gc – вложение сингулярной подкатегории $(\{S\}, \{1_S\})$). Напомним, что класс ее объектов состоит из всех E-морфизмов, направленных из D-объектов в S, а морфизмом объекта $f : X \to S$ в $g : Y \to S$ служит любой D-морфизм $d : X \to Y$, образующий коммутативный E-треугольник $f = g \circ d$. Имеется функтор $dom : (D \downarrow S) \to D : f \mapsto \mathrm{dom}\, f, d \mapsto d$, «забывающий» про S (а также постоянный функтор $codom : f \mapsto S, d \mapsto 1_S$, забывающий про все, кроме S). C-диаграммы рассматриваются как объекты категории запятой $\mathbf{Cat} \downarrow C$, однако понятие морфизма для диаграмм расширяется: коммутативность треугольника допускается с точностью до естественного преобразования. Напомним, что естественным преобразованием ε функтора $fun : F \to G$ в $fun' : F \to G$ называется семейство G-морфизмов $\varepsilon_A : fun(A) \to fun'(A)$, $A \in \mathrm{Ob}\, F$, такое, что для любого F-морфизма $f : A \to B$ выполняется соотношение $\varepsilon_B \circ fun(f) = fun'(f) \circ \varepsilon_A$. Морфизмом диаграммы $\Delta : X \to C$ в $\Xi : Y \to C$ служит любая пара вида $\langle \varepsilon, fd\rangle$, состоящая из функтора $fd : X \to Y$ и естественного преобразования $\varepsilon : \Delta \to \Xi \circ fd$; закон композиции имеет вид $\langle \varepsilon, fd\rangle \circ \langle \gamma, gd\rangle = \langle \varepsilon_{gd} \circ \gamma, fd \circ gd\rangle$. Категория диаграмм, определенная таким способом, обозначается через $\mathbf{D}C$. Имеется функтор $sch : \mathbf{D}C \to \mathbf{Cat} : \Delta \mapsto \mathrm{dom}\, \Delta, \langle \varepsilon, fd\rangle \mapsto fd$. Любой функтор $fun : C \to D$ определяет функтор

$$fun \circ - : \mathbf{D}C \to \mathbf{D}D : \Delta \mapsto fun \circ \Delta, \langle \varepsilon, fd\rangle \mapsto \langle fun(\varepsilon), fd\rangle,$$

так что имеется эндофунктор

$$\mathbf{D} : \mathbf{CAT} \to \mathbf{CAT} : C \mapsto \mathbf{D}C, fun \mapsto (fun \circ -).$$

Категория $\mathbf{D}C$ является частным случаем конструкции Гротендика (Grothendieck construction [145, п.12.2.10]) для функтора $C^- : \mathbf{Cat}^{\mathrm{op}} \to \mathbf{CAT} : X \mapsto C^X$, а функтор sch представляет собой соответствующее каноническое расслоение (split fibration). Из этого факта вытекает ряд

важных свойств категории диаграмм. Имеется вложение $ic : (\mathbf{Cat} \downarrow C) \hookrightarrow \mathbf{D}C$, переводящее функтор fd, удовлетворяющий условию $\Delta = \Xi \circ fd$, где Δ, Ξ – C-диаграммы, в $\mathbf{D}C$-морфизм $\langle 1_\Delta, fd\rangle$. При этом если функтор fd является вложением, то Δ называется поддиаграммой в Ξ (ввиду того, что граф Δ является подграфом графа Ξ). Также в $\mathbf{D}C$ вкладывается любая категория вида C^X, состоящая из всех функторов, действующих из категории X в C, и всех их естественных преобразований: естественному преобразованию ε соответствует $\mathbf{D}C$-морфизм $\langle \varepsilon, 1_X\rangle$. Категории функторов вместе с $\mathbf{Cat} \downarrow C$ порождают $\mathbf{D}C$, поскольку любой $\mathbf{D}C$-морфизм допускает разложение

$$\langle \varepsilon, fd\rangle = ic(fd) \circ \langle \varepsilon, 1_{sch(\Delta)}\rangle : \Delta \to \Xi \circ fd \to \Xi.$$

Поскольку $C^{\mathbf{1}} \cong C$, где $\mathbf{1}$ – терминальная категория, состоящая из одного объекта 0 и морфизма 1_0, то имеется (полное) вложение C в $\mathbf{D}C$, которое мы будем обозначать через $\ulcorner\_\urcorner$. Диаграммы вида $\ulcorner S\urcorner$, где S – произвольный C-объект, мы будем называть точками. Определим конструкцию коконуса, играющую ключевую роль в формальном анализе комплексирования систем.

Определение 2.1. *Коконусом* называется $\mathbf{D}C$-морфизм, имеющий точку в качестве кообласти. □

Зафиксируем произвольную диаграмму $\Delta : X \to C$. В литературе коконус с областью Δ часто определяют как естественное преобразование из Δ в постоянный функтор, действующий из X в C (см. например [93, с.81-82]). Это определение эквивалентно нашему во всех случаях, за исключением того, когда X – пустая категория. В этом последнем случае существует в точности один функтор, действующий из X в C (это пустая диаграмма), поэтому нельзя определить инициальный объект как вершину универсального коконуса с пустой диаграммой в качестве области.

Область Δ коконуса $\langle \sigma, !_X : X \to \mathbf{1}\rangle : \Delta \to \ulcorner S\urcorner$ называется его основанием, объект S – вершиной, компоненты естественного преобразования σ – ребрами. Коконус можно рассматривать как объект категории запятой $(\mathbf{D}C) \downarrow \ulcorner S\urcorner$. В то же время он является диаграммой со схемой X в категории запятой $C \downarrow S$, получающейся из Δ путем замены объектов исходящими из них ребрами. Эти представления коконусов эквивалентны:

Предложение 2.1. $((\mathbf{D}C) \downarrow \ulcorner S\urcorner) \cong \mathbf{D}(C \downarrow S)$ для любого C-объекта S.

Доказательство. Любой коконус $\langle \sigma, !_X\rangle : \Delta \to \ulcorner S\urcorner$ взаимно однозначно определяет $(C \downarrow S)$-диаграмму, переводящую любой X-объект I в σ_I и любой X-морфизм f в $\Delta(f)$. □

Пользуясь этим фактом, легко определить понятие *подкоконуса*: коконус θ является подкоконусом коконуса δ с вершиной S, если θ является поддиаграммой δ в $\mathbf{D}(C \downarrow S)$. Аналогично

определяется изоморфизм коконусов с вершиной S: два коконуса являются изоморфными, если существует $\mathbf{D}(C \downarrow S)$-изоморфизм между ними.

Для произвольного класса коконусов Cd будем обозначать через $Cd \downarrow S$ полную подкатегорию в $(\mathbf{D}C) \downarrow \ulcorner S \urcorner$, класс объектов которой состоит из всех коконусов из Cd, имеющих вершину S. Эта конструкция используется для формализации задачи оптимизации архитектуры.

Определение 2.2. Класс коконусов Cd называется *оптимизируемым*, если любая категория $Cd \downarrow S$ обладает инициальным и терминальным объектами. □

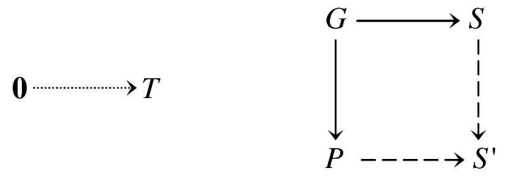
Например, класс всех коконусов, который мы будем обозначать через Cocone C, оптимизируем: инициальным объектом в категории $(\mathrm{Cocone}\ C) \downarrow S$ является единственный $\mathbf{D}C$-морфизм, направленный из пустой диаграммы в $\ulcorner S \urcorner$, а терминальным – $\ulcorner 1_S \urcorner$.

Копределом диаграммы Δ называется коконус $\operatorname{colim}\Delta : \Delta \to \ulcorner S \urcorner$, универсальный в том смысле, что для любых C-объекта T и коконуса $\delta : \Delta \to \ulcorner T \urcorner$ существует единственный C-морфизм $u : S \to T$ такой, что $\delta = \ulcorner u \urcorner \circ \operatorname{colim}\Delta$. При этом S называется объектом копредела, а u – стрелкой копредела. Копредел определяется однозначно с точностью до изоморфизма: коконус δ является копределом диаграммы Δ тогда и только тогда, когда существует C-изоморфизм i такой, что $\delta = \ulcorner i \urcorner \circ \operatorname{colim}\Delta$. В свою очередь, если $\mu : \Xi \to \Delta$ – $\mathbf{D}C$-изоморфизм, то $(\operatorname{colim}\Delta) \circ \mu$ – копредел диаграммы Ξ. Наконец, если $iso : C \to D$ – изоморфизм категорий, то $iso \circ (\operatorname{colim}\Delta)$ – копредел диаграммы $iso \circ \Delta$. Конструкция копредела естественна относительно произведения категорий: если $\Delta' : X \to C'$ – диаграмма в некоторой категории C' с той же схемой X, что и Δ, то произвольные коконусы $\langle \sigma, !_X \rangle : \Delta \to \ulcorner S \urcorner$ и $\langle \sigma', !_X \rangle : \Delta' \to \ulcorner S \urcorner$ являются копределами тогда и только тогда, когда коконус $\langle ((\sigma_I, \sigma'_I))_{I \in \mathrm{Ob}\, X}, !_X \rangle : \langle \Delta, \Delta' \rangle \to \ulcorner (S, S') \urcorner$ является копределом диаграммы $\langle \Delta, \Delta' \rangle : X \to C \times C'$.

Если X – пустая категория, то объект копредела представляет собой в точности инициальный C-объект, обычно обозначаемый через $\mathbf{0}$: из него в любой C-объект идет в точности один морфизм – стрелка копредела. Если X – дискретная категория (т.е. не имеющая нетождественных морфизмов), то объект копредела называется копроизведением семейства объектов $\Delta(\mathrm{Ob}\, X)$ и обозначается через $\coprod_{I \in \mathrm{Ob}\, X} \Delta(I)$, а ребра копредела – инъекциями компонентов. Напомним в связи с этим, что все дискретные категории образуют полную корефлексивную подкатегорию в **CAT**, которую мы обозначим через **d-CAT**: имеется канонический функтор $|\text{–}| : \mathbf{CAT} \to \mathbf{d\text{-}CAT}$, «забывающий» все нетождественные морфизмы, и его естественное преобразование ι в тождественный функтор $1_{\mathbf{CAT}}$, состоящее из вложений, тождественных на объектах. Функтор $|\text{–}|$ переводит диаграммы в свои поддиаграммы: соотношение $\iota_C \circ |\Delta| = \Delta \circ \iota_X$ определяет $(\mathbf{Cat} \downarrow C)$-морфизм $\iota_X : \iota_C \circ |\Delta| \hookrightarrow \Delta$, порождающий вложение диаграмм $\iota_\Delta = ic(\iota_X)$.

Еще один важный класс копределов известен под названием кодекартовых квадратов. Обозначим через $\boldsymbol{V}$ частично упорядоченное множество $\{1 \leftarrow 0 \rightarrow 1'\}$ (с несравнимыми элементами 1 и 1'), рассматриваемое как категория. Произвольная пара морфизмов с общей областью $a : A \leftarrow G \rightarrow B : b$ задает диаграмму со схемой $\boldsymbol{V}$, которую мы рассмотрим в качестве Δ. Любой коконус $\langle \sigma, !_V \rangle : \Delta \rightarrow \ulcorner S \urcorner$ однозначно определяется своими ребрами $\sigma_1 : A \rightarrow S$ и $\sigma_{1'} : B \rightarrow S$, поскольку выполняется соотношение $\sigma_1 \circ a = \sigma_{1'} \circ b$, задающее коммутативный квадрат в C (и определяющее третье ребро коконуса). Если такой коконус является копределом, то задающий его квадрат называется кодекартовым и характеризуется тем, что для любого коммутативного квадрата вида $u \circ a = v \circ b$ существует единственный морфизм w, удовлетворяющий соотношениям $w \circ \sigma_1 = u$ и $w \circ \sigma_{1'} = v$ – это в точности стрелка копредела. При помощи кодекартова квадрата можно определить понятие эпиморфизма: морфизм $e : A \rightarrow B$ называется эпиморфизмом, если соотношение $1_B \circ e = 1_B \circ e$ задает кодекартов квадрат. Это означает, что для любой пары морфизмов $s, t : B \rightrightarrows Z$ условие $s \circ e = t \circ e$ влечет $s = t$.

Если категория C обладает кодекартовыми квадратами (т.е. любая C-диаграмма со схемой $\boldsymbol{V}$ имеет копредел) и инициальным объектом, то любая конечная C-диаграмма имеет копредел (см. двойственное утверждение в [136, теорема 12.4]). Этот факт можно истолковать в контексте комплексирования систем, поскольку кодекартов квадрат представляет копредел диаграммы соединения из разд.2.1, причем его ребра образуют диаграмму подстановки (т.е. можно рассматривать результат интеграции «клея» с компонентом и системой как изменение системы, приводящее к появлению возможности интегрировать в нее компонент). В свою очередь, инициальный объект можно трактовать как универсальную единицу загрузки, задаваемой исходящими из него морфизмами. Из приведенного утверждения следует, что любой конечный акт сборки системы можно скомбинировать из трех приемов интеграции части в целое: изменение части (загрузка), изменение целого (подстановка) и добавление «клея» (соединение).



Для построения копределов произвольных диаграмм требуется еще одна конструкция – коуравнитель. Возьмем в качестве Δ диаграмму, схема которой состоит из двух объектов и двух параллельных морфизмов одного из них в другой: она задается парой морфизмов вида $f, g : P \rightrightarrows Q$. Любой коконус с основанием Δ однозначно определяется своим ребром q, исходящим из Q, поскольку выполняется соотношение $q \circ f = q \circ g$ (определяющее второе ребро). Если такой коконус является копределом, то это ребро называется коуравнителем пары f, g и характеризуется тем, что для любого морфизма r такого, что $r \circ f = r \circ g$, существует единственный

морфизм s, удовлетворяющий соотношению $s \circ p = r$ – это в точности стрелка копредела. Коуравнитель любой пары является эпиморфизмом. Морфизм, выступающий коуравнителем для некоторой пары, называется регулярным эпиморфизмом. Он часто рассматривается как категорный аналог факторизации – канонического отображения на фактор-объект по отношению эквивалентности, совместимому со структурой объекта-области.

Кодекартов квадрат, задающий копредел произвольной пары $a : A \leftarrow G \rightarrow B : b$, можно вычислить через копроизведение и коуравнитель: он имеет вид $(q \circ i_A) \circ a = (q \circ i_B) \circ b$, где $i_A : A \rightarrow A \coprod B$, $i_B : B \rightarrow A \coprod B$ – инъекции, q – коуравнитель пары $(i_A \circ a), (i_B \circ b) : G \rightrightarrows A \coprod B$. Более того, копредел любой диаграммы, если он существует, может быть вычислен через копроизведения и коуравнители (см. двойственное утверждение в [136, теорема 12.3]).

В работе требуется рассматривать функторы с точки зрения их воздействия на копределы, т.е. степени естественности относительно актов сборки систем. Согласно [136, разд.13], говорится, что функтор $fun : C \rightarrow D$:

- *сохраняет* (preserves) копределы диаграммы Δ, если для любого копредела $\delta : \Delta \rightarrow \ulcorner S \urcorner$ коконус $fun \circ \delta : fun \circ \Delta \rightarrow \ulcorner fun(S) \urcorner$ является копределом диаграммы $fun \circ \Delta$;
- *поднимает* (lifts) копределы диаграммы Δ, если для любого копредела $\xi : fun \circ \Delta \rightarrow \ulcorner T \urcorner$ существует копредел $\delta : \Delta \rightarrow \ulcorner S \urcorner$ такой, что $fun \circ \delta = \xi$.

Будем говорить, что функтор *детерминирует* копределы диаграммы Δ, имеющей копредел, если он сохраняет и поднимает их. Способность функтора детерминировать копределы означает, что они естественны относительно него в весьма «сильном» смысле. Например, приведенное выше утверждение о естественности копределов относительно произведения равносильно тому, что функтор-проекция произведения категорий на любую компоненту детерминирует копределы всех диаграмм, обладающих ими. Такой же способностью детерминировать копределы обладает функтор-инъекция любой компоненты в сумму категорий.

Имеются следующие простые критерии проверки способности произвольного функтора детерминировать копределы.

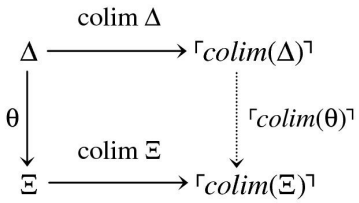
Предложение 2.2. Функтор fun детерминирует копределы диаграммы Δ тогда и только тогда, когда он поднимает их и диаграмма $fun \circ \Delta$ имеет копредел.

Доказательство. Необходимость очевидна, докажем достаточность. Пусть δ – произвольный копредел диаграммы Δ, ξ – произвольный копредел диаграммы $fun \circ \Delta$. По условию Δ имеет копредел δ' такой, что $fun \circ \delta' = \xi$. Имеем $\delta = \ulcorner i \urcorner \circ \delta'$ для некоторого изоморфизма i, поэтому $fun \circ \delta = \ulcorner fun(i) \urcorner \circ \xi$ является копределом диаграммы $fun \circ \Delta$. □

Предложение 2.3. Пусть функтор $fun : C \to D$ детерминирует копределы C-диаграммы Δ. Функтор $fun' : D \to E$ детерминирует копределы D-диаграммы $fun \circ \Delta$ тогда и только тогда, когда функтор $fun' \circ fun$ детерминирует копределы диаграммы Δ.

Доказательство. Необходимость очевидна, докажем достаточность. Если ξ – копредел диаграммы $fun' \circ fun \circ \Delta$, то по предположению существует копредел δ диаграммы Δ такой, что $fun' \circ fun \circ \delta = \xi$. Тогда $fun \circ \delta$ – копредел диаграммы $fun \circ \Delta$, переходящий в ξ под действием функтора $\mathbf{D}fun'$. Таким образом, функтор fun' поднимает копределы диаграммы $fun \circ \Delta$, поэтому ввиду предложения 2.2 детерминирует их. □

Пусть *Cdia* – некоторый класс C-диаграмм, имеющих копределы. Можно рассматривать его как полную подкатегорию в $\mathbf{D}C$, и определить функтор копредела *colim*, действующий из него в C, сопоставляя каждой диаграмме из *Cdia* объект некоторого ее копредела, а каждому $\mathbf{D}C$-морфизму $\theta : \Delta \to \Xi$, где $\Delta, \Xi \in Cdia$ – стрелку копредела $colim(\theta)$ такую, что $\operatorname{colim} \Xi \circ \theta = \ulcorner colim(\theta) \urcorner \circ \operatorname{colim} \Delta$.



Теперь кратко рассмотрим диаграммы, составленные из диаграмм, т.е. перейдем в категорию $\mathbf{DD}C$. Поскольку **Cat** кополна [148], $\mathbf{D}C$ имеет некоторые копределы (в частности, инициальный объект и копроизведения) независимо от существования каких-либо копределов в C. Это вытекает из следующего утверждения.

Предложение 2.4. $\mathbf{D}C$-диаграмма $ic \circ \Gamma$ имеет копредел для любой $(\mathbf{Cat} \downarrow C)$-диаграммы Γ, причем функтор $\mathbf{D}fun$ сохраняет его для произвольного функтора $fun : C \to D$.

Доказательство. Копредел диаграммы $\Gamma : X \to (\mathbf{Cat} \downarrow C)$ строится, как в любой категории запятой: он имеет вид $\operatorname{colim} \Gamma' : \Gamma \to \ulcorner \Pi \urcorner$, где $\Gamma' = dom \circ \Gamma : X \to \mathbf{Cat}$, $\Pi : colim(\Gamma') \to C$ – единственный функтор, удовлетворяющий условию $\delta = \ulcorner \Pi \urcorner \circ \operatorname{colim} \Gamma'$, $\delta : \Gamma' \to \ulcorner C \urcorner$ – коконус, изоморфный Γ в смысле предложения 2.1. Остается заметить, что функтор ic сохраняет копределы: в этом нетрудно убедиться, используя явную конструкцию копредела в **Cat**, описанную в [148]. А поскольку $fun \circ (ic \circ \operatorname{Mor}(\mathbf{Cat} \downarrow C)) \subseteq ic \circ \operatorname{Mor}(\mathbf{Cat} \downarrow D)$, коконус $\mathbf{D}fun \circ (ic \circ \operatorname{colim} \Gamma')$ является копределом диаграммы $\mathbf{D}fun \circ (ic \circ \Gamma)$. □

Используя копределы в $\mathbf{Cat} \downarrow C$, можно построить естественную проекцию (*функтор отрисовки*) $\mathbf{K} : \mathbf{DD}C \to \mathbf{D}C$. Грубо говоря, отрисовка диаграммы $\Gamma : X \to \mathbf{D}C$ заключается в замене каждой вершины A графа категории X графом C-диаграммы $\Gamma(A)$, а каждого ребра $f : A \to B$ – совокупностью ребер, по одному для каждой вершины I графа диаграммы $\Gamma(A)$,

направленному из I в вершину $fd(I)$ графа $\Gamma(B)$ и помеченному C-морфизмом ε_I, где $\langle \varepsilon, fd\rangle = \Gamma(f)$. Отрисовка естественна относительно вложения ic в том смысле, что $\mathbf{K}(ic \circ \mathrm{Mor}\,(\mathbf{Cat} \downarrow \mathbf{D}C)) \subseteq ic \circ \mathrm{Mor}\,(\mathbf{Cat} \downarrow C)$. Читателей, искушенных в теории категорий, отошлем за строгим определением отрисовки к работе [186]: в ней показано, что тройка $\langle \mathbf{D}, \ulcorner - \urcorner, \mathbf{K}\rangle$ образует монаду в **CAT**, по своим свойствам аналогичную монаде степени в категории **Set**, имеющей вид $\langle 2^{-}, \{-\}, \bigcup\rangle$.

Отрисовка позволяет рассматривать коконусы как диаграммы. Будем обозначать через $\llcorner - \lrcorner$ вложение категории морфизмов $C^{\boldsymbol{2}}$ в $\mathbf{D}C$ (напомним, что символом $\boldsymbol{2}$ обозначается двухэлементное линейно упорядоченное множество $\{0 \to 1\}$, рассматриваемое как категория, так что диаграммы вида $\boldsymbol{2} \to C$ находятся во взаимно-однозначном соответствии с C-морфизмами). Будем говорить, что C-диаграмма *имеет форму коконуса*, если она имеет вид $\mathbf{K}\llcorner\delta\lrcorner$ для некоторого коконуса δ (здесь $\llcorner - \lrcorner : (\mathbf{D}C)^{\boldsymbol{2}} \hookrightarrow \mathbf{DD}C$). Ясно, что диаграмма имеет форму коконуса тогда и только тогда, когда ее схема обладает терминальным объектом, из которого не исходит ни одного нетождественного морфизма. Такой объект является единственным, и ему соответствует вершина коконуса. Например, форму коконусов имеют диаграммы приемов загрузки и подстановки из разд.2.1: первая имеет вершину S, а вторая – S'. Кодекартов квадрат также представляет собой диаграмму в форме коконуса. Копредел диаграммы, имеющей форму произвольного коконуса $\delta : \Delta \to \ulcorner S \urcorner$, существует и имеет вид $\mathbf{K}(\langle \bar{\delta}, !_{\boldsymbol{2}}\rangle : \llcorner\delta\lrcorner \to \ulcorner\ulcorner S \urcorner\urcorner)$, где $\bar{\delta} : \delta \to \ulcorner 1_S \urcorner \in \mathrm{Mor}\,(\mathbf{D}C)^{\boldsymbol{2}}$ – естественное преобразование с компонентами $\delta_0 = \delta$ и $\bar{\delta}_1 = \ulcorner 1_S \urcorner$. Отрисовка индуцирует вложение категорий вида $(\mathbf{D}C) \downarrow \ulcorner S \urcorner$ в $\mathbf{D}C$: различные коконусы имеют различные отрисовки, а $((\mathbf{D}C) \downarrow \ulcorner S \urcorner)$-морфизму коконусов отвечает $\mathbf{D}C$-морфизм их отрисовок, переводящий вершину в вершину (причем обязательно путем тождественного морфизма 1_S), а объекты основания – в объекты основания.

Эндофунктор **D** сохраняет сопряжение функторов – одну из важнейших конструкций теории категорий. Напомним [93, гл.4], что для произвольных категорий C, D и функторов $fun : C \to D$, $fad : D \to C$ сопряжение $fad \dashv fun$ – это семейство биекций $\Phi : \mathrm{Mor}(fad(S), R) \cong \mathrm{Mor}(S, fun(R))$, $S \in \mathrm{Ob}\,D$, $R \in \mathrm{Ob}\,C$, естественное в том смысле, что для любых $f : fad(S) \to R$, $h : X \to S$ и $k : R \to Y$ выполняется соотношение $\Phi(k \circ f \circ fad(h)) = fun(k) \circ \Phi(f) \circ h$. Семейство C-морфизмов $\varepsilon_A = \Phi^{-1}(1_{fun(A)}) : fad(fun(A)) \to A$, $A \in \mathrm{Ob}\,C$, называется коединицей сопряжения: для любого $f : fad(S) \to R$ имеем $\Phi(1_R \circ \varepsilon_R \circ fad(\Phi(f))) = fun(1_R) \circ \Phi(\varepsilon_R) \circ \Phi(f) = \Phi(f)$, откуда получается «треугольное тождество» сопряжения $\varepsilon_R \circ fad(\Phi(f)) = f$. Это тождество позволяет вычислить действие сопряжения, если известна его коединица. Естественность сопряжения проявляется в том, что коединица представляет собой естественное преобразование эндофунктора $fad \circ fun$ в 1_C: для любого $s : A \to B$ имеем $\Phi(s \circ \varepsilon_A) = \Phi(s \circ \varepsilon_A \circ fad(1_{fun(A)})) = fun(s) \circ \Phi(\varepsilon_A) \circ 1_{fun(A)} = fun(s)$,

откуда, подставляя морфизм $s \circ \varepsilon_A : fad(fun(A)) \to B$ в треугольное тождество, получаем $\varepsilon_B \circ fad(fun(s)) = s \circ \varepsilon_A$. Двойственно, семейство D-морфизмов $\eta_G = \Phi(1_{fad(G)}) : G \to fun(fad(G))$, $G \in \mathrm{Ob}\, D$, образует единицу сопряжения – естественное преобразование функтора 1_D в $fun \circ fad$, порождающее второе треугольное тождество $fun(\Phi^{-1}(g)) \circ \eta_S = g$, $g : S \to fun(R)$.

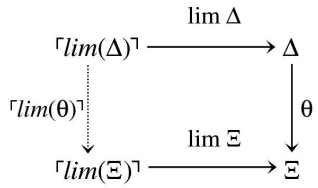
Очевидным примером сопряжения служит изоморфизм категорий – он сопряжен к своему обратному (как справа, так и слева), причем единица и коединица этого сопряжения состоят из тождественных морфизмов. В более общем случае, когда единица некоторого сопряжения $fad \dashv fun$ тождественна, второе треугольное тождество приобретает вид $fun(\Phi^{-1}(g)) = g$, откуда $\Phi(f) = fun(f)$ для любого $f : fad(S) \to R$, т.е. биекция сопряжения действует, как правый сопряженный функтор. В свою очередь, ввиду соотношения $fun \circ fad = 1_D$ левый сопряженный задает полное вложение категории D в C, и если D – подкатегория в C, то такое сопряжение является корефлексией (а функтор fun – корефлектором). Двойственно, если коединица тождественна, то биекция Φ^{-1} действует как левый сопряженный функтор, правый сопряженный ввиду соотношения $fad \circ fun = 1_C$ задает полное вложение категории C в D, и если C – подкатегория в D, то такое сопряжение является рефлексией с рефлектором fad [93, разд.4.3]. Сопряжения такого типа играют важную роль в настоящей работе.

Содержательно, сопряжение функторов, действующих между категориями, представляющими процессы синтеза систем, описывает процедуры двунаправленного согласования процессов. В частности, функтор, находящийся слева в некотором сопряжении, сохраняет копределы всех диаграмм [93, с.141]. Любое сопряжение $fad \dashv fun : C \to D$ индуцирует сопряжение $\mathbf{D}fad \dashv \mathbf{D}fun : \mathbf{D}C \to \mathbf{D}D$ – легко проверить, что оно имеет вид $\mathbf{D}\Phi : \langle \gamma, fd \rangle \mapsto \langle \Phi\gamma, fd \rangle$.

В заключительной части раздела кратко рассмотрим понятие предела – универсальной конструкции, двойственной по отношению к копределу (и, как видно из ее названия, первичной с точки зрения «чистой» теории категорий). Предел определяется в контравариантной категории диаграмм (категория «сверхзапятой» [93]), которую мы будем обозначать через $\overline{\mathbf{D}}C$ для произвольной категории C. Класс объектов категории $\overline{\mathbf{D}}C$ состоит из всех C-диаграмм, как и $\mathbf{D}C$, но морфизмом диаграммы $\Delta : X \to C$ в $\Xi : Y \to C$ здесь служит любая пара вида $\langle \varepsilon, fh \rangle$, состоящая из функтора $fh : Y \to X$ и естественного преобразования $\varepsilon : \Delta \circ fh \to \Xi$. Имеется эндофунктор

$$\overline{\mathbf{D}} : \mathbf{CAT} \to \mathbf{CAT} : C \mapsto \overline{\mathbf{D}}C,\ fun \mapsto (fun \circ -).$$

Любая категория вида C^X вкладывается в $\overline{\mathbf{D}}C$ таким же способом, как в $\mathbf{D}C$, в частности имеются полные вложения $\ulcorner - \urcorner : C \hookrightarrow \overline{\mathbf{D}}C$, $\llcorner - \lrcorner : C^2 \hookrightarrow \overline{\mathbf{D}}C$. Конусом называется $\overline{\mathbf{D}}C$-морфизм, имеющий точку в качестве области. Пределом диаграммы $\Delta : X \to C$ называется конус $\lim \Delta : \ulcorner S \urcorner \to \Delta$, универсальный в том смысле, что для любых C-объекта T и конуса $\delta : \ulcorner T \urcorner \to \Delta$ существует единственный C-морфизм $u : T \to S$ (стрелка предела) такой, что $\delta = (\lim \Delta) \circ \ulcorner u \urcorner$. Предел, как и копредел, определяется однозначно с точностью до изоморфизма. Из любой подкатегории в $\overline{\mathbf{D}}C$, все объекты которой обладают пределами, в C действует функтор lim, сопоставляющий каждому $\overline{\mathbf{D}}C$-морфизму $\theta : \Delta \to \Xi$ стрелку предела $lim(\theta)$ такую, что $\theta \circ \lim \Delta = \lim \Xi \circ \ulcorner lim(\theta) \urcorner$.



Объект предела пустой диаграммы называется терминальным объектом и обычно обозначается через **1** (в него из любого объекта A идет единственный морфизм – стрелка предела, которая обозначается через $!_A : A \to \mathbf{1}$). Объект предела дискретной диаграммы Δ называется произведением ее объектов (и обозначается через $\prod_{I \in \mathrm{Ob}\, X} \Delta(I)$), а ребра такого предела – проекциями. Если объекты A и B имеют произведение, то для любой пары морфизмов $x : T \to A$, $y : T \to B$, образующих конус с основанием $\ulcorner A \urcorner \coprod \ulcorner B \urcorner$, стрелка предела обозначается через $\langle x, y \rangle : T \to A \times B$. Предел диаграммы со схемой $\boldsymbol{V}^{\mathrm{op}}$ задается коммутативным квадратом, который называется декартовым. В частности, морфизм $m : A \to B$ называется мономорфизмом, если соотношение $m \circ 1_A = m \circ 1_A$ задает декартов квадрат, т.е. для любой пары морфизмов $s, t : Z \rightrightarrows A$ условие $m \circ s = m \circ t$ влечет $s = t$. Предел диаграммы-пары $f, g : P \rightrightarrows Q$ задается ребром, входящим в P, которое называется ее уравнителем и всегда является мономорфизмом. Морфизм, выступающий уравнителем для некоторой пары, называется регулярным мономорфизмом, и часто рассматривается как категорный аналог включения, не разрушающего структуру своей области в составе кообласти. Предел любой диаграммы, если он существует, может быть вычислен через произведения и уравнители. Свойства функторов сохранять, поднимать, детерминировать пределы определяются точно так же, как для копределов. Выполняются для пределов аналоги предложений 2.2 и 2.3.

В настоящей работе некоторые пределы появятся в гл.4 в связи с теоретико-категорной формализацией аспектно-ориентированного подхода к созданию систем. Дело в том, что главной целью его привлечения является обеспечение трассирования, семантика которого строится при помощи морфизмов, двойственных по отношению к действиям по интеграции, а значит и двойственных универсальных конструкций.

2.3. Формальные технологии проектирования

Для более полного описания процессов синтеза систем в дополнение к конфигурациям необходимо ввести понятия интеграционного интерфейса и трансформации [175]. В частности, хорошо известно, что интеграционные возможности компонентов определяются их интерфейсами – специально подготовленными частями, видимыми «снаружи». Примером интерфейса служит описание сервиса на языке WSDL (Web Service Definition Language) – список процедур, реализованных в нем, с указанием параметров и возвращаемых значений. Формальные модели интерфейсов образуют категорию, обозначаемую через *SIG*, а операция выделения интерфейса формализуется как функтор $sig : c\text{-}DESC \to SIG$, называемый сигнатурным. Поскольку разные компоненты могут иметь один и тот же интерфейс, *sig* не обязан быть инъективным на объектах. Однако *sig*-образы двух различных действий, интегрирующих один и тот же компонент в одну и ту же систему, должны быть различными: иначе получится, что интерфейсы недостаточно детально описывают интеграционные возможности компонентов. Иными словами, функтор *sig* должен быть унивалентным (faithful), т.е. инъективным на каждом множестве вида Mor(*A*, *B*), состоящем из всех морфизмов с областью *A* и кообластью *B*.

Для каждого интерфейса гарантируется наличие хотя бы одной реализации, поддерживающей его интеграционные возможности в полном объеме. Она называется (*sig*-)дискретной и задается функтором $sig^* : SIG \to c\text{-}DESC$. Интерфейс дискретной реализации любого *SIG*-объекта *I* должен совпадать с *I*, поэтому должно выполняться соотношение $sig \circ sig^* = 1_{SIG}$. Полнота дискретной реализации заключается в том, что для любого *SIG*-объекта *I* и *c-DESC*-объекта *S* функтор *sig* сюръективно (следовательно, биективно) отображает множество Mor($sig^*(I)$, *S*), описывающее все действия по интеграции компонента $sig^*(I)$ в систему *S*, на множество Mor(*I*, *sig*(*S*)), определяющее интеграционные возможности интерфейса *I*. Эти условия эквивалентны тому, что функтор sig^* должен быть сопряженным слева к *sig* с тождественной единицей. В частности, он задает полное вложение категории *SIG* в *c-DESC* – изоморфизм на ее корефлективную подкатегорию (причем корефлектор действует как $sig^* \circ sig$). Например, дискретные реализации WSDL-описаний сервисов состоят из «заглушек» (stubs) – пустых про-

цедур; они могут автоматически генерироваться CASE-средствами и позволяют быстро собирать отладочные версии приложений.

Дискретные реализации интерфейсов сложных моделей, при всей своей тривиальности, могут комплексироваться из наборов компонентов – при этом взаимосвязи между ними сводятся к минимуму. Возможности интеграции *c-DESC*-объекта P в качестве компонента в дискретную реализацию *SIG*-объекта I описываются множеством морфизмов $\mathrm{Mor}(P, sig^*(I))$. Представляет практический интерес ситуация, когда эти возможности полностью определяются на уровне интерфейсов: в этом случае интерфейсы в полной мере задают поведение моделей в роли как компонентов, так и систем. Для этого должен существовать функтор $sig^{**} : c\text{-}DESC \to SIG$, биективно отображающий любое множество $\mathrm{Mor}(P, sig^*(I))$ на множество $\mathrm{Mor}(sig^{**}(P), I)$, причем $sig^{**} \circ sig^* = 1_{SIG}$. Этот функтор выявляет «дискретную структуру» компонента – интерфейс, скрывающий все внутренние взаимосвязи между его составными частями (поэтому на дискретные реализации он действует тривиально). Выражаясь языком теории категорий, функтор sig^{**} является сопряженным слева к sig^* с тождественной коединицей. Он не обязан существовать в произвольной технологии синтеза систем, однако мы будем выделять технологии, формализации которых позволяют его построить, называя их (дискретно) *структурируемыми*. В разд.2.4 мы приведем нетривиальный пример, свидетельствующий, что структурируемость тесно связана с возможностью моделировать распараллеливание систем.

Существует естественная взаимосвязь между интерфейсами и конфигурациями. Если две *c-DESC*-диаграммы имеют один и тот же **D***sig*-образ, то они обе должны либо принадлежать классу *Conf*, либо нет. Это условие задает своего рода логический закон непротиворечия для интерфейсов: совокупность всех наборов компонентов, имеющих фиксированную схему интеграции интерфейсов, не должна содержать как конфигурации, так и наборы, не порождающие целостных систем. Кроме того, выделение интерфейса должно быть естественным относительно комплексирования (т.е. коммутативным с ним) в том «сильном» смысле, что копредел **D***sig*-образа любой конфигурации должен быть **D***sig*-образом ее копредела (т.е. *sig* поднимает копределы конфигураций). Как мы увидим далее (предложение 2.6), из этого и других указанных выше свойств функтора *sig* вытекает естественность и в «слабом» смысле, более привычном в теории категорий: *sig* сохраняет копределы конфигураций (иными словами, **D***sig*-образ копредела любой конфигурации является копределом ее **D***sig*-образа). Таким образом, функтор *sig* детерминирует копределы всех конфигураций.

Формализация интерфейса позволяет описать специфицирование многокомпонентных систем. Однако полноценный процесс создания систем в дополнение к актам интеграции содержит трансформации (refinements) – шаги разработки индивидуальных компонентов (уточнение требований с изменением формы их описания, реализация спецификации на языке про-

граммирования и др.). Процесс рассматривается как идущий в двух измерениях: по горизонтали располагаются действия по интеграции, реализующие отношения «часть–целое», а по вертикали – трансформации, реализующие отношения «абстрактное–конкретное» [244]. Совокупность всех трансформаций задается как категория, обозначаемая *r-DESC*, она обладает теми же объектами, что и *c-DESC*, но другими морфизмами. Тривиальным частным случаем трансформации является *c-DESC*-изоморфизм: единицы комплексирования, несущественно отличающиеся друг от друга с точки зрения отношения «часть–целое», не должны отличаться и по отношению «абстрактное–конкретное» [175].

Условие естественности трансформаций относительно комплексирования систем налагается в следующей форме: любой набор трансформаций объектов некоторой конфигурации должен порождать трансформацию собираемой из них системы как целого. Это условие можно формализовать с привлечением естественных преобразований, состоящих из *r-DESC*-морфизмов, поскольку любая дискретная *c-DESC*-диаграмма является также *r-DESC*-диаграммой. Произвольный набор трансформаций объектов диаграммы $\Delta : X \to c\text{-}DESC$ представляет собой в точности естественное преобразование $\varphi : |\Delta| \to \Sigma \in \mathrm{Mor}\ r\text{-}DESC^{|X|}$ дискретной диаграммы $|\Delta|$ в дискретную диаграмму Σ, состоящую из результатов всех трансформаций. Если $\Delta \in Conf$, то Σ должна быть поддиаграммой некоторой конфигурации, объект копредела которой может быть получен путем трансформации из $colim(\Delta)$.

Изложенные соображения приводят к следующей сложной математической (формальной) конструкции, в рамках которой можно проводить всесторонний анализ процессов синтеза систем. Она почти полностью совпадает с предложенной в [175].

Определение 2.3. Пусть задана категория *c-DESC*, объектами которой служат формальные модели компонентов и систем, а морфизмами – действия по интеграции (компонентов в системы). *Формальной технологией проектирования* (architecture school) над *c-DESC* называется четверка $\langle c\text{-}DESC, Conf, sig, r\text{-}DESC\rangle$, где:

- *Conf* – класс *c-DESC*-диаграмм, называемых допустимыми конфигурациями систем (configurations);
- *sig* – функтор из *c-DESC* в категорию, обозначаемую *SIG*, объекты которой называются интерфейсами или сигнатурами компонентов (signatures), а морфизмы – действиями по интеграции интерфейсов;
- *r-DESC* – категория, объектами которой являются формальные модели, а морфизмы называются трансформациями компонентов.

При этом выполняются следующие условия.

(i) Любая диаграмма из класса *Conf* имеет копредел.

(ii) Функтор *sig* унивалентен.

(iii) Функтор *sig* обладает левым сопряженным, обозначаемым через *sig**, с тождественной единицей сопряжения.

(iv) Функтор *sig* поднимает копределы всех диаграмм из класса *Conf*.

(v) Для любых *c-DESC*-диаграмм Δ, Ξ, если $sig \circ \Delta = sig \circ \Xi$ и $\Delta \in Conf$, то $\Xi \in Conf$.

(vi) Ob *r-DESC* = Ob *c-DESC*.

(vii) Подкатегория в *c-DESC*, состоящая из всех *c-DESC*-объектов и всех изоморфизмов, является подкатегорией в *r-DESC*.

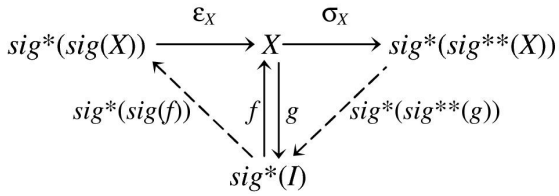
(viii) Для любых диаграммы $\Delta \in Conf$ и естественного преобразования вида $\varphi : |\Delta| \to \Sigma \in \mathrm{Mor}\ r\text{-}DESC^{|sch(\Delta)|}$ существуют диаграмма $\Delta \oplus \varphi \in Conf$, содержащая Σ в качестве поддиаграммы, и *r-DESC*-морфизм $r : colim(\Delta) \to colim(\Delta \oplus \varphi)$. □

На базе этой конструкции мы строим ряд производных.

Определение 2.4. Тройка $\langle c\text{-}DESC, Conf, sig\rangle$, удовлетворяющая условиям (i)-(v), называется *формальной технологией специфицирования*, а пара $\langle c\text{-}DESC, Conf\rangle$, удовлетворяющая условию (i) – *формальной технологией конфигурирования*. □

Определение 2.5. Формальная технология специфицирования либо проектирования называется (*дискретно*) *структурируемой*, если функтор *sig** обладает левым сопряженным с тождественной коединицей. □

Обозначим через ε коединицу сопряжения $sig^* \dashv sig$, через σ – единицу сопряжения $sig^{**} \dashv sig^*$ (если она существует). Действия этих естественных преобразований для произвольного *c-DESC*-объекта *X* («треугольные тождества» сопряжений) показаны на следующей диаграмме, где *I* – произвольный *SIG*-объект, *f* и *g* – произвольные *c-DESC*-морфизмы.



Произвольная категория *C* порождает тривиальную формальную технологию проектирования $\langle C, \varnothing, 1_C, (\mathrm{Ob}\ C, \mathrm{Iso}\ C)\rangle$, которую мы будем обозначать через ***triv***(*C*), причем она является структурируемой. Нетривиальные примеры формальных технологий приведены в работе [175], а также будут построены в ходе дальнейшего изложения. Формализация многих известных методов моделирования программ приводят к технологиям «над» категорией **Set**. Моделями в такой технологии служат множества, на которых задана некоторая структура (алгебраические системы, топологические пространства, графы и т.п.), а действиями по интеграции – отображения

множеств, совместимые со структурой (в подходящем смысле); *sig* является каноническим функтором взятия основного множества, «забывающим» структуру, а левый сопряженный к нему создает на произвольном множестве минимальную (дискретную) структуру. В качестве трансформаций обычно выступают антифункции – отношения, задающие «расширение» точек основного множества трансформируемых моделей до подмножеств. Мы подробно рассмотрим их в конце разд.2.5.

Характерным примером служит формальная технология моделирования сценариев поведения систем множествами событий, частично упорядоченными причинно-следственной связью [77]. Мы будем использовать ее для иллюстрации положений нашего подхода к синтезу систем на протяжении этой и следующей главы, и подробно изучим ее в разд.5.3. В конфигурациях сценариев результат комплексирования присутствует явно (с точностью до раздельного объединения). Трансформация сценария X в Y – это отношение $R \subseteq X \times Y$, удовлетворяющее следующим условиям:

- $\forall y \in Y\, \exists! x \in X\, xRy$ (антифункциональность);
- $\forall x \in X\, \exists y \in Y\, xRy$ (тотальность);
- $\forall x, x' \in X\, \forall y, y' \in Y\, ((xRy \wedge x'Ry' \wedge x \neq x') \Rightarrow (x \leq x' \Leftrightarrow y \leq y'))$.

Рассмотрим четверку

$$SM = \langle \mathbf{Pos}, CPos, |\text{–}|, r\text{-}Pos \rangle,$$

где **Pos** – категория всех частично упорядоченных множеств и всех их монотонных отображений; *CPos* – класс всех копроизведений **Pos**-коконусов; $|\text{–}| : \mathbf{Pos} \to \mathbf{Set}$ – канонический функтор, забывающий порядок; *r-Pos* – категория, состоящая из всех частично упорядоченных множеств и всех трансформаций сценариев. Покажем, что она является формальной технологией проектирования. Действительно, условие (iii) определения 2.3 обеспечивается функтором дискретного упорядочения множеств $id : \mathbf{Set} \to \mathbf{Pos} : S \mapsto \langle S, = \rangle$. В силу предложения 2.7 (см. ниже) тройка $SS = \langle \mathbf{Pos}, CPos, |\text{–}| \rangle$ является формальной технологией специфицирования. Далее, любой **Pos**-изоморфизм задает трансформацию сценариев. Справедливость условия (viii) вытекает из предложения 2.19, доказанного в разд.2.5 ниже.

В ходе дальнейшего изложения нам часто понадобится проверять, что та или иная теоретико-категорная конструкция является формальной технологией. В связи с этим целесообразно убедиться, что условия определения 2.3 независимы, т.е. что при проверке необходимо устанавливать справедливость каждого из них. В списке ниже перечислены четверки, каждая из которых удовлетворяет всем условиям (i)-(viii) за исключением того, номер которого совпадает с ее номером в списке.

(i) $\langle |\boldsymbol{2}|, \{1_{|\boldsymbol{2}|}\}, 1_{|\boldsymbol{2}|}, |\boldsymbol{2}|\rangle$ удовлетворяет условиям (ii)-(viii) (здесь подразумевается, что условие (viii) распространяется только на диаграммы, имеющие копредел);

(ii) $\langle \mathbf{Set}, \varnothing, !_{\mathbf{Set}}, \mathbf{Set}\rangle$;

(iii) $\langle |\boldsymbol{2}|, \varnothing, !_{|\boldsymbol{2}|}, |\boldsymbol{2}|\rangle$;

(iv) $\langle \mathbf{Pos}, \mathrm{Ob}\ \mathbf{D}(\mathbf{Pos}), |\text{–}|, (\mathrm{Ob}\ \mathbf{Pos}, \mathrm{Iso}\ \mathbf{Pos})\rangle$;

(v) $\langle \boldsymbol{2}, \{\ulcorner 0 \urcorner\}, !_{\boldsymbol{2}}, |\boldsymbol{2}|\rangle$;

(vi) $\langle \boldsymbol{1}, \varnothing, 1_{\boldsymbol{1}}, \boldsymbol{2}\rangle$;

(vii) $\langle \mathbf{Set}, \varnothing, 1_{\mathbf{Set}}, |\mathbf{Set}|\rangle$;

(viii) $\langle \boldsymbol{2}, \{\ulcorner 0 \urcorner\}, 1_{\boldsymbol{2}}, \boldsymbol{2}\rangle$.

Вернемся к рассмотрению произвольных формальных технологий. Докажем, что выделение интерфейсов перестановочно с комплексированием. Предварительно докажем одно утверждение, полезное для дальнейшего изложения. Зафиксируем произвольно категорию *c-DESC* и тройку

$$SC = \langle c\text{-}DESC, Conf \subseteq \mathrm{Ob}\ \mathbf{D}c\text{-}DESC, sig : c\text{-}DESC \to SIG\rangle.$$

Предложение 2.5. В любой формальной технологии специфицирования *SC* функтор *sig* сохраняет копределы любой диаграммы, все объекты которой *sig*-дискретны.

Доказательство. Пусть ε – коединица сопряжения $sig^* \dashv sig$; напомним, что ε – естественное преобразование функтора $sig^* \circ sig$ в $1_{c\text{-}DESC}$ такое, что ε_S является прообразом *SIG*-морфизма $1_{sig(S)}$ при биекции $sig : \mathrm{Mor}(sig^*(sig(S)), S) \cong \mathrm{Mor}(sig(S), sig(S))$ для любого $S \in \mathrm{Ob}\ c\text{-}DESC$, т.е. $sig(\varepsilon_S) = 1_{sig(S)}$. Из унивалентности функтора *sig* вытекает, что ε_S – эпиморфизм.

Выберем *c-DESC*-диаграмму Δ такую, что $\Delta(\mathrm{Ob}\ sch(\Delta)) \subseteq sig^*(\mathrm{Ob}\ SIG)$ (так что $sig^* \circ sig \circ \Delta = \Delta$), и ее копредел $\sigma : \Delta \to \ulcorner S \urcorner$. Рассмотрим коконус $\sigma^* = sig^* \circ sig \circ \sigma : \Delta \to \ulcorner S^* \urcorner$, где $S^* = sig^*(sig(S))$. Пусть $u : S \to S^*$ – стрелка копредела такая, что $\sigma^* = \ulcorner u \urcorner \circ \sigma$. Поскольку $\sigma = \ulcorner \varepsilon_S \urcorner \circ \sigma^*$, имеем $\varepsilon_S \circ u = 1_S$, так что ε_S, будучи обратимым справа мономорфизмом, является изоморфизмом, следовательно σ^* – копредел диаграммы Δ. Отсюда, ввиду унивалентности функтора sig^*, $sig \circ \sigma^* = sig \circ \sigma$ является копределом диаграммы $sig \circ \Delta$. □

Предложение 2.6. Если *SC* является технологией специфицирования, то функтор *sig* сохраняет копределы всех конфигураций.

Доказательство. Выберем произвольную диаграмму $\Delta \in Conf$, положим $\Delta^* = sig^* \circ sig \circ \Delta$. В силу закона непротиворечия (условие (v) определения 2.3) имеем $\Delta^* \in Conf$, поэтому диаграмма Δ^* обладает копределом, следовательно ввиду предложения 2.5 *SIG*-диаграмма

$sig \circ \Delta^* = sig \circ \Delta$ обладает копределом. В силу предложения 2.2 *sig* сохраняет копределы диаграммы Δ. □

Этот факт позволяет выявить все *c-DESC*-диаграммы, способные служить конфигурациями.

Определение 2.6. *SIG*-диаграмма Θ называется (*sig*-)*предконфигурацией*, если она имеет копредел и функтор *sig* поднимает копределы всех диаграмм из класса $\mathbf{D}sig^{-1}(\{\Theta\}) = \{\Delta \mid sig \circ \Delta = \Theta\}$. Класс всех *c-DESC*-диаграмм, переходящих в предконфигурации под действием функтора **D***sig*, называется *потенциалом комплексируемости* функтора *sig*. □

Предложение 2.7. Тройка *SC* является технологией специфицирования тогда и только тогда, когда выполнены условия (ii)-(iii) определения 2.3 и $Conf = \mathbf{D}sig^{-1}(IC)$ для некоторого класса *sig*-предконфигураций *IC*.

Доказательство. Необходимость вытекает из предложения 2.6, при выборе $IC = sig \circ Conf$. Достаточность проверяется непосредственно. □

Для заданного функтора выделения интерфейсов представляет интерес вопрос, насколько богат его потенциал комплексируемости. Ясно, что его наименьшим возможным значением является пустое множество.

Предложение 2.8. Существуют непустая категория *c-DESC* и функтор *sig* такие, что тройка *SC* удовлетворяет следующему условию: она является технологией специфицирования тогда и только тогда, когда класс *Conf* пуст.

Доказательство. Пусть *S2* – полная подкатегория в **Set**, класс объектов которой состоит из всех двухэлементных множеств. Для произвольного непустого множества *A* обозначим через $S3_A$ подкатегорию в **Set**, состоящую из всех трехэлементных множеств вида $\{a, b, A\}$, $a, b \in A$, $a \neq b$, и всех их отображений *f*, удовлетворяющих условию $f^{-1}(\{A\}) = \{A\}$, так что имеется полное вложение $is_A : S3_A \hookrightarrow S2 : B \mapsto B \setminus \{A\}$. Положим $S3_\varnothing = S2$, $is_\varnothing = 1_{S2}$. Рассмотрим подкатегорию в **Set**

$$S3 = \Big(\bigcup_{A \in \mathrm{Ob}\,\mathbf{Set}} \mathrm{Ob}\, S3_A,$$

$$\bigcup_{A \in \mathrm{Ob}\,\mathbf{Set}} (\mathrm{Mor}\, S3_A \cup \{u : P \to Q \mid P \in \mathrm{Ob}\, S2,\ Q \in \mathrm{Ob}\, S3_A,\ u(P) \not\ni A\})\Big).$$

Пусть $is : S3 \to S2$ – функтор, действующий на каждой подкатегории $S3_A \subseteq S3$ как is_A. Он унивалентен, и функтор, задающий вложение *S2* в *S3* в качестве подкатегории, является левым сопряженным к нему с тождественной единицей, поэтому в силу предложения 2.7 тройка $\langle S3, \varnothing, is\rangle$ является формальной технологией специфицирования. Покажем, что класс всех *is*-предконфигураций пуст. Пусть $\Theta : X \to S2$ – произвольная *S2*-диаграмма, обладающая копределом (в частности, схема *X* непуста, поскольку *S2* не имеет инициального объекта). Пусть

$D = \bigcup_{I \in \mathrm{Ob}\, X} \Theta(I)$, Δ – (единственная) $S3_D$-диаграмма, удовлетворяющая условию $is_D \circ \Delta = \Theta$ (она получается из Θ путем добавления элемента D к каждому множеству $\Theta(I)$, $I \in \mathrm{Ob}\, X$). Пусть $D_2 = \{D, \varnothing\} \in \mathrm{Ob}\, S2$, $\delta : \Theta \to \ulcorner R \urcorner$ – произвольный копредел диаграммы Θ, $i : R \to D_2$ – произвольная биекция. Тогда $\ulcorner i \urcorner \circ \delta$ – копредел диаграммы Θ, однако Δ не имеет копредела, объект которого переходит в D_2 под действием функтора is, поскольку $D_2 \notin is_D(\mathrm{Ob}\, S3_D)$, а для любых $S3$-объектов $P \in \mathrm{Ob}\, S3_D$ и $Q \notin \mathrm{Ob}\, S3_D$ множество $\mathrm{Mor}(P, Q)$ пусто. Поэтому Θ не является предконфигурацией. □

Это утверждение позволяет дать отрицательный ответ на следующий вопрос: должны ли входить в потенциал комплексируемости все точки – тривиальные акты «комплексирования», состоящие во взятии своего результата как заранее построенного? Более общим является вопрос о вхождении в него любого *c-DESC*-коконуса – акта комплексирования, результат которого присутствует в нем в явном виде как выделенный компонент (вершина). Положительный ответ на эти вопросы, характеризующий достаточно «разумно» построенную формальную технологию, тесно связан с категорным понятием подвижности. Многие понятия такого рода определяются в конкретной категории – паре, состоящей из некоторой категории и действующего из нее унивалентного функтора [136, определение 5.1]. Ввиду условия (ii) определения 2.3, пара (*c-DESC*, *sig*), составленная из компонентов формальной технологии, является конкретной категорией. Для краткости мы будем называть категорию моделей конкретной, подразумевая эту пару (если это не будет приводить к путанице). В частности, если тройка *SC* является технологией специфицирования, то согласно определению [136, определение 5.28] категория *c-DESC* называется подвижной (transportable), если для любых *c-DESC*-объекта A и *SIG*-изоморфизма i с областью $sig(A)$ существует *c-DESC*-изоморфизм j с областью A, удовлетворяющий условию $sig(j) = i$. Поскольку $\ulcorner i \urcorner$ – копредел точки $\ulcorner sig(A) \urcorner$, подвижность эквивалентна способности функтора *sig* поднимать копределы любой точки.

Предложение 2.9. Следующие условия эквивалентны в любой формальной технологии специфицирования *SC*:

(i) Любая *SIG*-точка является предконфигурацией.

(ii) Любая *c-DESC*-точка входит в потенциал комплексируемости функтора *sig*.

(iii) Любой *c-DESC*-коконус входит в потенциал комплексируемости функтора *sig*.

(iv) *c-DESC* является подвижной конкретной категорией.

Доказательство. Непосредственно проводится цепочка рассуждений (i) ⇒ (iv) ⇒ (iii) ⇒ (ii) ⇒ (i), с учетом конструкции копредела коконуса, описанной в разд.2.2. □

Покажем, что в подвижной категории проверка способности функтора *sig* поднимать копределы некоторой *c-DESC*-диаграммы сводится к проверке способности сохранять их, которая

часто более проста с технической точки зрения (например известно, что любой функтор, обладающий правым сопряженным, сохраняет все копределы [93, разд.5.5]). В частности, два условия естественности выделения интерфейса относительно комплексирования систем здесь эквивалентны друг другу.

Предложение 2.10. Категория моделей технологии специфицирования *SC* подвижна тогда и только тогда, когда следующие условия эквивалентны для любой диаграммы Δ:

(i) Диаграмма Δ обладает копределом и функтор *sig* сохраняет ее копределы.

(ii) *SIG*-диаграмма $sig \circ \Delta$ обладает копределом и функтор *sig* поднимает копределы диаграммы Δ.

Аналогичная эквивалентность имеет место для пределов диаграммы Δ.

Доказательство. Сначала предположим, что в категории *c-DESC* имеет место эквивалентность (i) $\Leftrightarrow$ (ii) для любой диаграммы. Тогда любая точка $\ulcorner I \urcorner$, $I \in \mathrm{Ob}\, SIG$, является предконфигурацией, поскольку любая *c-DESC*-диаграмма из класса $\mathbf{D}sig^{-1}(\{\ulcorner I \urcorner\}) = \ulcorner sig^{-1}(I) \urcorner$ очевидным образом имеет копредел, сохраняемый функтором *sig*, т.е. удовлетворяет условию (i), поэтому *sig* поднимает ее копределы. В силу предложения 2.9, *c-DESC* подвижна.

Обратно, предположим, что *c-DESC* подвижна, и установим эквивалентность (i) $\Leftrightarrow$ (ii).

(i) $\Rightarrow$ (ii). Пусть Δ – диаграмма, удовлетворяющая условию (i), $\delta : \Delta \to \ulcorner S \urcorner$ – ее копредел. По предположению, $sig \circ \delta$ – копредел диаграммы $sig \circ \Delta$. Поэтому для любого ее копредела $\xi : sig \circ \Delta \to \ulcorner T \urcorner$ существует *SIG*-изоморфизм $i : sig(S) \to T$ такой, что $\xi = \ulcorner i \urcorner \circ (sig \circ \delta)$. Ввиду подвижности существует *c-DESC*-изоморфизм j с областью S, удовлетворяющий условию $sig(j) = i$. Тогда $\ulcorner j \urcorner \circ \delta$ – копредел диаграммы Δ, переходящий в ξ под действием функтора $\mathbf{D}sig$.

(ii) $\Rightarrow$ (i). Вытекает из предложения 2.2. □

Забывающие функторы в категорию **Set**, как правило, порождают подвижные категории. Все содержательные технологии, описываемые в настоящей работе, также удовлетворяют условиям предложения 2.9. В то же время нетрудно построить пример формальной технологии, не удовлетворяющей им (см. разд.4.5).

Теперь рассмотрим функторы, обладающие наиболее богатым потенциалом комплексируемости. Ясно, что таковым является в точности любой функтор, который поднимает копределы всех диаграмм: это условие равносильно тому, что класс всех его предконфигураций совпадает с классом всех диаграмм в его кообласти, обладающих копределом. Поэтому в [174, разд.7.5] технологии, в которых интерфейсы выделяются таким функтором, названы скоординированными. Мы покажем, что их категории моделей эквиваленты так называемым топологическим категориям. Напомним [136, определение 21.1], что функтор $fun : C \to D$ называется топологическим, а пара (C, fun) – топологической категорией над D, если любой *fun*-

структурированный источник (*fun*-structured source) имеет единственное *fun*-инициальное поднятие (*fun*-initial lift). Поясним смысл использованных здесь конструкций. Источником [136, определение 10.1] называется произвольная пара $(A, f_i : A \to A_i)_{i \in I}$, состоящая из объекта A (называемого областью источника) и семейства морфизмов f_i с областью A, индексированного некоторым классом I. (Если I – множество, то источник – это конус, обладающий вершиной A и дискретным основанием со схемой I). Источник в категории D называется *fun*-структурированным [136, определение 17.1], если он имеет вид $(B, f_i : B \to fun(A_i))_{i \in I}$ для некоторого семейства C-объектов A_i. Поднятием такого *fun*-структурированного источника называется любой источник в категории C вида $(A, \overline{f_i} : A \to A_i)_{i \in I}$, удовлетворяющий условиям $fun(A) = B$, $fun(\overline{f_i}) = f_i$, $i \in I$. Такое поднятие называется *fun*-инициальным [136, определение 10.41], если для любых D-объекта B', D-морфизма $h : B' \to B$ и поднятия $(A', \overline{f_i}' : A' \to A_i)_{i \in I}$ *fun*-структурированного источника $(B', f_i \circ h : B' \to fun(A_i))_{i \in I}$ существует единственный C-морфизм $\overline{h} : A' \to A$ такой, что $fun(\overline{h}) = h$ и $\overline{f_i}' = \overline{f_i} \circ \overline{h}$, $i \in I$. Примером топологического функтора (как и следует ожидать) служит забывающий функтор $|\text{–}| : \mathbf{Top} \to \mathbf{Set}$, где **Top** – категория всех топологических пространств и всех непрерывных отображений. Здесь область инициального поднятия структурированного источника $(B, f_i : B \to |A_i|)_{i \in I}$ имеет топологию, порожденную объединением по всем $i \in I$ совокупностей прообразов всех множеств, открытых в A_i, относительно f_i.

Определение 2.7. Формальная технология специфицирования либо проектирования называется *скоординированной*, если функтор *sig* поднимает копределы всех *c-DESC*-диаграмм. □

Предложение 2.11. Пусть класс *Conf* является полным прообразом некоторого класса *SIG*-диаграмм, имеющих копредел, относительно функтора **D***sig*. Тогда следующие утверждения эквивалентны:

(i) Тройка *SC* является скоординированной технологией специфицирования.

(ii) Тройка *SC* удовлетворяет условиям (ii)-(iii) определения 2.3 и любая *SIG*-диаграмма, обладающая копределом, является предконфигурацией.

(iii) Тройка *SC* удовлетворяет условиям (ii)-(iii) определения 2.3 и потенциал комплексируемости функтора *sig* совпадает с классом всех *c-DESC*-диаграмм, обладающих копределом.

(iv) Имеет место разложение $sig = sig' \circ se$, где sig' – некоторый топологический функтор, se – эквивалентность категорий, сюръективная на объектах.

(v) Функтор *sig* унивалентен и любой *sig*-структурированный источник имеет *sig*-инициальное поднятие.

Доказательство. Напомним, что эквивалентностью двух категорий называется функтор из одной в другую, входящий в сопряжение, единица и коединица которого состоят из изоморфизмов [93, разд. 4.4].

(i) $\Rightarrow$ (iv). Определим на классе Ob *c-DESC* отношение эквивалентности, полагая объекты A и A' эквивалентными, если существует *c-DESC*-изоморфизм $i : A \to A'$, удовлетворяющий условию $sig(i) = 1_{sig(A)}$. Пусть *c-DESC*$^\circ$ – полная подкатегория в *c-DESC*, класс объектов которой содержит в точности по одному представителю каждого класса этой эквивалентности, $co : c\text{-}DESC^\circ \hookrightarrow c\text{-}DESC$ – вложение. *c-DESC*$^\circ$ имеет такой же скелет, что и категория *c-DESC*, поэтому эквивалентна ей, причем эквивалентность $se : c\text{-}DESC \to c\text{-}DESC^\circ$ сюръективна на объектах. Положим $sig' = sig \circ co$, так что $sig = sig' \circ se$. Функтор sig' поднимает все копределы, поскольку их поднимают функторы sig и co. Более того, если $\delta : \Delta \to \ulcorner S \urcorner$ и $\theta : \Delta \to \ulcorner T \urcorner$ – копределы произвольной *c-DESC*$^\circ$-диаграммы Δ такие, что $sig' \circ \delta = sig' \circ \theta = \operatorname{colim}(sig' \circ \Delta)$, то стрелка копредела $u : S \to T$ такая, что $\theta = \ulcorner u \urcorner \circ \delta$, представляет собой изоморфизм, удовлетворяющий условию $sig'(u) = 1_{sig'(S)}$, откуда $\delta = \theta$ по построению категории *c-DESC*$^\circ$. Иначе говоря, функтор sig' поднимает все копределы единственным образом (lifts colimits uniquely). Кроме того, он обладает левым сопряженным с естественным изоморфизмом в качестве единицы, поскольку таковыми обладают функторы sig и co (в частности, $se \dashv co$). Поэтому функтор $sig'(-^{\mathrm{op}})^{\mathrm{op}} : (c\text{-}DESC^\circ)^{\mathrm{op}} \to SIG^{\mathrm{op}}$ является топологическим [136, теорема 21.18]. Ввиду топологической двойственности (topological duality) [136, теорема 21.9], таковым является и функтор sig'.

(iv) $\Rightarrow$ (iii). Любой топологический функтор универсален [136, предложение 21.3], обладает левым сопряженным с тождественной единицей [136, предложение 21.12(1)], сохраняет все копределы (поскольку обладает правым сопряженным [136, предложение 21.12(2)]), и поднимает все копределы [136, предложение 21.15]. Любая эквивалентность категорий, сюръективная на объектах, также обладает этими четырьмя свойствами [93, разд.4.4]. Поэтому произвольная *c-DESC*-диаграмма Δ имеет копредел тогда и только тогда, когда диаграмма $sig \circ \Delta$ имеет копредел, т.е. является предконфигурацией.

(iii) $\Rightarrow$ (ii). Пусть Θ – произвольная *SIG*-диаграмма, обладающая копределом, положим $\Sigma = sig^* \circ \Theta$. Поскольку функтор sig^* обладает правым сопряженным, он сохраняет все копределы, следовательно *c-DESC*-диаграмма Σ имеет копредел, так что по предположению *SIG*-диаграмма $sig \circ \Sigma$ является предконфигурацией. Вместе с тем $sig \circ \Sigma = sig \circ sig^* \circ \Theta = \Theta$.

(ii) $\Rightarrow$ (i). Ввиду предложения 2.7 *SC* является технологией специфицирования. Ее скоординированность вытекает из того факта, что функтор *sig* по определению поднимает копределы любой диаграммы, **D***sig*-образ которой не имеет копредела.

(iv) $\Rightarrow$ (v). Пусть $(B, f_i : B \to sig(A_i))_{i \in I}$ – произвольный *sig*-структурированный источник. Его можно рассматривать как *sig*'-структурированный источник, поэтому он обладает (единственным) *sig*'-инициальным поднятием, которое мы обозначим через $(A, \overline{f_i} : A \to se(A_i))_{i \in I}$. Этот источник является *se*-структурированным, а поскольку *se* – эквивалентность, сюръективная на объектах, в категории *c-DESC* существует его поднятие, которое мы обозначим через $(\dot{A}, \dot{f_i} : \dot{A} \to A_i)_{i \in I}$. Ясно, что оно является также поднятием *sig*-структурированного источника $(B, f_i)_{i \in I}$. Проверим, что оно *sig*-инициально. Пусть $(A', \overline{f_i}' : A' \to A_i)_{i \in I}$ – поднятие *sig*-структурированного источника $(B', f_i \circ h : B' \to sig(A_i))_{i \in I}$ для произвольного *SIG*-морфизма $h : B' \to B$. Тогда $(se(A'), se(\overline{f_i}') : se(A') \to se(A_i))_{i \in I}$ – поднятие этого же источника, рассматриваемого как *sig*'-структурированный, поэтому существует (единственный) морфизм $\overline{h} : se(A') \to A$ такой, что $sig'(\overline{h}) = h$ и $se(\overline{f_i}') = \overline{f_i} \circ \overline{h}$, $i \in I$. Поскольку *se* – эквивалентность, существует *c-DESC*-морфизм $\dot{h} : A' \to \dot{A}$ такой, что $se(\dot{h}) = \overline{h}$, так что $se(\overline{f_i}') = se(\dot{f_i} \circ \dot{h})$, откуда $\overline{f_i}' = \dot{f_i} \circ \dot{h}$, $i \in I$, ввиду унивалентности любой эквивалентности категорий. Кроме того, он является единственным морфизмом из A' в $\dot{A}$, удовлетворяющим условию $sig(\dot{h}) = h$, ввиду унивалентности функтора *sig*, установленной в доказательстве импликации (iv) $\Rightarrow$ (ii). Свойство *sig*-инициальности источника $(\dot{A}, \dot{f_i})_{i \in I}$ установлено.

(v) $\Rightarrow$ (iv). Определим категорию *c-DESC*$^\circ$, функторы *co*, *se* и *sig*' так же, как в доказательстве импликации (i) $\Rightarrow$ (iv). Пара (*c-DESC*$^\circ$, *sig*') представляет собой конкретную категорию, обладающую свойством амнестичности (amnesticity) [136, определение 3.27(4)] –любой *c-DESC*$^\circ$-изоморфизм, переходящий в тождественный морфизм под действием функтора *sig*', сам является тождественным морфизмом. Установим, что любой *sig*'-структурированный источник обладает *sig*'-инициальным поднятием: в этом случае функтор *sig*' является топологическим [136, предложение 21.5]. Пусть $(B, f_i : B \to sig'(A_i))_{i \in I}$ – произвольный *sig*'-структурированный источник. Поскольку $sig' = sig \circ co$, его можно рассматривать как *sig*-структурированный источник, поэтому он обладает *sig*-инициальным поднятием, которое мы обозначим через $(A, \overline{f_i} : A \to co(A_i))_{i \in I}$. Поскольку $se \circ co = 1_{c\text{-}DESC^\circ}$, источник $(se(A), se(\overline{f_i}) : se(A) \to A_i)_{i \in I}$ является поднятием *sig*'-структурированного источника $(B, f_i)_{i \in I}$. Проверим, что оно *sig*'-инициально. Пусть $(A', \overline{f_i}' : A' \to A_i)_{i \in I}$ – поднятие *sig*'-структурированного источника $(B', f_i \circ h : B' \to sig'(A_i))_{i \in I}$ для произвольного *SIG*-морфизма $h : B' \to B$. Тогда $(co(A'), co(\overline{f_i}') : co(A') \to co(A_i))_{i \in I}$ – поднятие этого же источника, рассматриваемого как *sig*-

структурированный, поэтому существует (единственный) морфизм $\overline{h} : co(A') \to A$ такой, что $sig(\overline{h}) = h$ и $co(\overline{f_i}') = \overline{f_i} \circ \overline{h}$, $i \in I$. Морфизм $\hat{h} = se(\overline{h}) : A' \to se(A)$ удовлетворяет условию $\overline{f_i}' = se(\overline{f_i}) \circ \hat{h}$, $i \in I$. Кроме того, он является единственным морфизмом из A' в $se(A)$, удовлетворяющим условию $sig'(\hat{h}) = h$, ввиду унивалентности функтора sig'. Свойство sig'-инициальности источника $(se(A), se(\overline{f_i}))_{i \in I}$ установлено. □

Ясно, что категория моделей любой скоординированной технологии подвижна. Она наследует многие «хорошие» свойства категории интерфейсов, например полноту и кополноту (поскольку топологические функторы поднимают пределы и копределы всех диаграмм [136, предложение 21.15]). В ней функтор *sig* имеет, в дополнение к левому сопряженному, правый сопряженный sig_* с тождественной коединицей [136, предложение 21.12(2)] – он сопоставляет любому *SIG*-объекту I область *sig*-инициального поднятия источника $(I, \varnothing)$. Иными словами, любой интерфейс имеет «всеобъемлющую» (антидискретную, indiscrete) реализацию, поддерживающую все его возможности по включению в себя компонентов: для любых *c-DESC*-объекта C и *SIG*-объекта I функтор *sig* задает биекцию множества $\mathrm{Mor}(C, sig_*(I))$ на $\mathrm{Mor}(sig(C), I)$. В структурируемых скоординированных технологиях функтор выделения интерфейсов порождает цепочку из трех сопряжений, в которой среднее имеет тождественную единицу, а остальные – тождественную коединицу:

$$sig^{**} \dashv sig^{*} \dashv sig \dashv sig_{*}.$$

Нетривиальный пример такой технологии будет построен в разд.3.4 настоящей работы для формализации синтеза вычислительных систем.

Теперь рассмотрим два «предельных» случая выбора интерфейсов [65]: когда они совпадают с моделями (с точностью до изоморфизма), так что формальная технология никак не отражает реализацию компонентов, и когда все модели имеют один и тот же интерфейс (образующий сингулярную категорию, изоморфную терминальной категории ***1***).

Предложение 2.12. Если функтор *sig* является изоморфизмом, то *SC* является технологией специфицирования, причем обязательно структурируемой и скоординированной, тогда и только тогда, когда пара $\langle c\text{-}DESC, Conf \rangle$ является технологией конфигурирования.

Доказательство. Любой изоморфизм категорий *c-DESC* и *SIG* удовлетворяет условиям (ii)-(v) определения 2.3 (в частности, обратный к нему является как левым, так и правым сопряженным) и поднимает копределы всех диаграмм. □

Примером применения этого результата служит построение «технологии специфицирования интерфейсов» – тройки $\langle SIG, sig \circ Conf, 1_{SIG} \rangle$, которая ввиду предложения 2.6 действительно является формальной технологией специфицирования, если таковой является *SC*. Как

мы увидим далее (следствие 4.4.1), она описывает дискретную структуру технологии *SC* (в смысле, указанном в начале настоящего раздела).

Предложение 2.13. Если $SIG \cong \mathbf{1}$, то *SC* является технологией специфицирования тогда и только тогда, когда выполняются следующие условия:

(i) *c-DESC* является предпорядком, имеющим наименьший элемент.

(ii) Множество всех объектов любой диаграммы $\Delta \in Conf$ имеет точную верхнюю грань.

(iii) Если $\Delta \in Conf$, то любая *c-DESC*-диаграмма Ξ, обладающая схемой $sch(\Delta)$, является конфигурацией.

При этих условиях технология *SC* является структурируемой тогда и только тогда, когда предпорядок в *c-DESC* задается отношением $\mathrm{Ob}\ c\text{-}DESC \times \mathrm{Ob}\ c\text{-}DESC$, а скоординированной – тогда и только тогда, когда он является полной верхней (эквивалентно, полной нижней) полурешеткой.

Доказательство.

(i). Функтор $!_{c\text{-}DESC} : c\text{-}DESC \to \mathbf{1}$ унивалентен тогда и только тогда, когда из любого *c-DESC*-объекта в любой другой направлено не более одного морфизма, т.е. когда *c-DESC* является предпорядком, рассматриваемым как категория. Левый сопряженный к $!_{c\text{-}DESC}$, если он существует, переводит единственный объект категории $\mathbf{1}$ в наименьший элемент этого предпорядка, единица этого сопряжения тождественна.

(ii). Копределом любой диаграммы предпорядка Δ, если он существует, является точная верхняя грань семейства $\Delta(\mathrm{Ob}\ sch(\Delta))$, и функтор $!_{c\text{-}DESC}$ поднимает его.

(iii). Условие $!_{c\text{-}DESC} \circ \Delta = !_{c\text{-}DESC} \circ \Xi$ эквивалентно условию $sch(\Delta) = sch(\Xi)$.

Критерий структурируемости вытекает из того, что если функтор $!_{c\text{-}DESC}^* : \mathbf{1} \to c\text{-}DESC$ обладает левым сопряженным, то он сохраняет терминальный объект, следовательно в этом случае наименьший элемент предпорядка *c-DESC* является одновременно наибольшим.

Критерий скоординированности вытекает из определения 2.7 и топологической двойственности (topological duality) [136, теорема 21.9]. □

Примером технологии специфицирования с $SIG \cong \mathbf{1}$ служит задание спецификаций множествами аксиом – предложений формального исчисления фиксированной сигнатуры, выступающей в качестве единственного интерфейса (см. например [119]). Предпорядок определяется теоретико-множественным включением теорий, определяемых спецификациями (т.е. замыканий множеств аксиом по отношению выводимости). Наименьшая аксиоматическая спецификация имеет пустое множество аксиом, в ней истинны только тавтологии. Комплексирование сложной спецификации состоит в теоретико-множественном объединении спецификаций всех

компонентов. Оно возможно для любой совокупности компонентов, хотя способно привести к противоречивой спецификации, если в одном из компонентов выводимо предложение, отрицание которого выводимо в другом. Такую технологию можно формально расширить до технологии проектирования, дополнив ее категорией, состоящей из всех спецификаций и всех их изоморфизмов, т.е. не вводя нетривиальных трансформаций. Введем обозначения: σ – сигнатура, $\mathrm{L}(\sigma)$ – язык (множество всех правильно построенных предложений) сигнатуры σ, $\vdash$ – отношение выводимости, $LS_\sigma = (2^{\mathrm{L}(\sigma)}, \{(S, T) \mid S, T \subseteq \mathrm{L}(\sigma), T \vdash S\})$ – предупорядоченное множество всех спецификаций сигнатуры σ, $ddLS_\sigma = \mathrm{Ob}\,((\mathbf{d\text{-}CAT} \cap \mathbf{Cat}) \downarrow LS_\sigma)$ – класс всех дискретных LS_σ-диаграмм. Согласно предложению 2.13, четверка

$$TS_\sigma = \langle LS_\sigma, ddLS_\sigma, \sigma(-) : LS_\sigma \to (\{\sigma\}, \{1_\sigma\}), (\mathrm{Ob}\ LS_\sigma, \mathrm{Iso}\ LS_\sigma)\rangle$$

является формальной технологией проектирования, причем неструктурируемой (для нетривиальных исчислений, когда $\mathrm{L}(\sigma)$ не совпадает с множеством всех тавтологий) и скоординированной. В ней можно выявлять категорными методами метатеоретические свойства спецификаций: например спецификация непротиворечива тогда и только тогда, когда она не является терминальным LS_σ-объектом, полна – тогда и только тогда, когда она непротиворечива и кообласть любого исходящего из нее LS_σ-морфизма изоморфна ей самой либо терминальному объекту, тривиальна (т.е. состоит из тавтологий) – тогда и только тогда, когда она является инициальным объектом, и т.д. Отметим, что ввиду предложения 2.13 формально в качестве класса всех конфигураций технологии разработки аксиоматических спецификаций сигнатуры σ может выступать любой класс LS_σ-диаграмм вида $\mathrm{Ob}\,(SCat \downarrow LS_\sigma)$, где $SCat$ – произвольная подкатегория в **Cat**.

2.4. Распараллеливание

Интерфейсы играют важную роль в решении задачи распараллеливания – разбиения некоторой системы на части, в той или иной степени изолированные друг от друга [65]. Это частный случай задачи выявления архитектуры, поэтому с формальной точки зрения она ставится в классе коконусов определенного вида. Действительно, степень изоляции компонентов можно формально оценить как степень структурного сходства модели системы с копроизведением моделей своих компонентов. Конечно, отличие от копроизведения возможно ввиду взаимовлияния компонентов (синергии), однако на уровне интерфейсов это сходство должно иметь вид изоморфизма, поскольку иначе компоненты нельзя считать образующими разбиение системы. Кроме того, в распараллеливании не должны участвовать «лишние» компоненты, не вносящие никакого вклада в интерфейс системы.

Зафиксируем произвольную формальную технологию специфицирования $SC = \langle c\text{-}DESC, Conf, sig \rangle$.

Определение 2.8. *c-DESC*-коконус $\pi : \Delta \to \ulcorner S \urcorner$ называется (*sig*-) *разбиением* модели *S*, если он удовлетворяет следующим условиям:

(i) Диаграмма Δ дискретна.

(ii) π является подкоконусом копредела некоторой конфигурации.

(iii) $sig \circ \pi$ является копределом.

(iv) Если θ – собственный подкоконус коконуса π, то $sig \circ \theta$ не является копределом.

Ребра разбиения называются его *компонентами*. Технология *SC* называется *поддерживающей* некоторый класс разбиений, если он оптимизируем (в смысле определения 2.2). Разбиение называется *суммой*, если оно является копределом. Технология *SC* называется *поддерживающей параллелизм*, если класс всех сумм оптимизируем. □

Особый интерес при распараллеливании представляет ситуация, когда оно не разрушает внутреннюю структуру компонентов, т.е. когда их вхождения в систему являются вложениями. В этом случае, как мы увидим далее, оно определяется по существу однозначно (с точностью до изоморфизма) на уровне интерфейсов. В теории категорий обобщение вложения определяется при помощи понятия инициального морфизма в конкретной категории [136, определение 8.6] (он не имеет ничего общего с инициальным объектом!). *c-DESC*-морфизм $f : T \to S$ называется инициальным, если для любых *c-DESC*-морфизма $m : X \to S$ и *SIG*-морфизма $k : sig(X) \to sig(T)$ таких, что $sig(f) \circ k = sig(m)$, существует *c-DESC*-морфизм $k^+ : X \to T$ такой, что $sig(k^+) = k$ (так что $f \circ k^+ = m$ ввиду унивалентности функтора *sig*). Иными словами, морфизм *f* является инициальным тогда и только тогда, когда он образует *sig*-инициальное поднятие однокомпонентного *sig*-структурированного источника $(sig(T), sig(f) : sig(T) \to sig(S))$ (см. пояснения перед определением скоординированной технологии). Он также является декартовым морфизмом (cartesian morphism) [174, определение 6.1.12] для объекта *S* и морфизма $sig(f)$. Инициальный морфизм, *sig*-образ которого является мономорфизмом, называется вложением (embedding), он сам является мономорфизмом. Любой изоморфизм является вложением (более того, *c-DESC*-изоморфизмы – это в точности все инициальные *c-DESC*-морфизмы, переходящие в изоморфизмы под действием функтора *sig* [136, предложение 8.14]). Ввиду условия (iii) определения 2.3 вложением также является любой регулярный *c-DESC*-мономорфизм [136, предложение 8.7(3)].

Определение 2.9. *c-DESC*-вложение называется *неразрушающим включением*. Разбиение называется *неразрушающим*, если любое его ребро является неразрушающим включением. □

Предложение 2.14. Любые два неразрушающих разбиения некоторой модели изоморфны тогда и только тогда, когда изоморфны их **D***sig*-образы.

Доказательство. Пусть $\pi = \langle \sigma, !_X \rangle : \Delta \to \ulcorner S \urcorner$, $\pi' = \langle \sigma', !_{X'} \rangle : \Delta' \to \ulcorner S \urcorner$ – неразрушающие разбиения, $\langle \rho, fi \rangle : sig \circ \Delta \to sig \circ \Delta'$ – **D***sig*-изоморфизм такой, что $sig \circ \pi = (sig \circ \pi') \circ \langle \rho, fi \rangle$ (ввиду предложения 2.1 существование такого изоморфизма равносильно тому, что $sig \circ \pi \cong sig \circ \pi'$). Выберем произвольный объект $I \in X$, положим $D = \Delta(I)$, $D' = \Delta'(fi(I))$. Поскольку $sig(\sigma'_I) \circ \rho_I = sig(\sigma_I)$ и *c-DESC*-морфизм σ'_I инициален, существует *c-DESC*-морфизм $\gamma_I : D \to D'$ такой, что $sig(\gamma_I) = \rho_I$. Рассуждая аналогично, строим *c-DESC*-морфизм $\gamma'_I : D' \to D$ такой, что $sig(\gamma'_I) = \rho_I^{-1}$. Ввиду унивалентности функтора *sig* имеем $\gamma_I \circ \gamma'_I = 1_{D'}$ и $\gamma'_I \circ \gamma_I = 1_D$, поэтому пара $\langle \gamma, fi \rangle$ представляет собой **D***c-DESC*-изоморфизм, действующий из Δ в Δ', причем $\pi = \pi' \circ \langle \gamma, fi \rangle$. □

Тривиальное однокомпонентное «распараллеливание» системы фактически сохраняет интерфейсы, поэтому его можно назвать перегрузкой, заимствуя термин из объектно-ориентированного программирования.

Определение 2.10. *c-DESC*-морфизм *o* называется *перегрузкой*, если порожденный им коконус $\ulcorner o \urcorner$ является разбиением. □

Покажем, что при наличии достаточного количества конфигураций среди перегрузок имеются оптимальные. Обозначим через *Ovload* класс всех перегрузок. Заметим, что $sig(Ovload) \subseteq \mathrm{Iso}\ SIG$, поэтому $Ovload \subseteq \mathrm{Epi}\ SIG \bigcap \mathrm{Mono}\ SIG$ ввиду унивалентности функтора *sig* (любой унивалентный функтор отражает как эпиморфизмы [136, предложение 7.44], так и мономорфизмы [136, предложение 7.37(2)]).

Предложение 2.15. Если $Conf \supseteq \{ \llcorner 1_{\mathrm{codom}\ o} \lrcorner \mid o \in Ovload \}$, то *SC* поддерживает перегрузки.

Доказательство. Пусть, как в доказательстве предложения 2.6, ε – коединица сопряжения $sig^* \dashv sig$. Пусть *S* – кообласть некоторой перегрузки; в частности, $sig(S)$ не является инициальным *SIG*-объектом (в противном случае единственное разбиение *S* имеет пустое основание). Имеем $\llcorner \varepsilon_S \lrcorner \in Conf$, поскольку $sig(\varepsilon_S) = sig(1_S)$. Коконус $\ulcorner \varepsilon_S \urcorner$ является инициальным объектом в $\ulcorner Ovload \urcorner \downarrow S$, а $\ulcorner 1_S \urcorner$ – терминальным. В частности, для любой перегрузки $o : T \to S$ имеем $\varepsilon_S \circ sig^*(sig(o)) = o \circ \varepsilon_T$ (в силу определения коединицы), а поскольку $sig(o)$ – изоморфизм, имеется $(\ulcorner Ovload \urcorner \downarrow S)$-морфизм $\ulcorner \varepsilon_T \circ sig^*(sig(o))^{-1} \urcorner : \ulcorner \varepsilon_S \urcorner \to \ulcorner o \urcorner$. □

Предложение 2.16. Следующие утверждения эквивалентны для любой перегрузки *o*.

(i) *o* порождает неразрушающее разбиение.

(ii) *o* порождает сумму.

(iii) *o* является изоморфизмом. □

Предположим, что *SC* является технологией над **Set**. Тогда $|S| = \coprod_{I \in \mathrm{Ob}\, sch(\Delta)} |\Delta(I)|$ для любого разбиения $\pi : \Delta \to \ulcorner S \urcorner$, т.е. основные множества компонентов образуют разбиение (в буквальном смысле этого слова) основного множества системы. Если π не является суммой, то *S* обладает структурой, дополнительной по отношению к структурам всех $\Delta(I)$. Любая перегрузка представляет собой биекцию, поэтому она сводится к обогащению структуры. Если структура позволяет ввести понятие связности (и имеется достаточно дискретных конфигураций), то *SC* поддерживает параллелизм: инициальным объектом в $Sums \downarrow S$, где *Sums* – класс всех сумм, является разбиение на связные компоненты, а терминальным – $\ulcorner 1_S \urcorner$. При этом сопоставление *c-DESC*-объекту множества всех его связных компонент часто представляет собой функцию объектов функтора дискретной структуры $sig^{**} : c\text{-}DESC \to \mathbf{Set}$, сопряженного слева к функтору дискретной реализации sig^*, причем коединица этого сопряжения тождественна. Таким образом, параллелизм оказывается тесно связанным со структурируемостью. Примером служит технология моделирования сценариев *SM*.

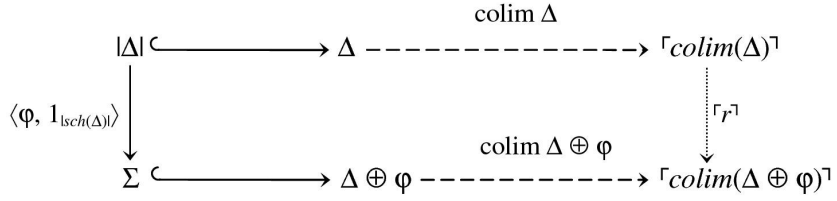
Формально поддерживает параллелизм также технология аксиоматизации спецификаций TS_σ. Действительно, в ней ввиду условия (iv) определения 2.8 сумма не может содержать более одного компонента. В частности, для любой аксиоматической спецификации *S*, не все утверждения которой являются тавтологиями, объекты категории $Sums \downarrow S$ образуют класс $\{\ulcorner i \urcorner \mid i \in \mathrm{Iso}\, LS_\sigma, \mathrm{codom}\, i = S\}$, так что все они попарно изоморфны.

2.5. Трансформации конфигураций

Зафиксируем произвольную формальную технологию проектирования

$$AR = \langle c\text{-}DESC, Conf, sig, r\text{-}DESC \rangle.$$

Условие (viii) определения 2.3 можно наглядно изобразить в виде следующего помеченного графа, отвечающего трансформации конфигураций.

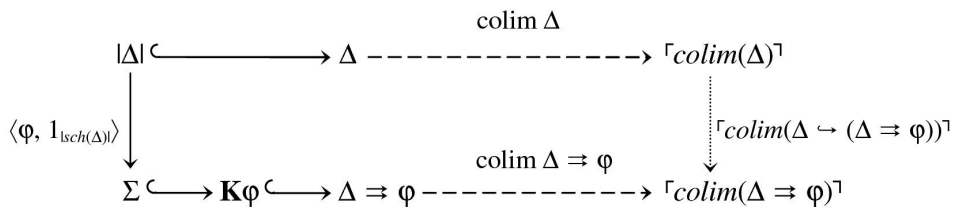


Конечно, этот граф не изображает диаграмму, поскольку его горизонтальные ребра помечены **D***c-DESC*-морфизмами, а вертикальные – **D***r-DESC*-морфизмами (здесь напомним, что условие (vii) определения 2.3 позволяет рассматривать дискретные *c-DESC*-диаграммы как *r-DESC*-диаграммы). Поэтому при анализе трансформаций систем в общем случае не удается применить традиционные теоретико-категорные методы, состоящие в выявлении условий есте-

ственности, универсальности и т.д. Чтобы воспользоваться ими, необходимо перевести все составляющие графа трансформации конфигураций в категорию диаграмм подходящей категории. Простой пример такого рода получается, если Δ и $\Delta \oplus \varphi$ представляют собой точки (тривиальные акты комплексирования). В этом случае копределы по существу являются *c-DESC*-изоморфизмами, а φ – *r-DESC*-морфизмом. В силу условия (vii) определения 2.3 весь граф можно рассматривать как образ *r-DESC*-диаграммы относительно функтора $\ulcorner\!-\!\urcorner$, причем всегда существует единственный *r-DESC*-морфизм *r*, превращающий ее в коммутативную – он совпадает с φ с точностью до изоморфизма.

Более общий способ перевода предложен в [175]: если имеется функтор $r\text{-}sig : r\text{-}DESC \to SIG$ с такой же функцией объектов, что и *sig*, то граф трансформации можно преобразовать в **D***SIG*-диаграмму, действуя на горизонтальные морфизмы функтором **D***sig*, а на вертикальные – функтором **D***r-sig*. Однако при этом удастся рассмотреть только трансформации интерфейсов. Чтобы остаться на уровне моделей, нужны условия, при выполнении которых граф трансформации становится (коммутативной) **D***c-DESC*-диаграммой – «мегамегамоделью». Следуя работе [65], мы рассмотрим два варианта таких условий, которым удовлетворяют формальные технологии, представляющие ряд известных методов инженерии информационных систем.

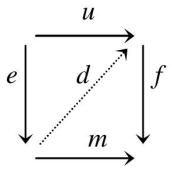
Первый вариант предполагает, что категория *r-DESC* является подкатегорией в *c-DESC*. Это означает, что трансформация любой модели задает ее включение в результат трансформации в качестве компонента. Поэтому чтобы произвести покомпонентную трансформацию системы, порожденной некоторой конфигурацией, трансформации компонентов нужно включить в состав этой конфигурации. Более формально, для диаграммы $\Delta : X \to c\text{-}DESC$ и естественного *c-DESC*-преобразования вида $\varphi : |\Delta| \to \Sigma$ обозначим через $\mathbf{K}\varphi$ *c-DESC*-диаграмму $\mathbf{K}\llcorner\langle\varphi, 1_{|X|}\rangle\lrcorner$ – отрисовку **D***c-DESC*-стрелки, порожденной естественным преобразованием φ. Постоянный функтор $0 : \mathbf{1} \hookrightarrow \mathbf{2}$, рассматриваемый как $(\mathbf{Cat} \downarrow \mathbf{D}c\text{-}DESC)$-морфизм $0 : \ulcorner|\Delta|\urcorner \hookrightarrow \llcorner\langle\varphi, 1_{|X|}\rangle\lrcorner$, порождает каноническое вложение $\mathbf{K}ic(0) : |\Delta| \hookrightarrow \mathbf{K}\varphi$, а функтор $1 : \mathbf{1} \hookrightarrow \mathbf{2}$ – вложение $\mathbf{K}ic(1) : \Sigma \hookrightarrow \mathbf{K}\varphi$. **D***c-DESC*-диаграмма $\iota_\Delta : \Delta \hookleftarrow |\Delta| \hookrightarrow \mathbf{K}\varphi : \mathbf{K}ic(0)$, состоящая из канонических вложений, в силу предложения 2.4 имеет копредел, объект которого мы будем называть *натяжкой* (pull) диаграммы Δ семейством *c-DESC*-морфизмов φ и обозначать $\Delta \rightrightarrows \varphi$. Граф натяжки имеет вид «ежа», полученного из графа диаграммы Δ путем восставления из каждой вершины I стрелки-«иголки» $\varphi_I : \Delta(I) \to \Sigma(I)$. Используя канонические вложения $\Delta \hookrightarrow (\Delta \rightrightarrows \varphi) \hookleftarrow \mathbf{K}\varphi$, представляющие собой ребра копредела, можно определить следующий способ трансформаций конфигураций, гарантирующий коммутативность их графов:



Определение 2.11. Пусть ⟨*c-DESC, Conf, sig*⟩ – формальная технология специфицирования, *hr-DESC* – подкатегория в *c-DESC*, содержащая все изоморфизмы. Если для любых диаграммы $\Delta \in$ *Conf* и естественного *hr-DESC*-преобразования вида $\varphi : |\Delta| \to \Sigma$ выполняются условия $\Delta \rightrightarrows \varphi \in$ *Conf* и *colim*$(\Delta \hookrightarrow (\Delta \rightrightarrows \varphi)) \in$ Mor *hr-DESC*, то четверка ⟨*c-DESC, Conf, sig, hr-DESC*⟩ называется (*трансформационно*) *однородной технологией*. □

Предложение 2.17. Любая однородная технология является формальной технологией проектирования. □

Однородную технологию проектирования можно формально построить по заданной технологии специфицирования, используя конструкции, при помощи которых строятся факторизационные системы в категориях ([136, 159]). Основной конструкцией является отношение Difip (diagonal fill-in property): напомним, что для морфизмов *e*, *f* выполняется Difip(*e*, *f*), если для всякого коммутативного квадрата $f \circ u = m \circ e$ существует единственный «диагональный» морфизм *d* такой, что $d \circ e = u$ и $f \circ d = m$.



Для произвольного класса морфизмов M вводится обозначение $\mathrm{M}^{\uparrow} = \{e \mid \mathrm{Difip}(e, \mathrm{M})\}$.

Предложение 2.18. Пусть ⟨*c-DESC, Conf, sig*⟩ – произвольная формальная технология специфицирования, M – произвольный класс *c-DESC*-морфизмов. Если класс *Conf* замкнут относительно натяжек $\mathrm{M}^{\uparrow}$-морфизмами, то четверка ⟨*c-DESC, Conf, sig*, (Ob *c-DESC*, $\mathrm{M}^{\uparrow}$)⟩ является трансформационно однородной технологией.

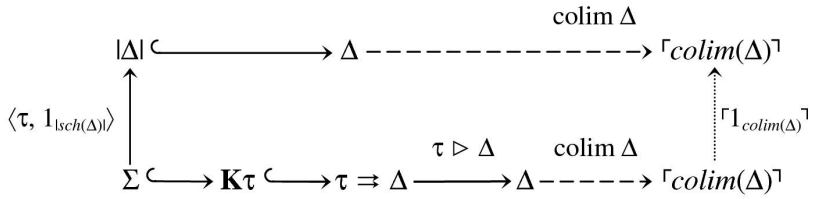
Доказательство. Условие Difip(Iso *c-DESC*, M) выполняется для любого класса M, поэтому Iso *c-DESC* $\subseteq \mathrm{M}^{\uparrow}$. Остальные условия определения 2.11 проверяются непосредственно. □

Имеет место соотношение Iso *c-DESC* = (Mor *c-DESC*)$^{\uparrow}$, определяющее тривиальные трансформации, и Mor *c-DESC* = (Iso *c-DESC*)$^{\uparrow}$, определяющее возможность выбора *r-DESC* = *c-DESC*. Также в скоординированной технологии класс sig^{-1}(Iso *SIG*) совпадает с Init$^{\uparrow}$, где Init – класс всех инициальных морфизмов (это вытекает из предложения 5.1 и наличия у

функтора *sig* правого сопряженного sig_* с тождественной коединицей: непосредственно проверяется, что $sig_*(\mathrm{Mor}\, SIG) \subseteq \mathrm{Init}$, так что для любого $\mathrm{Init}^{\uparrow}$-морфизма $e : X \to Y$ коммутативный квадрат $sig_*(sig(e)) \circ \xi_X = \xi_Y \circ e$, где ξ – единица сопряжения $sig \dashv sig_*$, имеет диагональ, поэтому, действуя на него функтором *sig*, получаем, что $sig(e) \in \mathrm{Iso}\, SIG$); в частности, трансформациями могут служить в точности все перегрузки (обогащения структуры). Обычно однородные технологии возникают в контексте алгебраического подхода к проектированию систем, где и интеграция, и трансформация сводятся к обогащению вычислительных возможностей модели, описываемых множеством термов некоторой алгебры (типа данных). Примером однородной технологии с *r-DESC* = *c-DESC* служит **GAMMA**, рассмотренная в [175] (без явной формулировки свойства однородности). Модель программы в ней представляет собой разновидность машины переписывания термов (term rewriting), а действие по интеграции или трансформации – расширение множества правил переписывания с одновременным применением сигнатурного гомоморфизма типов данных. Еще один пример будет построен в разд.3.4 настоящей работы – это технология синтеза вычислительных систем, в которой класс всех трансформаций совпадает с классом всех перегрузок.

Другой класс формальных технологий, в которых трансформации конфигураций являются **D***c-DESC*-диаграммами, состоит из двойственных к однородным: в них *r-DESC* является подкатегорией в $c\text{-}DESC^{\mathrm{op}}$ (напомним, что последняя получается из *c-DESC* обращением направления всех морфизмов). Отметим, что такая ситуация не противоречит условию (vii) определения 2.3, поскольку двойственный к изоморфизму отождествляется с обратным к нему [136]. Здесь результаты трансформаций входят в исходные модели, так что появляется возможность трассирования – прослеживания трансформаций в обратном порядке в целях выявления назначения отдельных фрагментов моделей. Как мы увидим в разд.4.2, трассирование является ключевым механизмом расширения технологий проектирования приемами аспектно-ориентированного синтеза систем. Покомпонентная трансформация конфигурации системы может оставлять последнюю неизменной: результаты трансформаций уже содержатся в компонентах, а значит и в системе. Действительно, для диаграммы $\Delta : X \to c\text{-}DESC$ и естественного *c-DESC*-преобразования вида $\tau : \Sigma \to |\Delta|$ (двойственность!) обозначим, как выше, через $\mathbf{K}\tau$ *c-DESC*-диаграмму $\mathbf{K}\llcorner\langle\tau, 1_{|X|}\rangle\lrcorner$. Имеются канонические вложения $\mathbf{K}ic(0) : \Sigma \hookrightarrow \mathbf{K}\tau$, $\mathbf{K}ic(1) : |\Delta| \hookrightarrow \mathbf{K}\tau$. Объект копредела **D***c-DESC*-диаграммы $\iota_\Delta : \Delta \hookleftarrow |\Delta| \hookrightarrow \mathbf{K}\tau : \mathbf{K}ic(1)$ будем называть *накачкой* (push) диаграммы Δ семейством τ и обозначать $\tau \rightrightarrows \Delta$. Граф накачки можно представлять как «мишень для стрельбы из лука» – он получается из графа диаграммы Δ путем «попадания» в

каждую вершину «стрелой» из семейства τ. Пусть $\varkappa : \Delta \hookrightarrow (\tau \rightrightarrows \Delta) \hookleftarrow \mathbf{K}\tau : \varkappa'$ – ребра копредела. (**Cat** $\downarrow$ **D***c-DESC*)-морфизм $\langle \tau, !_{\boldsymbol{2}} : \boldsymbol{2} \to \boldsymbol{1} \rangle : \iota\langle \tau, 1_{|X|} \rangle\lrcorner \to \ulcorner |\Delta| \urcorner$ является левым обратным к морфизмам *0* и *1*, откуда, в частности, $1_{|\Delta|} \circ \iota_\Delta = (\iota_\Delta \circ \mathbf{K}ic(\langle \tau, !_{\boldsymbol{2}} \rangle)) \circ \mathbf{K}ic(\boldsymbol{1})$. По определению копредела получаем, что существует **D***c-DESC*-морфизм $\tau \triangleright \Delta : (\tau \rightrightarrows \Delta) \to \Delta$ такой, что $(\tau \triangleright \Delta) \circ \varkappa = 1_{|\Delta|}$ и $(\tau \triangleright \Delta) \circ \varkappa' = \iota_\Delta \circ \mathbf{K}ic(\langle \tau, !_{\boldsymbol{2}} \rangle)$. Непосредственно проверяется, что $\operatorname{colim}(\tau \rightrightarrows \Delta) = (\operatorname{colim} \Delta) \circ (\tau \triangleright \Delta)$, так что любая накачка диаграммы Δ имеет копредел с тем же объектом, что и Δ. Трансформации, двойственные к действиям по интеграции, являются неразрушающими (non-invasive) по отношению к сборке систем: если $\Delta \in \mathit{Conf}$, τ состоит из двойственных к трансформациям и $(\tau \rightrightarrows \Delta) \in \mathit{Conf}$, то условие (viii) определения 2.3 можно обеспечить, взяв накачку в качестве $\Delta \oplus \tau^{\mathrm{op}}$.



Определение 2.12. Пусть $\langle \mathit{c\text{-}DESC}, \mathit{Conf}, \mathit{sig} \rangle$ – формальная технология специфицирования, *hr-DESC* – подкатегория в *c-DESC*, содержащая все изоморфизмы. Если класс *Conf* замкнут относительно накачек *hr-DESC*-морфизмами, то четверка $\langle \mathit{c\text{-}DESC}, \mathit{Conf}, \mathit{sig}, \mathit{hr\text{-}DESC}^{\mathrm{op}} \rangle$ называется (*трансформационно*) *кооднородной технологией*. □

Предложение 2.19. Любая кооднородная технология является формальной технологией проектирования. □

Замкнутыми относительно любых накачек являются: класс всех конечных диаграмм, всех коконусов, всех копроизведений коконусов (в частности, технология моделирования сценариев *SM* кооднородна), а также потенциал комплексируемости функтора *sig* (см. предложение 2.7).

Кооднородность естественным образом возникает при построении технологий над **Set**. Действие трансформации $t^{\mathrm{op}} : X \to Y$, двойственной по отношению к отображению $t : |Y| \to |X|$, можно описать как раскрытие (expansion) точек множества $|X|$ в подмножества множества $|Y|$, образующие его разбиение, с (частичным) переносом структуры объекта X на них. Точку множества $|X|$ можно рассматривать как обозначение класса задач, реализуемого путем раскрытия, в соответствие с интуитивным пониманием трансформации [244]. Например, в разнообразных технологиях моделирования предметных областей посредством графов (таких как онтологии,

схемы рабочих процессов и т.д.) трансформации состоят в углублении предметных знаний, приводящем к замене вершин целыми подграфами.

Теперь рассмотрим технологии проектирования, тривиальные в том смысле, что любая трансформация в них является *c-DESC*-изоморфизмом. Такие технологии могут быть одновременно и однородными, и кооднородными. Более того, легко превратить графы трансформаций конфигураций в коммутативные **D***c-DESC*-диаграммы с ребром $\ulcorner r \urcorner$, удовлетворяющим условиям существования и единственности: нужно потребовать, чтобы класс *Conf* был замкнут относительно изоморфизмов диаграмм. В этом случае для любых конфигурации $\Delta : X \to c\text{-}DESC$ и семейства изоморфизмов $\nu : |\Delta| \to \Sigma \in \mathrm{Mor}\, c\text{-}DESC^{|X|}$ в качестве $\Delta \oplus \nu$ можно взять диаграмму $\Xi : X \to c\text{-}DESC$, заданную соотношением $\Xi(f) = \nu_B \circ \Delta(f) \circ \nu_A^{-1} : \Sigma(A) \to \Sigma(B)$ для любого X-морфизма $f : A \to B$. Действительно, существует **D***c-DESC*-изоморфизм $\langle \nu, 1_X \rangle : \Delta \to \Xi$, поэтому $\Xi \in Conf$ и существует единственный *c-DESC*-изоморфизм $r = colim(\langle \nu, 1_X \rangle) : colim(\Delta) \to colim(\Xi)$, обеспечивающий коммутативность диаграммы трансформации конфигураций. Примером такого рода является технология разработки аксиоматических спецификаций TS_σ из разд.2.3. Критерием замкнутости класса всех конфигураций относительно изоморфизмов служит распространение закона непротиворечия интерфейсов на действия изоморфизмов:

Предложение 2.20. Следующие условия эквивалентны в любой формальной технологии:

(i) Класс *Conf* замкнут относительно изоморфизмов диаграмм.

(ii) Класс $sig \circ Conf$ замкнут относительно изоморфизмов диаграмм.

(iii) Для любой пары *c-DESC*-диаграмм Δ, Ξ, если $sig \circ \Delta \cong sig \circ \Xi$ и $\Delta \in Conf$, то $\Xi \in Conf$.

Доказательство.

(i) $\Rightarrow$ (ii). Выберем диаграммы $\Delta \in Conf$ и $\Theta \cong sig \circ \Delta$. Имеем $sig^* \circ \Theta \cong sig^* \circ sig \circ \Delta \in Conf$, откуда $sig^* \circ \Theta \in Conf$ по предположению, поэтому $\Theta = sig \circ (sig^* \circ \Theta) \in sig \circ Conf$.

(ii) $\Rightarrow$ (iii). Пусть Δ, Ξ – *c-DESC*-диаграммы такие, что $sig \circ \Delta \cong sig \circ \Xi$ и $\Delta \in Conf$. По предположению, существует конфигурация Ξ' такая, что $sig \circ \Xi = sig \circ \Xi'$, откуда $\Xi \in Conf$ ввиду условия (v) определения 2.3.

(iii) $\Rightarrow$ (i). Очевидно. □

В заключение раздела опишем процедуру построения формальной технологии проектирования, в которой основная категория моделей состоит из всех конфигураций заданной технологии *AR*: ее можно использовать при проектировании мегамоделей над *AR*. Пусть *c-Conf* – полная подкатегория в **D***c-DESC* с классом объектов *Conf*, *s-Conf* – полная подкатегория в **D***SIG* с классом объектов $sig \circ Conf$, $dsig : c\text{-}Conf \to s\text{-}Conf$ – функтор, действующий как **D***sig*. Функтор *dsig* способен служить для выделения интерфейсов из мегамоделей, поскольку он унива-

лентен, и ввиду условия (v) определения 2.3 имеет левый сопряженный $dsig^* : s\text{-}Conf \to c\text{-}Conf$, действующий как $\mathbf{D}sig^*$, причем единица этого сопряжения тождественна. Трансформацией произвольной конфигурации $\Delta : X \to c\text{-}DESC$ служит граф, изображенный в начале настоящего раздела, т.е. любая четверка $(\Delta, \varphi, r, \Delta \oplus \varphi)$, удовлетворяющая условию (viii) определения 2.3 (подчеркнем, что $\Delta \oplus \varphi$ и r определяются по заданным Δ и φ, вообще говоря, неоднозначно). Такую четверку правомерно рассматривать как морфизм с областью Δ и кообластью $\Delta \oplus \varphi$ в категории, обладающей классом объектов *Conf*, поскольку четверка $(\Delta, 1_{|X|}, 1_{colim(\Delta)}, \Delta)$ удовлетворяет условию (viii), и кроме того, это условие устойчиво относительно композиции: любые четверки $(\Delta, \varphi, r, \Delta \oplus \varphi)$ и $(\Delta \oplus \varphi, \psi, s, (\Delta \oplus \varphi) \oplus \psi)$ порождают четверку $(\Delta, \omega, s \circ r, (\Delta \oplus \varphi) \oplus \psi)$, где $\omega_I = \psi_I \circ \varphi_I$, $I \in \mathrm{Ob}\, X$. Отметим, что произвольный *c-Conf*-изоморфизм $\langle \nu, fi \rangle : \Delta \to \Sigma$ является трансформацией конфигураций, поскольку он порождает четверку $(\Delta, |\nu|, colim(\langle \nu, fi \rangle), \Sigma)$. Конфигурации мегамоделей выбираются среди *c-Conf*-диаграмм так, чтобы удовлетворить всем условиям определения 2.3 и формализовать исследуемые приемы синтеза мегамоделей.

2.6. Синтез технологий конфигурирования

Специализированные технологии комплексирования разрабатываются в целях повышения эффективности синтеза систем, аналогично тому, как создаются технологии программирования. При этом сложные технологии могут комплексироваться из простых так же, как другие системы. Как указывалось в разд.1.4, перенос основных затрат труда с создания систем на создание средств их создания характерен для разработки, управляемой моделями (MDE). В целях теоретического обоснования такого подхода мы построим формальную технологию проектирования, объектами в которой служат формальные технологии конфигурирования (т.е. все пары вида $\langle c\text{-}DESC, Conf \rangle$, где *Conf* – класс *c-DESC*-диаграмм, имеющих копределы). Ясно, что действием по интеграции $cm : \langle c\text{-}DESC_1, Conf_1 \rangle \to \langle c\text{-}DESC_2, Conf_2 \rangle$ может служить любой функтор $cm : c\text{-}DESC_1 \to c\text{-}DESC_2$, сохраняющий конфигурации (выполняется условие $cm \circ Conf_1 \subseteq Conf_2$) и естественный в традиционном «слабом» смысле относительно комплексирования систем (*cm* сохраняет копределы всех диаграмм из $Conf_1$). Совокупность всех формальных технологий конфигурирования и всех действий по их интеграции (со стандартным законом композиции функторов) образует категорию, которую мы обозначим через **CONF**.

Поскольку действия по интеграции технологий конфигурирования задаются на их категориях моделей, интерфейсом технологии $\langle c\text{-}DESC, Conf \rangle$ служит категория *c-DESC*. Непосредственно проверяется, что отображение $\langle c\text{-}DESC, Conf \rangle \mapsto c\text{-}DESC$ является функцией объектов

функтора, «забывающего» конфигурации, который мы обозначим через ***desc*** : **CONF** → **CAT**. Возможности комплексирования технологий определяются потенциалом комплексируемости функтора ***desc***, который обозначается через *DSConf* и имеет следующий состав.

Предложение 2.21. Диаграмма $\Sigma : X \to \mathbf{CAT}$, имеющая копредел, является ***desc***-предконфигурацией тогда и только тогда, когда любое ребро ее копредела σ_I, $I \in \mathrm{Ob}\, X$, сохраняет копределы всех $\Sigma(I)$-диаграмм, сохраняемые всеми функторами $\Sigma(f)$, где f – X-морфизм с областью I.

Доказательство. Рассмотрим диаграмму $\Sigma : X \to \mathbf{CAT}$ с копределом $\langle \sigma, !_X \rangle : \Sigma \to \ulcorner S \urcorner$. Если она удовлетворяет условию предложения, то любая **CONF**-диаграмма $\Xi \in \mathbf{D}\boldsymbol{desc}^{-1}(\{\Sigma\})$ имеет копредел $\langle \sigma, !_X \rangle : \Xi \to \ulcorner \langle S, \bigcup_{I \in \mathrm{Ob}\, X} \sigma_I \circ \mathit{Conf}_I \rangle \urcorner$, где Conf_I – класс конфигураций технологии $\Xi(I)$. Допустим теперь, что имеются X-объект I и $\Sigma(I)$-диаграмма Δ такие, что все функторы $\Sigma(f)$, где $f \in \mathrm{Mor}\, X$ и $\mathrm{dom}\, f = I$, сохраняют копределы Δ, но σ_I не сохраняет их. Тогда диаграмма

$$\Theta : X \to \mathbf{CONF} : J \mapsto \langle \Sigma(J), \{\Sigma(f) \circ \Delta \mid f : I \to J \in \mathrm{Mor}\, X\} \rangle, h \mapsto \Sigma(h)$$

не имеет копредела вида $\langle \sigma, !_X \rangle : \Sigma \to \ulcorner CF \urcorner$ с ***desc***$(CF) = S$, так что ***desc*** не сохраняет копределы Θ, несмотря на то, что ***desc*** $\circ\, \Theta = \Sigma$. □

Предконфигурациями являются, в частности, все дискретные **CAT**-диаграммы. Приведем пример **CAT**-диаграммы, обладающей копределом, но не являющейся предконфигурацией. Пусть Σ – **CAT**-диаграмма вложений $\boldsymbol{V}^{\mathrm{op}} \hookleftarrow \boldsymbol{V}^{\mathrm{op}} \setminus \{0\} \hookrightarrow \boldsymbol{V}^{\mathrm{op}}$. Дискретная диаграмма $\ulcorner 1 \urcorner \coprod \ulcorner 1 \urcorner$ имеет копредел в $\boldsymbol{V}^{\mathrm{op}}$, однако не имеет его в объекте копредела диаграммы Σ.

Трансформации технологий должны отражать реализацию систем, т.е. переход от технологий конфигурирования интерфейсов к технологиям конфигурирования их реализаций. В силу соображений, приведенных в начале разд.2.3, некоторый функтор *sig* : *c-DESC* → *SIG* можно интерпретировать как правило выделения интерфейсов моделей технологии ⟨*c-DESC*, *Conf*⟩, если он удовлетворяет условиям (ii)-(v) определения 2.3. Его можно рассматривать как **CONF**-морфизм *sig* : ⟨*c-DESC*, *Conf*⟩ → ⟨*SIG*, *sig* ∘ *Conf*⟩, удовлетворяющий этим условиям (ср. пример, приведенный после предложения 2.12). Обозначим через *r-CONF* класс всех таких **CONF**-морфизмов. Ясно, что он замкнут относительно композиций. *Технологией синтеза технологий конфигурирования* называется четверка

$$\boldsymbol{SCONF} = \langle \mathbf{CONF}, \mathit{DSConf}, \boldsymbol{desc}, (\mathrm{Ob}\, \mathbf{CONF}, \mathit{r\text{-}CONF})^{\mathrm{op}} \rangle.$$

Предложение 2.22. ***SCONF*** является неструктурируемой нескоординированной кооднородной технологией.

Доказательство. Очевидно, что функтор ***desc*** унивалентен. Левый сопряженный к нему имеет вид ***desc**** : **CAT** → **CONF** : *c-DESC* ↦ ⟨*c-DESC*, ∅⟩, единица этого сопряжения тожде-

ственна. Однако функтор ***desc**** не имеет левого сопряженного, поскольку он не сохраняет терминальный объект: ***desc****(***1***) не изоморфен терминальному **CONF**-объекту, имеющему вид $\langle \boldsymbol{1}, \mathrm{Ob}\ \mathbf{D}(\boldsymbol{1})\rangle$. Кроме того, как показывает пример, приведенный после предложения 2.21, не все **CAT**-диаграммы, обладающие копределом, являются ***desc***-предконфигурациями. Далее применяем предложение 2.7. Осталось заметить, что в силу предложения 2.12 класс *r-CONF* содержит все **CONF**-изоморфизмы. □

Аналогичным образом можно строить технологии синтеза технологий специфицирования и проектирования. В частности, совокупность всех технологий проектирования можно превратить в категорию, описывающую их интеграцию, если определить морфизмы формальных технологий следующим образом.

Определение 2.13. *Морфизмом формальной технологии проектирования* $\langle c\text{-}DESC_1, Conf_1, sig_1 : c\text{-}DESC_1 \to SIG_1, r\text{-}DESC_1\rangle$ в технологию $\langle c\text{-}DESC_2, Conf_2, sig_2 : c\text{-}DESC_2 \to SIG_2, r\text{-}DESC_2\rangle$ называется тройка функторов

$\langle cm : c\text{-}DESC_1 \to c\text{-}DESC_2, sm : SIG_1 \to SIG_2, rm : r\text{-}DESC_1 \to r\text{-}DESC_2\rangle$,

удовлетворяющая следующим условиям:

(i) $cm \circ Conf_1 \subseteq Conf_2$.

(ii) cm сохраняет копределы всех диаграмм из $Conf_1$.

(iii) $sig_2 \circ cm = sm \circ sig_1$.

(iv) $rm(i) = cm(i)$ для любого $i \in \mathrm{Iso}\ c\text{-}DESC_1$.

Пара функторов $\langle cm, sm\rangle$, удовлетворяющая условиям (i)-(iii), называется *морфизмом формальной технологии специфицирования* $\langle c\text{-}DESC_1, Conf_1, sig_1\rangle$ в $\langle c\text{-}DESC_2, Conf_2, sig_2\rangle$. Функтор cm, удовлетворяющий условию условиям (i)-(ii), называется *морфизмом формальной технологии конфигурирования* $\langle c\text{-}DESC_1, Conf_1\rangle$ в $\langle c\text{-}DESC_2, Conf_2\rangle$. Если все компоненты морфизма технологий являются вложениями подкатегорий, то область морфизма называется *подтехнологией* его кообласти. □

Отметим, что условие (viii) определения 2.3 естественно относительно **ARCH**-морфизмов в следующем смысле. Для любых диаграммы $\Delta \in Conf_1$ и естественного преобразования вида $\varphi : |\Delta| \to \Sigma \in \mathrm{Mor}\ r\text{-}DESC_1^{|sch(\Delta)|}$ трансформации из семейства $rm(\varphi)$ действуют на объекты диаграммы $cm \circ \Delta \in Conf_2$, причем диаграмма $cm \circ (\Delta \oplus \varphi)$ может служить результатом ее трансформации (т.е. выступать в качестве $(cm \circ \Delta) \oplus rm(\varphi)$), а $r\text{-}DESC_2$-морфизм $rm(r)$ – соответствующей трансформацией объекта ее копредела. Действительно, ввиду условия (iv) определения 2.13 имеем $rm(\varphi) : |cm \circ \Delta| \to cm \circ \Sigma$. Диаграмма $cm \circ (\Delta \oplus \varphi)$ содержит поддиаграмму $cm \circ \Sigma$, а поскольку функтор cm сохраняет копределы диаграмм Δ и $\Delta \oplus \varphi$, имеем $rm(r) : colim(cm \circ \Delta) \to colim(cm \circ (\Delta \oplus \varphi))$.

При помощи конструкций определения 2.13 вводятся категории технологий **ARCH** и **SPEC** по аналогии с **CONF**. Имеется цепочка забывающих функторов

$$\mathbf{CAT} \leftarrow \mathbf{CONF} \leftarrow \mathbf{SPEC} \leftarrow \mathbf{ARCH},$$

включающая введенный выше функтор $\boldsymbol{desc} : \mathbf{CONF} \to \mathbf{CAT}$, а также очевидные функторы $\boldsymbol{conf} : \mathbf{SPEC} \to \mathbf{CONF}$, $\boldsymbol{spec} : \mathbf{ARCH} \to \mathbf{SPEC}$. Мы отложим детальное рассмотрение соответствующих технологий проектирования до разд.4.5, поскольку оно требует привлечения понятий, введенных в гл.4: в частности, построение технологии синтеза технологий специфицирования можно описать как формальное преобразование технологии ***SCONF***, отвечающее переходу от модульного проектирования к аспектно-ориентированному. Здесь отметим, что каждая категория технологий имеет инициальный и терминальный объекты (в **ARCH** это четверки $\boldsymbol{triv}(\varnothing)$ и $\langle \boldsymbol{1}, \mathrm{Ob}\,\mathbf{D}(\boldsymbol{1}), 1_{\boldsymbol{1}}, \boldsymbol{1}\rangle$, соответственно). Также имеются суммы технологий, вычисляемые «почленно»:

$$\begin{aligned}\langle c\text{-}DESC_1, Conf_1, sig_1, r\text{-}DESC_1\rangle + \langle c\text{-}DESC_2, Conf_2, sig_2, r\text{-}DESC_2\rangle = \\ \langle c\text{-}DESC_1 \coprod c\text{-}DESC_2, \\ iu_1 \circ Conf_1 \bigcup iu_2 \circ Conf_2, \\ sig_1 + sig_2, \\ r\text{-}DESC_1 \coprod r\text{-}DESC_2\rangle,\end{aligned}$$

где $iu_j : c\text{-}DESC_j \hookrightarrow c\text{-}DESC_1 \coprod c\text{-}DESC_2$, $j = 1, 2$ – инъекции компонентов в раздельное объединение категорий. В частности, технология ***SCONF*** поддерживает параллелизм, поскольку разбиение категории моделей на связные компоненты всегда индуцирует разбиение класса конфигураций (диаграмма, содержащая объекты из разных связных компонент, не может иметь копредела). Вопросы существования пределов и копределов других видов в категориях технологий требуют отдельного исследования, выходящего за рамки настоящей работы.

Приведем пример процедуры синтеза формальных технологий проектирования: сконструируем технологию автоматизированной разработки алгебраических спецификаций и онтологий SPECWARE [252]. «Строительным материалом» нам послужат технологии вида TS_σ, определенные в разд.2.3. Рассмотрим полную подкатегорию в **ARCH**, состоящую из всех таких технологий. Любой морфизм в ней однозначно определяется своей первой компонентой – отображением множеств теорий (спецификаций), согласованным со структурой исчислений, в том числе:

- с построением предложений: любое множество правильно построенных предложений переходит во множество правильно построенных предложений;
- с выводом: теории, связанные отношением выводимости, переходят в связанные отношением выводимости;

- с логической базой: теория, состоящая только из тавтологий, переходит в теорию, состоящую только из тавтологий;
- с расширением: любое объединение теорий переходит в объединение.

Вторая компонента любого морфизма технологий вида TS_σ представляет собой биекцию терминальных категорий, которая определена единственным образом, а третья действует так же, как первая.

Технология SPECWARE не позволяет оперировать произвольными исчислениями. В ней спецификации записываются на языке классического многосортного исчисления первого порядка (расширенного некоторыми конструкциями высшего порядка). Любая его сигнатура представляет собой конечное множество, состоящее из имен сортов и профилей (rank) многосортных операций. Поскольку автоматизированные инструменты могут оперировать только с конечными моделями, выделим в каждой технологии TS_σ с такой сигнатурой подтехнологию, состоящую из всех конечных множеств предложений сигнатуры σ и всех конечных дискретных диаграмм, составленных из них, и обозначим ее через FTS_σ. Среди морфизмов технологий FTS_σ выделим такие, которые индуцируются теоретико-множественными отображениями сигнатур. Они действуют на любое множество предложений путем замены каждого вхождения каждого сигнатурного символа в каждое предложение вхождением образа этого символа. Вышеперечисленные условия согласованности гарантируют, что отображение сигнатур $\phi : \sigma \to \sigma'$ индуцирует морфизм технологий тогда и только тогда, когда оно согласовано с профилями операций в том смысле, что для любой операции $f : S_1, \ldots, S_k \to S_0 \in \sigma$ имеем $\phi(f) : \phi(S_1), \ldots, \phi(S_k) \to \phi(S_0)$ (поэтому его называют сигнатурным морфизмом). Класс всех конечных многосортных сигнатур вместе со всеми сигнатурными морфизмами образуют подкатегорию в **Set**, которую мы обозначим через *MSS*. Положим $FTS = \coprod_{\sigma \in \mathrm{Ob}\, MSS} FTS_\sigma$, $FLS = \boldsymbol{desc}(\boldsymbol{conf}(\boldsymbol{spec}(FTS)))$.

Сигнатурные морфизмы «поднимаются» на уровень спецификаций следующим образом. Обозначим через *SLS* категорию с классом объектов $\mathrm{Ob}\, FLS$, в которой $\mathrm{Mor}(T, T') = \{\phi : \sigma(T) \to \sigma(T') \mid T' \vdash \phi T\}$ для любых *FLS*-объектов T, T'. Стандартный закон композиции отображений действительно делает ее категорией, поскольку если $T' \vdash \phi T$ и $T'' \vdash \vartheta T'$ для произвольных спецификаций T, T', T'' и сигнатурных морфизмов $\phi : \sigma(T) \to \sigma(T')$, $\vartheta : \sigma(T') \to \sigma(T'')$, то $T'' \vdash \vartheta\phi T$. Операция взятия сигнатуры спецификации расширяется до функтора $\sigma(-) : SLS \to MSS$.

Категория *SLS* конечно кополна. Действительно, категория *MSS*, рассматриваемая как подкатегория в **Set**, замкнута относительно копределов конечных диаграмм, поскольку, как легко проверить, кодекартов квадрат, построенный в **Set** над любой парой *MSS*-морфизмов, со-

держится в *MSS*. Копредел любой конечной диаграммы $\Delta : X \to SLS$ имеет вид $\langle \varphi, !_X \rangle : \Delta \to \ulcorner \bigcup_{I \in \mathrm{Ob}\, X} \varphi_I \Delta(I) \urcorner$, где φ – семейство ребер копредела *MSS*-диаграммы $\sigma \circ \Delta$. В частности, функтор $\sigma(-)$ поднимает копределы всех конечных *SLS*-диаграмм.

Трансформацией *SLS*-объекта T в T' служит любая пара *SLS*-морфизмов с общим концом вида $\phi : T \to X \leftarrow T' : \phi'$ такая, что ϕ' задает дефинициальное расширение теории T' (т.е. ϕ' инъективен и для любого символа из $\sigma(X) \setminus \phi'(\sigma(T'))$ в X имеются аксиомы, однозначно определяющие его через символы из $\phi'(\sigma(T'))$). Грубо говоря, трансформация превосходит сигнатурный морфизм возможностью удалить из его кообласти производные символы. Тождественная трансформация – это пара тождественных морфизмов. Композиция трансформаций вычисляется при помощи кодекартова квадрата: если пара $\vartheta : T' \to Y \leftarrow T'' : \vartheta'$ также является трансформацией, то их композиция представляет собой пару $\vartheta''\phi : T \to Z \leftarrow T'' : \phi''\vartheta'$, где $\vartheta'' : X \to Z \leftarrow Y : \phi''$ – ребра кодекартова квадрата $\vartheta''\phi' = \phi''\vartheta$. В [252] доказано, что класс всех дефинициальных расширений замкнут относительно композиций и кодекартовых квадратов, поэтому все *SLS*-объекты и все их трансформации действительно образуют категорию, которую мы обозначим через *r-SLS*.

Обозначим через **FinCat** категорию всех конечных категорий и всех их функторов. Положим

$$\mathrm{SPECWARE} = \langle SLS, \mathrm{Ob}\,(\mathbf{FinCat} \downarrow SLS), \sigma(-) : SLS \to MSS, r\text{-}SLS \rangle.$$

Предложение 2.23. Четверка SPECWARE является неструктурируемой нескоординированной формальной технологией проектирования, содержащей все технологии FTS_σ, $\sigma \in \mathrm{Ob}\, MSS$.

Доказательство. Очевидно, что функтор σ унивалентен. Левый сопряженный к нему, который мы будем как обычно обозначать через σ^*, сопоставляет любой сигнатуре пустую спецификацию этой сигнатуры, единица этого сопряжения тождественна. Далее применяем предложение 2.7. Условие (vii) определения 2.3 вытекает из того, что *r-SLS* содержит *SLS* в качестве (собственной) подкатегории: любому *SLS*-морфизму $\phi : T \to T'$ соответствует трансформация $\phi : T \to T' \leftarrow T' : 1_{T'}$. Выполнение условия (viii) обосновано в [252], где приведена явная процедура построения диаграммы $\Delta \oplus \varphi$ и трансформации объектов копределов r для любой конечной *SLS*-диаграммы Δ и любого семейства трансформаций ее объектов φ.

Проверим, что SPECWARE не структурируема. Пусть $\xi = \{S\}$ – сигнатура, состоящая из единственного имени сорта S, $P = \{\exists z : S\ (\neg z \equiv z)\}$ – противоречивая спецификация сигнатуры ξ. Предположим, что существует функтор σ^{**}, левый сопряженный к σ^* с тождественной коединицей, положим $\zeta = \sigma^{**}(P)$. Множество $\mathrm{Mor}(P, \sigma^*(\zeta))$ пусто при любом ζ, поскольку множество

тавтологий многосортного исчисления непротиворечиво, в то время как множество $\mathrm{Mor}(\zeta, \zeta)$, которое должно быть равномощно ему ввиду определения сопряжения, содержит по крайней мере один морфизм 1_ζ. Полученное противоречие доказывает, что SPECWARE не структурируема.

Проверим, что SPECWARE не скоординирована. Обозначим через X частично упорядоченное множество, состоящее из наименьшего элемента **0** и счетного числа попарно несравнимых элементов x_n, $n = 1, 2, \ldots$. Пусть $\Delta : X \to SLS$ – диаграмма, сопоставляющая элементу **0** пустую спецификацию сигнатуры ξ, а каждому x_n – спецификацию сигнатуры ξ вида $\{\exists z_1 : S \ldots \exists z_n : S \bigwedge_{1 \le i < j \le n} (\neg z_i \equiv z_j)\}$. Диаграмма $\sigma \circ \Delta$ состоит из тождественных морфизмов и очевидно имеет копредел с объектом ξ, однако невозможно построить конечную спецификацию сигнатуры ξ, из которой можно было бы вывести спецификацию $\Delta(x_n)$ для любого $n > 0$. Таким образом, функтор $\sigma(-)$ не поднимает копределы диаграммы Δ. □

Глава 3. АЛГЕБРАИЧЕСКИЕ МЕТОДЫ ПРОЕКТИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

3.1 Отображение алгоритмов на архитектуру вычислительных систем

Функции расчета и анализа данных относятся к числу основных составляющих информационно-управляющих систем: от результатов их проектирования зависит эффективность всей системы. Как указывалось в разд.1.2, в больших информационно-управляющих системах, несмотря на достаточно высокую длительность одного типового цикла обработки данных (10^3–10^6 с), значительное количество источников данных о состоянии объекта (до 10^7 ед.) диктует жесткие требования к производительности средств обработки данных и объему ресурсов для их хранения. Поэтому целесообразно реализовывать компоненты расчета и анализа данных в среде распределенных вычислений (в том числе с динамическим развертыванием на базе технологий Grid [183]), узлы которой обладают разными ограничениями по объему доступных ресурсов памяти и поддерживают разнообразные модели вычислений. В то же время в спецификациях вычислительных задач алгоритмы описываются в терминах абстрактных типов данных (числа, списки и т.д.), образующих бесконечные множества и не предполагающих распределенного доступа. Для обеспечения эффективности необходимо в ходе проектирования алгоритмов выполнять специальные процедуры их отображения на архитектуру вычислительной среды [25]. В их число входят редукция на конечные множества доступных ресурсов, подбор рациональных средств управления потоком вычислений, комплексирование вычислительных компонентов в распределенные системы, оценка и минимизация сложности вычислений. В контексте технологий разработки, управляемой моделями (MDE), отображение на архитектуру представляет собой частный случай перехода от платформенно-независимых моделей (platform-independent model, PIM) к платформенно-зависимым (platform-specific model, PSM) [178]. Чтобы создавать средства, позволяющие автоматически выполнять такой переход в ходе проектирования вычислительных компонентов, необходимо формализовать отображение. Следуя категорному подходу, в настоящей главе мы представим его функтором из категории, описывающей постановку вычислительных задач, в категорию их реализации. Таким образом, сформирована формальная база для применения алгоритмического стиля в проектировании больших информационно-управляющих систем.

Разработка технологий практической реализации процедур отображения, обладающих широкой сферой применимости, отталкиваются от следующих особенностей доступных нам вычислительных средств:

(С1) функционирование компьютера сводится к выполнению арифметических операций над числовыми кодами информационных объектов (самих чисел, текстов и т. д.);

(С2) все допустимые числовые коды образуют конечное множество;

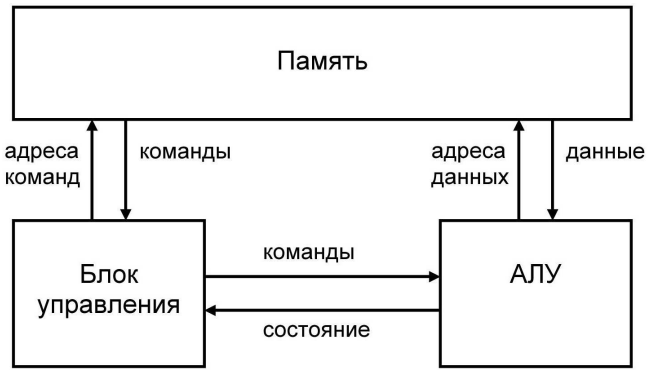
(С3) существуют средства для обнаружения переполнения в ходе выполнения вычислений;

(С4) все возможные элементарные операции, из которых составляются вычисления, образуют конечное множество;

(С5) интеграция вычислительных компонентов состоит в перекодировании значений поддерживаемых ими чисел;

(С6) скорость выполнения вычислений определяется правилами развертывания вычислительного процесса вдоль стрелы реального времени.

Классическая модель архитектуры вычислительного устройства общего назначения, отражающая эти особенности, была построена Дж. фон Нейманом в ходе анализа машины ENIAC (1945 г.). Основными блоками этой архитектуры являются арифметико-логическое устройство (АЛУ), блок управления потоком вычислений, а также память, обеспечивающая единообразный доступ к данным, командам и устройствам ввода-вывода. Взаимосвязи между блоками представлены на следующей схеме [121, с.34].



Результат отображения вычислительного алгоритма на такую архитектуру представляет собой последовательную запись преобразований, выполняемых АЛУ над значениями переменных – ячеек памяти, хранящих числовые данные, при помощи арифметических операций и оператора присваивания, сохраняющего их результаты в переменных. Предполагается, что в АЛУ реализован достаточно большой набор команд, включающий связки классической двузначной логики, обычные арифметические операции над целыми и (округляемыми) вещественными числами, а также средства обработки переменных составных типов (массивов, списков и т.п.). Для повторения некоторой последовательности действий несколько (0 или более) раз в зависи-

мости от значений каких-либо данных, предусматриваются инструкции для блока управления, задающие ветвления, циклы, вызовы процедур и возвраты из них. Эти принципы лежат в основе классического подхода к проектированию вычислительных алгоритмов с использованием псевдокода РАМ-машины (равнодоступная адресная машина, random access machine) – абстрактного фоннеймановского вычислительного устройства [14]. Для моделирования параллельных и многопроцессорных вычислительных устройств используются совокупности взаимодействующих РАМ-машин, классифицируемые по способам интеграции: последовательные каналы связи, общая память и т.д. [26]. Отметим также потоковые (dataflow) устройства, применяющиеся в вычислительных узлах реального времени. Потоковый вычислитель порождает неограниченные последовательности значений конечного набора переменных, индексированные последовательными тактами дискретного времени, на каждом из которых выполняется ограниченное количество вычислительных операций [190].

Для построения формальных технологий отображения алгоритмов на вычислительные среды, обладающие вышеописанными особенностями, мы считаем целесообразным привлечь алгебраический аппарат. Действительно, основным понятием здесь служит модель вычислений – совокупность правил представления чисел и вычислительных операций в рамках заданных ресурсных ограничений. Модель вычислений формально описывается как конечная алгебра, основное множество которой представляет собой совокупность поддерживаемых чисел. Ее сигнатура состоит из вычислительных примитивов, отвечающих (в зависимости от способа реализации модели) аппаратным инструкциям, операторам языка программирования или библиотечным функциям математического пакета программ [42]. В целях управления вычислениями модель обогащается условными операциями и флагом – специальной константой, возвращаемой при выходе значений вычислительных операций за пределы поддерживаемого множества чисел. В настоящей главе рассматриваются примеры моделей вычислений общего назначения, сигнатуры которых заимствуются из арифметики (в частности, обнаружена их связь с многозначной логикой Лукасевича). В проблемно-ориентированных вычислительных узлах, особенно потоковых, используются значительно более сложные примитивы, например преобразование Фурье или алгоритмы расчета потерь электроэнергии в элементах электрической сети (мы приведем пример в разд.5.5).

Для анализа процедур комплексирования вычислительных систем из компонентов, формально описанных моделями вычислений, мы привлекаем аппарат формальных технологий проектирования. Действия по интеграции моделей вычислений, формализующие перекодирование значений поддерживаемых ими чисел, представляют собой подходящим образом обобщенные гомоморфизмы, способные действовать между алгебрами, обладающими различными сигнатурами. Класс всех алгебр, представляющих модели вычислений, и всех таких действий

образует категорию, эквивалентную топологической (над категорией конечных множеств), поэтому в ней существуют все конечные пределы и копределы, разнообразные факторизации и другие конструкции. В частности, любая конечная диаграмма в этой категории имеет копредел, причем его основное множество строится из основных множеств объектов диаграммы по правилам построения копредела в категории множеств. Поэтому любая конечная диаграмма моделей вычислений может выступать в качестве мегамодели вычислений – конфигурации вычислительной системы. Это облегчает реализацию динамического развертывания: топология размещения вычислительных компонентов диктуется ресурсными, а не структурными ограничениями. Класс всех трансформаций в технологии проектирования вычислительных систем совпадает с классом всех перегрузок, поэтому она обладает свойством однородности.

Алгебраический подход к моделированию вычислений создает основу для анализа сложности компьютерных реализаций алгоритмов – результатов их отображения. Поскольку вычислительный процесс в рамках модели вычислений сводится к суперпозиции ее примитивов, все операции, возможность выполнения которых она предоставляет, образуют ее клон – совокупность всех термальных операций (интерпретаций термов ее сигнатуры). Сложность выполнения вычислительного действия в модели оценивается как минимальное необходимое количество обращений к элементам некоторого ее базиса (минимального набора термальных операций, все суперпозиции которых образуют ее клон [42]), причем практический интерес представляют базисы, основу которых составляют вычислительные примитивы моделей. В оценку сложности вычислений в распределенной среде включается также минимальное количество применений морфизмов моделей вычислений, необходимых для обращения ко всем нужным компонентам. Представления потоков вычисления алгоритмов комбинациями термов и морфизмов представляют собой, по существу, абстрактные инварианты их поведения, для которых разработаны методы оценки производительности, не зависящие от выбора целевой аппаратуры [117]. В основе этих методов лежит трактовка базовых конструкций (в данном случае синтаксических элементов алгебраического языка) как обращений к некоторому логическому вычислительному ресурсу. Использование этого ресурса имеет определенную стоимость (меру вычислительной работы), отражающую показатели его эффективности. Оптимизированные инварианты поведения могут автоматически транслироваться в программный код, пригодный к исполнению в вычислительной среде: термы переводятся в команды вычислительных компонентов путем замены базисных операций примитивами, а морфизмы – в акты взаимодействия компонентов. Таким способом удается реализовать подход к разработке вычислительных систем, управляемый моделями.

В заключение раздела отметим, что в настоящее время интенсивно разрабатываются компьютеры, архитектура которых коренным образом отличается от фоннеймановской. К ним

относятся, в частности, квантовые и биологические компьютеры. Одной из их отличительных черт является статистический характер вычислений, соответствующий превращению арифметических операций в случайные функции, заданные на множестве числовых значений. Рассмотрение отображения алгоритмов на такие архитектуры выходит за рамки настоящей работы.

3.2 Частичная интерпретация арифметики

Хорошо известно, что арифметика может быть формализована как теория, т. е. множество предложений языка первого порядка. Поэтому естественным кандидатом на роль математического объекта, способного служить моделью ее реализации, является алгебраическая система – стандартная семантическая конструкция первого порядка. В силу условий (С2) и (С4) из разд.3.1 она должна иметь конечное основное множество и конечную сигнатуру. Отсюда следует, что на ней не могут быть истинными все аксиомы арифметики, поскольку всякая модель последней бесконечна. Машинная реализация арифметики способна верифицировать только такие предложения арифметической теории, которые характеризуют поведение чисел из множества, поддерживаемого компьютером. Поэтому необходим математический метод построения алгебраических систем, на которых эти предложения истинны, и оценки степени их «близости» к стандартной модели арифметики.

В теоретическом программировании алгебраическая система, описывающая поведение информационных объектов определенного типа, называются абстрактным типом данных (АТД) [220]. Числовые АТД обычно строятся *ad hoc*, например, посредством той или иной модификации аксиом арифметики. Однако подобная модификация заставляет заново доказывать основные метатеоремы непротиворечивости, адекватности и т. д. Кроме того, аксиоматические теории, как правило, очень неустойчивы: небольшое изменение аксиом может привести к кардинальному изменению свойств теории. Поэтому следует модифицировать не саму теорию, а способ построения ее модели. Такая модификация теоретико-модельного подхода названа в работе [61] частичной интерпретацией теории первого порядка *T*. Она заключается в построении алгебраической системы, которая должна верифицировать только такие предложения теории *T*, для которых любые входящие в них термы могут быть заменены константами из заданной конечной подсигнатуры ее сигнатуры. Таким образом, формальной моделью ресурса, доступного для представления объектов теории *T*, является явно заданное множество константных символов. Вне этого множества даже стандартные аксиомы равенства могут не выполняться.

Точные определения этих конструкций выглядят следующим образом. Сведения из теории моделей, необходимые для их понимания, можно найти, например, в [56].

Определение 3.1. Пусть σ – сигнатура первого порядка, содержащая символ равенства $\equiv$, T – теория сигнатуры σ, содержащая стандартные аксиомы равенства, σ_0 – конечная подсигнатура сигнатуры σ. *Проекцией* $T \downarrow \sigma_0$ называется множество всех формул φ сигнатуры σ_0 таких, что $T \vdash \varphi$ и для любого терма t, входящего в φ, существует константный символ $c \in \sigma_0$, такой, что $T \vdash t \equiv c$. *Частичной интерпретацией* теории T в сигнатуре σ_0 называется пара $\langle \mathfrak{A}, \sigma_0 \rangle$, где $\mathfrak{A}$ – такая конечная алгебраическая система сигнатуры $\sigma(\mathfrak{A}) \supseteq \sigma_0$, что $\mathfrak{A} \models T \downarrow \sigma_0$. □

Определение 3.2. Пусть $C(\sigma_0) = \{c_1, \ldots, c_n\}$ – множество константных символов конечной подсигнатуры $\sigma_0 \subseteq \sigma$. *Релятивизацией* формулы φ сигнатуры σ на σ_0 называется формула $\varphi \restriction \sigma_0$, получающаяся из φ заменой всех подформул вида $\forall x_0 \psi(x_0, x_1, \ldots, x_k)$ конъюнкциями $\bigwedge_{i=1,\ldots,n} \psi(c_i, x_1, \ldots, x_k)$ и вида $\exists x_0 \psi(x_0, x_1, \ldots, x_k)$ – дизъюнкциями $\bigvee_{i=1,\ldots,n} \psi(c_i, x_1, \ldots, x_k)$. Говорится, что частичная интерпретация $\langle \mathfrak{A}, \sigma_0 \rangle$ *поддерживает* предложение φ сигнатуры σ_0, если $\mathfrak{A} \models \varphi \restriction \sigma_0$. □

Предложение 3.1. Если теория T произвольной сигнатуры σ непротиворечива, то она имеет частичную интерпретацию в любой конечной подсигнатуре сигнатуры σ.

Доказательство. Легко видеть, что отношение

$$c_1 \sim c_2 \Leftrightarrow T \vdash c_1 \equiv c_2$$

является отношением эквивалентности на множестве $C(\sigma_0)$ для произвольной подсигнатуры $\sigma_0 \subseteq \sigma$. Рассмотрим множество $U = (C(\sigma_0)/\sim) \bigcup \{\varnothing\}$, состоящее из его классов эквивалентности и дополнительного элемента, в качестве которого выбрано пустое множество. Построим на нем интерпретацию I символов из σ_0 следующим образом:

(i) если c – произвольный константный символ сигнатуры σ_0, то

$\mathrm{I}(c) = [c]_\sim$;

(ii) если f – произвольный k-местный функциональный символ сигнатуры σ_0, то

$$\mathrm{I}(f)(x_1, \ldots, x_k) = \begin{cases} [c]_\sim, \text{ если существуют } c \in C(\sigma_0) \text{ и } c_i \in x_i,\ i = 1, \ldots, k \\ \qquad\qquad \text{такие, что } (f(c_1, \ldots, c_k) \equiv c) \in T \downarrow \sigma_0 \\ \varnothing \quad \text{иначе} \end{cases}$$

(iii) если P – произвольный k-местный предикатный символ сигнатуры σ_0, то

$\mathrm{I}(P)(x_1, \ldots, x_k) \Leftrightarrow$ существуют $c_i \in x_i$, $i = 1, \ldots, k$ такие, что $P(c_1, \ldots, c_k) \in T \downarrow \sigma_0$.

Если σ_0 конечна, то $\langle \langle U, \mathrm{I} \rangle, \sigma_0 \rangle$ – частичная интерпретация теории T (это проверяется индукцией по построению формул, составляющих проекцию $T \downarrow \sigma_0$). В ней при выходе за пределы $C(\sigma_0)$ все выражения принимают значение «ошибка» или «неопределенность», обозначаемое

пустым множеством. Отметим, что в $\langle U, \mathrm{I}\rangle$ не выполняется аксиома рефлексивности равенства $\forall x(x \equiv x)$ (хотя любая частичная интерпретация поддерживает ее). □

Любая проекция $T \downarrow \sigma_0$ состоит из формул, эквивалентных бескванторным предложениям сигнатуры σ_0. В связи с этим возникает вопрос о структуре классов, состоящих из всех алгебраических систем, аксиоматизируемых заданными наборами бескванторных предложений некоторой фиксированной сигнатуры. Такие классы описывают полиморфизм типов – степень произвола в выборе машинной реализации набора аксиом, характеризующих некоторый АТД.

Ограничимся рассмотрением нормальных алгебраических систем – таких, в которых равенство интерпретируется стандартным образом (как совпадение). Простые теоретико-модельные соображения показывают, что, поскольку любое бескванторное предложение эквивалентно и некоторой Σ_1-формуле, и некоторой Π_1-формуле, любой класс нормальных алгебраических систем, аксиоматизируемый бескванторными предложениями, должен быть замкнут относительно взятия подсистем и расширений. Сформулируем это условие на языке теории категорий. Напомним, что категория всех алгебр сигнатуры σ и всех их гомоморфизмов обозначается через **Alg**(σ).

Определение 3.3. Класс $\mathcal{R}$ нормальных алгебраических систем сигнатуры σ, рассматриваемый как полная подкатегория в **Alg**(σ), называется *полиморфной структурой* сигнатуры σ, если для любого **Alg**(σ)-мономорфизма μ выполняется условие $\mathrm{dom}\,\mu \in \mathcal{R} \Leftrightarrow \mathrm{codom}\,\mu \in \mathcal{R}$. Алгебраическая система $\mathfrak{A}$ называется *образующей* в полиморфной структуре $\mathcal{R}$, если $\mathcal{R}$ состоит из всех алгебраических систем, содержащих $\mathfrak{A}$ в качестве подмодели. Полиморфная структура, обладающая образующей, называется *полиморфным атомом*. □

Основные свойства полиморфных структур кратко описаны в работе [59]. Любой полиморфный атом аксиоматизируется совокупностью всех формул вида φ либо $\neg\varphi$, где φ – атомное предложение, истинных в его образующей. Чтобы алгебраическая система была образующей некоторого полиморфного атома $\wp$, необходимо и достаточно, чтобы любой элемент ее основного множества совпадал со значением некоторого замкнутого терма. Ясно, что образующая является инициальным объектом в $\wp$, т.е. классической формой алгебраической спецификации АТД [50]. Более того, единственный гомоморфизм из образующей в любую систему из $\wp$ является изоморфным вложением, т.е. образующая является изоинициальным объектом в смысле Бертони [155]. В работе Бертони показано, что изоинициальные модели в ряде случаев более адекватно формализуют требования к АТД, чем инициальные. Образующие полиморфных ато-

мов, будучи одновременно инициальными и изоинициальными, являются «идеальными» алгебраическими спецификациями.

Обозначим через $\mathbf{B}_\sigma$ класс всех полиморфных структур сигнатуры σ. Фундаментальная роль полиморфных атомов установлена в следующем утверждении.

Предложение 3.2. Если σ содержит константные символы, то класс $\mathbf{B}_\sigma$ является полной атомной булевой алгеброй относительно теоретико-множественных операций.

Доказательство. Непосредственно проверяется, что класс $\mathbf{B}_\sigma$ замкнут относительно дополнений и произвольных объединений. Атомами в нем являются в точности все полиморфные атомы. □

Из предложения 3.2 вытекает, что любая алгебраическая система $\mathfrak{A}$ содержится в точности в одном полиморфном атоме сигнатуры $\sigma(\mathfrak{A})$, который мы будем обозначать через $\mathbf{B}[\mathfrak{A}]$.

Обратимся к вопросам построения машинной реализации арифметики. Здесь в качестве теории T берется арифметика некоторого множества чисел (натуральных, целых и т.д.), обогащенная достаточным количеством константных символов, представляющих числа. Модели вычислений строятся на базе частичных интерпретаций таких теорий.

Как мы увидим далее (предложение 3.3), в качестве основных множеств машинных арифметик целесообразно использовать начальные отрезки натуральных чисел $E_n = \{0, 1, \ldots, n-1\} \subset \omega$, $n > 0$, где ω – множество неотрицательных целых чисел. Функции (отношения) на них задаются как отображения множества E_n в себя (во множество булевых значений $\{true, false\}$), причем для задания правил отображения используются в том числе стандартные арифметические операции (отношения) над элементами E_n как над целыми числами. В дальнейшем обычные знаки арифметических операций, функций и отношений применяются к элементам E_n именно для этой цели. Подчеркнем, что речь идет только о соглашении об обозначениях, которое само по себе ни в коем случае не наделяет основные множества частичных интерпретаций арифметики структурой числовых множеств.

Рассмотрим подходы к построению машинной арифметики неотрицательных целых чисел. Зафиксируем сигнатуру $\sigma_S = \{\equiv, 0, S\}$, где 0 – константный символ, S – одноместный функциональный символ. Теория, задаваемая следующими аксиомами в дополнение к стандартным аксиомам равенства, содержится в арифметике (см. [56]):

(S1) $\forall x(\neg 0 \equiv Sx)$ (0 является начальным элементом),

(S2) $\forall x \forall y(Sx \equiv Sy \Rightarrow x \equiv y)$ (функция следования S инъективна),

(S3) для любой формулы $\varphi(x_0, x_1, \ldots, x_k)$ сигнатуры σ_S верно

$$\forall x_1 \ldots \forall x_k([\varphi(0, x_1, \ldots, x_k) \wedge \forall x_0(\varphi(x_0, x_1, \ldots, x_k) \Rightarrow \varphi(Sx_0, x_1, \ldots, x_k))] \Rightarrow$$
$$\forall x_0 \varphi(x_0, x_1, \ldots, x_k))$$

(схема аксиом индукции).

Обогащение сигнатуры σ_S константными символами, представляющими натуральные числа, приводит к сигнатуре $\sigma_\omega = \sigma_S \cup \omega$, при этом добавляется следующая схема аксиом:

(S4) для любого $i > 0$ верно $i \equiv S^i 0$, где $S^i = S \ldots S$ (i раз).

Пусть T_ω – теория сигнатуры σ_ω, задаваемая аксиомами (S1)-(S4) и стандартными аксиомами равенства, $\mathfrak{S} = \langle \mathfrak{A}, \sigma_0 \rangle$ – ее произвольная частичная интерпретация. Как вытекает из определений 3.1 и 3.2, частичная интерпретация некоторой теории в общем случае не должна поддерживать все предложения, входящие в эту теорию. Установим, какими свойствами должна обладать $\mathfrak{S}$, чтобы она поддерживала аксиомы (S1)-(S3). В частности, должно выполняться соотношение $\sigma_0 \supseteq \sigma_S$.

Предложение 3.3. Чтобы $\mathfrak{S}$ поддерживала все частные случаи схемы аксиом индукции (S3), необходимо и достаточно, чтобы множество $C(\sigma_0)$ константных символов сигнатуры σ_0 совпадало с E_n для некоторого $n > 0$. Если это условие выполняется, то $\mathfrak{S}$ поддерживает аксиому инъективности (S2) в одном из следующих двух взаимоисключающих случаев:

(i) $\mathfrak{A} \models \neg S(n-1) \equiv i$ для любого $i \in E_n$;

(ii) $\mathfrak{A} \models S(n-1) \equiv 0$.

Аксиома (S1) поддерживается только в случае (i).

Доказательство. Релятивизация схемы индукции имеет вид

$$\bigwedge_{(q_1, \ldots, q_k) \in C(\sigma_0)^k} ([\varphi_0(0, q_1, \ldots, q_k) \wedge \bigwedge_{c \in C(\sigma_0)} (\varphi_0(c, q_1, \ldots, q_k) \Rightarrow \varphi_0(Sc, q_1, \ldots, q_k))] \Rightarrow$$

$$\bigwedge_{c \in C(\sigma_0)} \varphi_0(c, q_1, \ldots, q_k)),$$

где через φ_0 для краткости обозначена релятивизация $\varphi \restriction \sigma_0$. Если $C(\sigma_0) = E_n$ для некоторого n, то эта формула является тавтологией при любой φ. Обратно, пусть $\mathfrak{S}$ поддерживает схему индукции, так что эта формула должна быть истинной на $\mathfrak{A}$ при любой φ. Если $C(\sigma_0) \neq E_n$ ни для какого $n > 0$, то существует константа $q \in \sigma_0$ такая, что $\mathfrak{A} \models \neg q \equiv 0$ и $\mathfrak{A} \models \neg q \equiv Sc$ для любой константы $c \in \sigma_0$. Подставим в релятивизацию схемы индукции формулу $\varphi(x_0) = (\neg q \equiv x_0)$. Антецедент внешней импликации получившейся формулы истинен на $\mathfrak{A}$, однако консеквент ложен (ввиду того, что $(q \equiv q) \in T_\omega \downarrow \sigma_0$), так что вся формула ложна на $\mathfrak{A}$. Получаем противоречие. □

Чтобы частичная интерпретация $\mathfrak{S}$ реализовывала случай (i) предложения 3.3, в качестве основного множества алгебры $\mathfrak{A}$ можно выбрать E_{n+1}, и значение терма $S(n-1)$ положить равным n. Если дополнительно потребовать, чтобы Sn совпадало с n, то процесс последовательного счета, реализованный на базе такой машинной арифметики, продолжится до значения n, на ко-

тором произойдет *переполнение* (overflow), и, следовательно, остановка. Поэтому такие частичные интерпретации называются *арифметиками с переполнением*. При их использовании следует иметь в виду, что они не поддерживают предложение $\forall x \exists y (y \equiv Sx)$, выражающее полную определенность функции следования (хотя оно, разумеется, истинно на $\mathfrak{A}$).

Если же $\mathfrak{S}$ удовлетворяет условию (ii) предложения 3.3, то процесс счета никогда не останавливается – после достижения максимального значения $n - 1$ он заново начинается с нуля. Тем не менее, как будет видно из дальнейшего изложения, выбор E_{n+1} в качестве основного множества оправдан и в этом случае. Значение n играет роль *флага переноса* (carry flag), и хотя оно не может получиться в качестве результата какого-либо арифметического выражения над числами из σ_0, оно позволяет обнаруживать переход через максимальное значение в процессе вычислений. Требуется, чтобы функция следования не различала флаг и нуль, поэтому значение Sn полагается равным 1, и арифметическое равенство интерпретируется отношением сравнимости по $\mathrm{mod}\, n$, отождествляющим эти элементы. Такие частичные интерпретации называются *арифметиками по модулю*.

Операции сложения и умножения определяются через символы сигнатуры σ_S в обоих классах частичных интерпретаций согласно стандартным рекурсивным схемам. В арифметике с переполнением они «обрезаются сверху» значением n, а в арифметике по модулю совпадают с операциями кольца $\mathbf{Z}/n\mathbf{Z}$ (с точностью до отождествления флага переноса с нулем). При этом выполняются стандартные законы ассоциативности, коммутативности и дистрибутивности.

Подробный анализ этих частичных интерпретаций проделан в разд.3.5 ниже, после введения всех необходимых для этого конструкций. Сначала мы рассмотрим арифметику по модулю, ввиду ее широкого распространения, а затем – арифметику с переполнением. Они служат основой для построения мощных вычислительных средств, поддерживающих такие механизмы, как представление чисел в позиционной системе счисления (в т.ч. с неограниченным количеством разрядов [21]), аппроксимацию действительных чисел (например стандарт IEEE 754 [204]) и т.д. Эти механизмы рассмотрены в [68]; их подробный анализ выходит за рамки настоящей работы.

3.3 Полупримальные модели вычислений

Аксиомы, описывающие свойства АТД, традиционно задаются в форме равенств [132]. Существуют соображения в пользу того, что в (многосортной) сигнатуре, состоящей из функциональных символов и предиката равенства, можно описать все практически значимые АТД [41]. В частности, предикаты могут выражаться через равенство. Такое ограничение значитель-

но облегчает требуемый формальный аппарат, позволяя переходить от произвольных алгебраических систем к алгебрам, применять методы универсальной алгебры и дискретной математики.

Пусть $\mathfrak{A}$ – конечная алгебра. Используем обычные обозначения: $|\mathfrak{A}|$ – ее основное множество, $\mathrm{Con}\,\mathfrak{A}$ – совокупность всех ее конгруэнций, $\mathrm{Sub}\,\mathfrak{A}$ – совокупность всех ее подалгебр, рассматриваемая здесь как нижняя полурешетка относительно включения (следуя часто используемому соглашению, считаем, что всегда $\varnothing \in \mathrm{Sub}\,\mathfrak{A}$, см. например [153]). Для любого множества $X \subseteq |\mathfrak{A}|$ через $\mathfrak{A}[X]$ обозначается наименьшая подалгебра $\mathfrak{A}$, содержащая X. Через $\mathrm{Clo}\,\mathfrak{A}$ обозначается семейство всех термальных операций алгебры $\mathfrak{A}$. Оно является клоном – множеством функций на множестве $|\mathfrak{A}|$, замкнутым относительно суперпозиции и содержащим все функции-селекторы $I_k^j : (x_1, \ldots, x_k) \mapsto x_j$, $1 \le j \le k$. Совокупность CL_A всех клонов операций, заданных на конечном множестве A, образует решетку по включению, обладающую коатомами (максимальными клонами), их семейство полностью описано в [133, 242]. Ее единица называется полным клоном и обозначается P_A (этим же символом мы будем для краткости обозначать алгебру $\langle A, \mathrm{P}_A\rangle$). Говорится, что функция $f : A^k \to A$ сохраняет отношение $\rho \subseteq A^m$, если имеется гомоморфизм (**Alg**($\{\rho\}$)-морфизм) $f : \langle A, \rho\rangle^k \to \langle A, \rho\rangle$. Клон, состоящий из всех функций, сохраняющих семейство отношений R, обозначается через $\mathrm{Pol}\,R$; любой клон можно представить как $\mathrm{Pol}\,R$ с подходящим R [242].

Совокупность постоянных функций (констант) из $\mathrm{Clo}\,\mathfrak{A}$, рассматриваемое как подмножество $|\mathfrak{A}|$, называется *центроидом* алгебры $\mathfrak{A}$ (см. [177]) и обозначается через $\mathrm{Cen}\,\mathfrak{A}$. Он содержится в любой непустой подалгебре алгебры $\mathfrak{A}$ и сам является ее подалгеброй – образующей полиморфного атома $\mathbf{B}[\mathfrak{A}]$. Алгебра, обладающая непустым центроидом, называется главной. Центроид называется сильным, если любая функция из $\mathrm{P}_{\mathrm{Cen}\,\mathfrak{A}}$ может быть продолжена до функции из $\mathrm{Clo}\,\mathfrak{A}$, т.е. если клон операций образующей в $\mathbf{B}[\mathfrak{A}]$ полон. Клон $\mathrm{Clo}\,\mathfrak{A}$ называется функционально (или слабо) полным, если клон обогащения алгебры $\mathfrak{A}$ множеством $|\mathfrak{A}|$, элементы которого рассматриваются как константы, совпадает с $\mathrm{P}_{|\mathfrak{A}|}$. Ясно, что с точки зрения компьютерной реализации арифметики только функционально полные клоны представляют интерес. Они гарантируют возможность вычислить любую функцию при наличии возможности порождать все поддерживаемые числа, и в то же время позволяют выявлять структурные свойства реализаций арифметических операций, не зависящие от представления чисел. Достаточным условием функциональной полноты клона является наличие в нем тернарного дискриминатора – функции, определяемой следующим образом:

$$d_3(x, y, z) = \begin{cases} z, & x = y, \\ x & \text{иначе.} \end{cases}$$

Конечные алгебры, клон операций которых содержит d_3, называются *квазипримальными*.

Как показано в работе [60], в задачах формального моделирования вычислительных систем ключевую роль играют *полупримальные* (semi-primal) алгебры. Напомним, что так называются алгебры, удовлетворяющие условию Clo $\mathfrak{A}$ = Pol Sub $\mathfrak{A}$ (см. [177]). Очевидно, что любая полупримальная алгебра квазипримальна, поскольку тернарный дискриминатор сохраняет любое подмножество ее основного множества. Кроме того, центроид любой полупримальной алгебры является сильным и совпадает с пересечением всех ее непустых подалгебр.

Приведем несколько примеров полупримальных алгебр. Если |Sub $\mathfrak{A}$| = 1, то Sub $\mathfrak{A}$ = {∅}, так что $\mathfrak{A}$ – тривиальная (пустая) алгебра. Если |Sub $\mathfrak{A}$| = 2, т.е. Sub $\mathfrak{A}$ = {∅, |$\mathfrak{A}$|}, где |$\mathfrak{A}$| непусто, то Pol Sub $\mathfrak{A}$ = $P_{|\mathfrak{A}|}$ (алгебры, удовлетворяющие этому условию, называются *примальными*). Если |Sub $\mathfrak{A}$| = 3, т.е. Sub $\mathfrak{A}$ = {∅, *X*, |$\mathfrak{A}$|}, где ∅ ⊂ *X* ⊂ |$\mathfrak{A}$|, то клон Pol Sub $\mathfrak{A}$ (а также алгебра ⟨|$\mathfrak{A}$|, Pol Sub $\mathfrak{A}$⟩) обозначается через $C^X_{|\mathfrak{A}|}$; он является коатомом в $CL_{|\mathfrak{A}|}$. Для нашего рассмотрения важную роль играет также случай Sub $\mathfrak{A}$ = $2^{|\mathfrak{A}|}$ (для непустой алгебры $\mathfrak{A}$). Здесь Pol Sub $\mathfrak{A}$ = $C_{|\mathfrak{A}|}$, где через C_A обозначается клон всех функций на множестве *A*, значение которых на каждом наборе аргументов равно одному из них. Этот случай является предельным: клон любой полупримальной алгебры $\mathfrak{A}$ содержит $C_{|\mathfrak{A}|}$. Покажем, что это свойство является характеристическим для полупримальных алгебр и, по существу, определяет их пригодность в качестве формальных моделей вычислительных систем, реализуемых на базе современных цифровых компьютеров [69].

Предложение 3.4. Конечная алгебра $\mathfrak{A}$ является полупримальной тогда и только тогда, когда ее клон содержит $C_{|\mathfrak{A}|}$. □

Доказательство. Известно [109, с.64], что клон любой квазипримальной алгебры состоит из всех функций, сохраняющих все ее подалгебры и все изоморфизмы между ними. Полупримальные алгебры выделяются среди квазипримальных отсутствием нетождественных изоморфизмов между своими неодноэлементными подалгебрами. Поэтому любое обогащение любой полупримальной алгебры дает снова полупримальную алгебру. Следовательно, если клон конечной алгебры $\mathfrak{A}$ содержит $C_{|\mathfrak{A}|}$, то $\mathfrak{A}$ полупримальна. □

Чтобы некоторая конечная алгебра могла служить адекватной формальной моделью машинной арифметики, в ее клоне должны присутствовать функции, отвечающие вычислительным действиям компьютера: арифметическим операциям (редуцированным на ее основное множество), операторам управления потоком вычислений, обращениям к ячейкам памяти и т.д. Сигнатурные функции модели вычислений отвечают вычислительным примитивам, из которых эти действия строятся путем последовательного применения, т.е. суперпозиции. Такой способ моделирования позволяет выполнять формальную оценку сложности вычислений: сложность

вычислительного действия определяется как количество вхождений примитивов в терм, задающий его [42].

В связи с этим требуется строить сигнатуры, представляющие естественные наборы базовых вычислительных примитивов. Напомним основные термины, используемые в задачах поиска сигнатуры, порождающей заданный клон (см. например [42]). Пусть $F \subseteq \mathrm{P}_A$ – произвольное множество функций, заданных на некотором множестве A. Оно называется:

- полным в некотором клоне K, если $\mathrm{Clo}\,\langle A, F\rangle = K$;
- независимым, если для любой функции $f \in F$ множество $F \setminus \{f\}$ не полно в клоне $\mathrm{Clo}\,\langle A, F\rangle$;
- базисом в клоне K, если оно полно в K и независимо;
- предполным в клоне K, если оно не полно в K, но для любой функции $f \in K \setminus \mathrm{Clo}\,\langle A, F\rangle$ множество $F \cup \{f\}$ полно в K.

Функция f называется штрихом Шеффера в клоне K, если множество $\{f\}$ полно в нем.

При построении сигнатур полупримальных алгебр ключевую роль играет функция

$$\mathrm{If}^a(x, y, z) = \begin{cases} y, & x = a, \\ z & \text{иначе}, \end{cases}$$

определенная на произвольном непустом множестве A для произвольного элемента $a \in A$. Для всякого $X \subseteq A$ обозначим $\mathrm{IFS}^X_A = \{\mathrm{If}^a \mid a \in X\}$, $\mathrm{IFS}_A = \mathrm{IFS}^A_A$.

Предложение 3.5.

(i) Множество функций IFS_A полно в C_A.

(ii) Если $|X| = |A| - 1$, то IFS^X_A образует базис в C_A.

(iii) Если $0 < |X| < |A| - 1$, то IFS^X_A образует базис в некотором функционально неполном клоне.

Доказательство. Нумеруя при необходимости множество A числами от 0 до $n = |A| - 1$, будем считать, что оно совпадает с E_{n+1}. Любая функция If^i принадлежит клону C_{n+1}, поскольку ее значение всегда совпадает со значением одного из аргументов. Далее, выберем произвольную k-местную функцию $f \in \mathrm{C}_{n+1}$. Определим отображение $\overline{f}: {E_{n+1}}^k \to \{1, \ldots, k\}$, полагая $\overline{f}(x_1, \ldots, x_k)$ равным индексу первого из таких аргументов из набора $(x_1, \ldots, x_k)$, значение которых равно $f(x_1, \ldots, x_k)$. Имеет место соотношение

$$f(x_1, \ldots, x_k) = \mathrm{If}^0(x_1, \ldots, \mathrm{If}^0(x_k, x_{\overline{f}(0, \ldots, 0, 0)}, \ldots, \mathrm{If}^{n-1}(x_k, x_{\overline{f}(0, \ldots, 0, n-1)}, x_{\overline{f}(0, \ldots, 0, n)})\ldots)\ldots,$$
$$\mathrm{If}^n(x_1, \ldots, \mathrm{If}^0(x_k, x_{\overline{f}(n, \ldots, n, 0)}, \ldots, \mathrm{If}^{n-1}(x_k, x_{\overline{f}(n, \ldots, n, n-1)}, x_{\overline{f}(n, \ldots, n, n)})\ldots)\ldots)).$$

Далее, для всякого $j \in E_{n+1}$ имеем

$$\mathrm{If}^j(x, y, z) = \mathrm{If}^0(x, z, \ldots, \mathrm{If}^{j-1}(x, z, \mathrm{If}^{j+1}(x, z, \ldots, \mathrm{If}^n(x, z, y)\ldots))\ldots).$$

Таким образом, множество IFS^{X}_{n+1}, где $X = E_{n+1} \setminus \{j\}$, полно в C_{n+1}. Его независимость следует из того, что для всякого непустого $Y \subset X$ множество IFS^{Y}_{n+1} сохраняет нетривиальное отношение эквивалентности на E_{n+1}, отождествляющее друг с другом все элементы множества $E_{n+1} \setminus Y$ и попарно различающее все остальные значения. В свою очередь, любой клон, все функции которого сохраняют некоторое нетривиальное отношение эквивалентности, не является функционально полным, поскольку любая постоянная функция сохраняет любое отношение эквивалентности. □

Отметим, что отсюда получается прямое доказательство функциональной полноты клона операций любой полупримальной алгебры. Действительно, произвольную функцию $f : {E_{n+1}}^{k} \to E_{n+1}$ можно представить суперпозицией функций If^{i}, аналогичной приведенной в доказательстве предложения 3.5 с заменой термов $x_{\overline{f(a_1, \ldots, a_k)}}$ константами $f(a_1, \ldots, a_k)$. По аналогии с ДНФ можно назвать такое представление If-*нормальной формой*. Оно также дает элементарное доказательство функциональной полноты клона операций любой квазипримальной алгебры (поскольку $\mathrm{If}^{a}(x, y, z) = d_3(d_3(x, a, y), d_3(x, a, z), z)$ для любых a, x, y, z) и того факта, что центроид любой квазипримальной алгебры является сильным.

Определим на множестве E_{n+1} $(n + 2)$-местную операцию CASE, полагая

$$\mathrm{CASE}(x_0, \ldots, x_{n+1}) = \mathrm{If}^{0}(x_{n+1}, x_0, \mathrm{If}^{1}(x_{n+1}, x_1, \ldots, \mathrm{If}^{n-1}(x_{n+1}, x_{n-1}, x_n)\ldots)).$$

Предложение 3.6. Функция CASE является штрихом Шеффера в C_{n+1}.

Доказательство. Для любого $i = 0, \ldots, n$ имеем $\mathrm{If}^{i}(x, y, z) = \mathrm{CASE}(\ldots, y, \ldots, x)$, где y находится на i-м месте, а на месте каждого из пропущенных аргументов находится z. □

Пусть $\mathfrak{A}$ – произвольная полупримальная алгебра с основным множеством E_{n+1}. Положим

$$F^{S}(x_0, \ldots, x_{n+1}) = \begin{cases} c, & x_0 = \ldots = x_{n+1} = c,\ \{c\} \in S, \\ \max(\{x_0, \ldots, x_{n+1}\} \setminus \{x_{n+1}\}), & |\{x_0, \ldots, x_{n+1}\}| > 1,\ \{x_0, \ldots, x_{n+1}\} \in S, \\ \max(S[\{x_0, \ldots, x_{n+1}\}] \setminus \{x_0, \ldots, x_{n+1}\}) & \text{в остальных случаях,} \end{cases}$$

$$F^{S}_{\mathrm{CASE}}(x_0, \ldots, x_{n+2}) = \begin{cases} F^{S}(x_0, \ldots, x_{n+1}), & x_{n+2} = x_{n+1}, \\ \mathrm{CASE}(x_0, \ldots, x_{n+1}) & \text{иначе.} \end{cases}$$

Предложение 3.7. Функция $F^{\mathfrak{A}}_{\mathrm{CASE}}$ является штрихом Шеффера в $\mathrm{Clo}\,\mathfrak{A}$.

Доказательство. Имеем

$$F^{\mathfrak{A}}(x_0, \ldots, x_{n+1}) = F^{\mathfrak{A}}_{\mathrm{CASE}}(x_0, \ldots, x_{n+1}, x_{n+1}),$$

$$\mathrm{CASE}(x_0, \ldots, x_{n+1}) = F^{\mathfrak{A}}_{\mathrm{CASE}}(x_0, \ldots, x_{n+1}, F^{\mathfrak{A}}(x_0, \ldots, x_{n+1})).$$

Доказываемое утверждение следует из предложений 3.4 и 3.6, поскольку $\mathrm{Sub}\,\langle |\mathfrak{A}|, F^{\mathfrak{A}}_{\mathrm{CASE}} \rangle = \mathrm{Sub}\,\mathfrak{A}$. □

Задачи подбора сигнатур конечных алгебр возникают, в частности, при построении алгебраического представления конечнозначных пропозициональных логик. Такое представление имеет форму логической матрицы [54] – алгебраической системы вида $\mathfrak{M} = \langle V, f_1, \ldots, f_l, \boldsymbol{D}\rangle$, где V – основное множество, $f_1, \ldots, f_l$ – функциональные символы, $\boldsymbol{D}$ – одноместный предикатный символ. Элементам основного множества соответствуют логические значения, а функциональным символам – логические связки (операции), поэтому термы матрицы представляют собой логические формулы. Если логическая формула $f(x_1, \ldots, x_k)$ удовлетворяет условию $\mathfrak{M} \models \forall x_1 \ldots \forall x_k \boldsymbol{D}(f(x_1, \ldots, x_k))$, то она называется тавтологией данной логической матрицы, так что предикат $\boldsymbol{D}$ выделяет подмножество всех логических значений, трактуемых как истинные. Таким образом строятся матричные представления пропозициональных логик – множеств тавтологий в языке, состоящем из имен переменных и связок. Интерес представляют нетривиальные логики, тавтологии которых образуют непустые собственные подмножества множества всех предложений языка.

Для одной и той же логики возможны эквивалентные матричные представления с различными наборами связок такие, что любая связка одного представления выражается через связки другого. Эти матрицы характеризуются тем свойством, что они имеют одинаковые клоны операций. В то же время не всякий клон может быть клоном операций матрицы некоторой нетривиальной логики. Кроме того, в ряде случаев возможно представление логической матрицы в виде алгебры, т.е. задание предиката $\boldsymbol{D}$ равенством. Дадим строгое определение этих свойств и исследуем их выполнимость для полупримальных алгебр.

Определение 3.4. *Матричным обогащением* конечной алгебры $\mathfrak{A}$ называется ее обогащение одноместным предикатным символом $\boldsymbol{D}$ (обозначаемое через $\mathfrak{A}^{\boldsymbol{D}}$) такое, что выполняются следующие условия:

(i) $\mathfrak{A}^{\boldsymbol{D}} \models \exists x \neg \boldsymbol{D}(x)$;

(ii) существует такое $k \geq 0$ и k-местный терм $f \in \mathrm{Clo}\,\mathfrak{A}$, что $\mathfrak{A}^{\boldsymbol{D}} \models \forall x_1 \ldots \forall x_k \boldsymbol{D}(f(x_1, \ldots, x_k))$;

(iii) существуют такие одноместные термы $t, u \in \mathrm{Clo}\,\mathfrak{A}$, что $\mathfrak{A}^{\boldsymbol{D}} \models \forall x(\boldsymbol{D}(x) \Leftrightarrow t(x) = u(x))$. □

Предложение 3.8. Матричное обогащение полупримальной алгебры $\mathfrak{A}$ существует тогда и только тогда, когда существует элемент $a \in |\mathfrak{A}|$ такой, что $\{a\} \notin \mathrm{Sub}\,\mathfrak{A}$.

Доказательство. Необходимость вытекает из того, что если все одноэлементные подмножества $|\mathfrak{A}|$ являются подалгебрами $\mathfrak{A}$, то единственным одноместным термом в $\mathrm{Clo}\,\mathfrak{A}$ является I_1^1. Для доказательства достаточности заметим, что если $\{a\} \notin \mathrm{Sub}\,\mathfrak{A}$ для некоторого

$a \in |\mathfrak{A}|$, то $|\mathfrak{A}[\{a\}]| > 1$. Выберем произвольный элемент $b \in \mathfrak{A}[\{a\}] \setminus \{a\}$, рассмотрим функцию $t : |\mathfrak{A}| \to |\mathfrak{A}| : x \mapsto \mathrm{If}^a(x, b, x)$. Ясно, что $t \in \mathrm{Clo}\,\mathfrak{A}$, и равенство $t(x) = x$ определяет одноместный предикат $\boldsymbol{D}_t$ такой, что: (i) $\boldsymbol{D}_t(a)$ ложно, (ii) $\boldsymbol{D}_t(t(x))$ истинно для любого $x \in |\mathfrak{A}|$. □

Отметим, что условия предложения 3.8 выполняются, если $\mathfrak{A}$ является главной полупримальной алгеброй. Любое ее матричное обогащение определяет такую логику, среди истинностных значений которой присутствуют элементы ее центроида.

Полученные результаты имеют прозрачное содержательное истолкование в контексте моделирования машинной арифметики. Именно, функции If^i и CASE хорошо знакомы программистам – это условные операторы вида **`if x = i then y else z`** и легко реализуемый через них оператор выбора с вариантами. Предложение 3.4 показывает, что C_{n+1} является наименьшим набором операций, содержащим достаточное количество условных операторов, чтобы различать все реализуемые значения числовых данных. Согласно предложению 3.5, этот набор можно построить, используя только операцию выбора значения, явно заданного для каждого возможного значения аргумента. Поэтому C_{n+1} естественно назвать табличной моделью вычислений, это самая бедная модель, позволяющая строить практически полезные вычислительные единицы. На практике к ней прибегают в условиях наличия большого объема памяти и высокой сложности реализуемого алгоритма арифметических вычислений.

Другие модели строятся путем ее обогащения: из предложений 3.4 и 3.5 вытекает, что критерием пригодности алгебры в качестве формальной модели вычислений является ее принадлежность классу полупримальных алгебр. Предложение 3.7 утверждает, что любую такую модель можно построить из единственной операции (правда, достаточно большой арности). Наконец, предложение 3.8 задает критерий наличия возможности использовать технику доказательства конечнозначных логик для верификации вычислительных компонентов. Например, как мы увидим далее, в приложениях возникают матрицы $(n + 1)$-значных логик, обладающих единственным истинностным значением n, которое отвечает арифметическому переполнению. Для обработки появления этого значения (флага) в ходе вычислений применяется условный оператор If^n, реализуемый в любом вычислительном узле.

Отметим, что операции CASE можно придать другое толкование. А именно, она реализует понятие одномерного массива – функции, определенной на множестве индексов (адресов), совпадающем с множеством реализуемых чисел. Видно, что для всякой конечной модели вычислений наличие адресуемой памяти достаточного объема эквивалентно наличию различающего множества условных операторов. Любой из этих признаков характеризует PAM-машину – классическую модель вычислительного устройства общего назначения, исходя из возможностей которого специфицируются и анализируются вычислительные алгоритмы [14].

3.4 Формальная технология проектирования вычислительных систем

Для превращения классов алгебр в категории традиционно привлекается понятие гомоморфизма. Однако для целей моделирования процессов синтеза вычислительных систем оно является слишком узким, т.к. здесь требуется рассматривать в едином контексте алгебры, обладающие различными сигнатурами и вычислительными возможностями. Процедура интеграции вычислительного компонента в систему состоит в кодировании значений чисел, поддерживаемых компонентом – в сопоставлении им значений, поддерживаемых системой (см. условие (C5) из разд.3.1). Формально действиям по интеграции отвечают отображения основных множеств алгебр, задающие правила кодирования так, что структура тех вычислительных операций, которые компонент должен выполнять в составе системы, не разрушается [66]. Например, если при интеграции компонента в систему от него требуется поддержка сложения, то код суммы, вычисленной в компоненте, должен совпадать с суммой кодов слагаемых, вычисленной в системе. При этом для реализации сложения в системе могут применяться примитивы, отсутствующие в компоненте.

Условия поддержки операций задаются очевидным образом при взаимно-однозначном кодировании. Будем говорить, что биекция множеств $\iota : A \leftrightarrow B$ индуцирует вложение некоторого клона X_A в клон Y_B, если для любого $k > 0$ и любой k-местной функции $t \in X_A$ функция $\iota(t) : (x_1, \ldots, x_k) \mapsto \iota t(\iota^{-1}x_1, \ldots, \iota^{-1}x_k)$ принадлежит Y_B. Это означает, что вычислительные возможности клона X_A не превосходят возможностей клона Y_B. Если в дополнение к этому ι^{-1} индуцирует вложение Y_B в X_A, то эти клоны обладают равными возможностями и задают эквивалентные модели вычислений. Поэтому говорится, что такие биекции индуцируют эквивалентность клонов. В категориях алгебр, представляющих синтез вычислительных систем, они должны составлять класс всех изоморфизмов. Подобный подход к оценке вычислительной мощности алгебр проводится в работе [44], где клоны квазипримальных алгебр названы *потенциалами вычислимости*.

Ясно, что структура всех операций сохраняется при кодировании посредством гомоморфизмов. Если $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ – гомоморфизм алгебры $\mathfrak{A}$ в $\mathfrak{B}$, то $\ker\varphi \in \mathrm{Con}\,\mathfrak{A}$ и $\varphi(|\mathfrak{A}|) \in \mathrm{Sub}\,\mathfrak{B}$. Если разложить φ в композицию $\varphi = \mathrm{im}\,\varphi \circ \mathrm{id}\,\varphi \circ \overline{\varphi}$ (где $\mathrm{im}\,\varphi : \varphi(|\mathfrak{A}|) \hookrightarrow |\mathfrak{B}|$ – включение, $\mathrm{id}\,\varphi : |\mathfrak{A}|/\ker\varphi \leftrightarrow \varphi(|\mathfrak{A}|)$ – биекция, $\overline{\varphi} : |\mathfrak{A}| \to |\mathfrak{A}|/\ker\varphi$ – естественный гомоморфизм), то $\mathrm{id}\,\varphi$ индуцирует эквивалентность клонов $\mathrm{Clo}\,\mathfrak{A}/\ker\varphi$ и $\mathrm{Clo}\,\mathfrak{B}\restriction_{\varphi(|\mathfrak{A}|)}$. Чтобы определить произвольное отображение, описывающее интеграцию вычислительных компонентов, мы заменим эти клоны «частичными аналогами» и ослабим условие их эквивалентности до вложения:

Определение 3.5. Пусть $\mathfrak{A}$ и $\mathfrak{B}$ – конечные алгебры. Отображение $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ называется *структурным*, если биекция множеств $\mathrm{id}\,\varphi : |\mathfrak{A}|/\ker\varphi \leftrightarrow \varphi(|\mathfrak{A}|)$ индуцирует вложение клона $\mathrm{Clo}\,\mathfrak{A}/\varphi = (\mathrm{Clo}\,\mathfrak{A} \cap \mathrm{Pol}\,\{\ker\varphi\})/\ker\varphi$ в клон $\mathrm{Clo}\,\mathfrak{B}\restriction_{\varphi} = (\mathrm{Clo}\,\mathfrak{B} \cap \mathrm{Pol}\,\{\varphi(|\mathfrak{A}|)\})\restriction_{\varphi(|\mathfrak{A}|)}$. □

Обозначим через Str класс всех структурных отображений. К сожалению, он не замкнут относительно композиции (рассмотрим композицию $\psi \circ \varphi$, где $\psi : |\mathrm{F}_3| \to |\langle E_2, 0, 1\rangle| : x \mapsto [x/2]$, $\varphi : |\langle E_2, 0, \neg\rangle| \to |\mathrm{F}_3| : x \mapsto 2x$, $\mathrm{F}_3 = \langle E_3, 0, \sim\rangle$, $\neg x = 1 - x$, $\sim x = 2 - x$). Для целей моделирования вычислительных систем требуются подклассы в Str, содержащие, в дополнение к гомоморфизмам, по крайней мере все инъективные структурные отображения – формальные аналоги разнозначного кодирования, обеспечивающего включение компонентов в системы без изменения структуры вычислительных операций [66]. Такие отображения представляют переходы к подсистемам и надсистемам, и в совокупности обобщают конструкцию полиморфной структуры (см. определение 3.3). Введем обозначения: HAlg – класс всех гомоморфизмов конечных алгебр, MStr – класс всех структурных инъекций, **FinAlg** – категория всех конечных алгебр и всех отображений их основных множеств (со стандартным законом композиции отображений). **FinAlg** эквивалентна категории **FinSet** всех конечных множеств и всех их отображений. Все категории алгебр, рассматриваемые в настоящей работе, являются подкатегориями в **FinAlg**. Имеем $\mathrm{HAlg} = \mathrm{Mor}\,\mathbf{FinAlg} \cap \bigcup_{\sigma} \mathrm{Mor}\,\mathbf{Alg}(\sigma)$. В частности, произвольная подалгебра – это не что иное, как инъективный гомоморфизм, т.е. морфизм из класса $\mathrm{HAlg} \cap \mathrm{Mono}\,\mathbf{FinAlg}$.

Определение 3.6. Категория $\mathit{SCA} \subseteq \mathbf{FinAlg}$ называется *структурной*, если $\mathrm{HAlg} \cup \mathrm{MStr} \subseteq \mathrm{Mor}\,\mathit{SCA} \subseteq \mathrm{Str}$. □

Для любой структурной категории *SCA* существует унивалентный «забывающий» функтор $\mathfrak{A} \mapsto |\mathfrak{A}|$ в **FinSet**, обладающий левым сопряженным. Поэтому, в частности, класс всех *SCA*-мономорфизмов совпадает с MStr. Для произвольного конечного множества A скелет его слоя (fibre [136, определение 5.4]), т.е. подкатегории в *SCA*, состоящей из всех алгебр с основным множеством A и всех их тождественных структурных отображений, представляет собой решетку клонов CL_A.

Обозначим через $\mathcal{S}$ совокупность всех структурных категорий, частично упорядоченную включением.

Предложение 3.9. $\mathcal{S}$ является непустой полной нижней полурешеткой.

Доказательство. Покажем, что класс $\mathrm{HAlg} \cup \mathrm{MStr}$ порождает структурную категорию, класс морфизмов которой состоит из всех структурных отображений вида $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ таких, что $\ker\varphi \in \mathrm{Con}\,\mathfrak{A}$ (такое отображение задает кодирование, позволяющее перенести в систему

все вычислительные операции компонента). С одной стороны, каждое такое отображение разлагается в композицию естественного гомоморфизма $\overline{\varphi} : |\mathfrak{A}| \to |\mathfrak{A}|/\ker \varphi$ и структурной инъекции $\mathrm{im}\, \varphi \circ \mathrm{id}\, \varphi$. С другой стороны, пусть отображения $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ и $\psi : |\mathfrak{B}| \to |\mathfrak{C}|$ удовлетворяют условиям $\ker \varphi \in \mathrm{Con}\, \mathfrak{A}$ и $\ker \psi \in \mathrm{Con}\, \mathfrak{B}$. Тогда каждой k-местной функции $t \in \mathrm{Clo}\, \mathfrak{A}$ соответствует хотя бы одна функция $t_\varphi \in \mathrm{Clo}\, \mathfrak{B}$ такая, что $t_\varphi \restriction_{\varphi(|\mathfrak{A}|)} = (\mathrm{id}\, \varphi)(t/\ker \varphi)$. Для любых кортежей $(x_1, \ldots, x_k), (y_1, \ldots, y_k) \in |\mathfrak{A}|^k$, если $\psi\varphi x_i = \psi\varphi y_i$ для всех $i = 1, \ldots, k$, то

$$\psi\varphi t(x_1, \ldots, x_k) = \psi t_\varphi(\varphi x_1, \ldots, \varphi x_k) = \psi t_\varphi(\varphi y_1, \ldots, \varphi y_k) = \psi\varphi t(y_1, \ldots, y_k),$$

поэтому $\ker(\psi \circ \varphi) \in \mathrm{Con}\, \mathfrak{A}$, и кроме того, значения функции $(\mathrm{id}\, \psi)(t_\varphi/\ker \psi)$ на множестве $\psi\varphi(|\mathfrak{A}|)$ не зависят от выбора t_φ и принадлежат $\psi\varphi(|\mathfrak{A}|)$. □

Мы будем обозначать через HS нуль нижней полурешетки $\mathcal{S}$ – категорию, описанную в доказательстве предложения 3.9. Как мы увидим далее (предложение 3.12), $\mathcal{S}$ имеет мощность 2^c и не является решеткой, так что наибольшей структурной категории, которая могла бы формально задать все возможные системные взаимосвязи конечных алгебр, не существует. Тем не менее, имеются структурные категории, способные предоставить наиболее богатые средства для формализации процессов комплексирования вычислительных систем из компонентов. В таких категориях можно оценивать интеграционные возможности моделей вычислений. Согласно результатам разд.3.3, такие категории должны содержать в качестве морфизмов как можно больше структурных отображений полупримальных алгебр. Кроме того, должна быть возможность представить процедуру превращения произвольной конечной алгебры в вычислительный компонент как действие рефлектора – функтора, полностью воспроизводящего ее возможности по интеграции в вычислительные системы.

Определение 3.7. *Подкатегорией вычислительных систем* произвольной структурной категории алгебр *SCA* (обозначается cs(*SCA*)) называется полная подкатегория в *SCA*, класс объектов которой состоит из всех полупримальных алгебр. Категория *SCA* называется *нормальной*, если cs(*SCA*) рефлективна в *SCA*. □

Предложение 3.10. В любой нормальной структурной категории *SCA* рефлектор $\mathrm{cs} : SCA \to \mathrm{cs}(SCA)$ переводит конечную алгебру $\mathfrak{A}$ в полупримальную алгебру, изоморфную алгебре $\langle |\mathfrak{A}|, \mathrm{Pol\ Sub}\, \mathfrak{A} \rangle$; единица рефлексии является биекцией множеств.

Доказательство. Пусть $\eta_\mathfrak{A} : |\mathfrak{A}| \to |\mathrm{cs}(\mathfrak{A})|$ – компонента единицы рефлексии, $\mathfrak{A}'$ – полупримальная алгебра $\langle |\mathfrak{A}|, \mathrm{Pol\ Sub}\, \mathfrak{A} \rangle$, $\iota : |\mathfrak{A}| \leftrightarrow |\mathfrak{A}'| \in \mathrm{MStr}$ – тождественное отображение множества $|\mathfrak{A}|$ на себя, $\nu : |\mathrm{cs}(\mathfrak{A})| \to |\mathfrak{A}'|$ – (единственный) cs(*SCA*)-морфизм, удовлетворяющий условию $\iota = \nu \circ \eta_\mathfrak{A}$. Поскольку ι – мономорфизм, η состоит из мономорфизмов. Но тогда $\eta_\mathfrak{A}$ – биек-

ция множеств (см. [136, предложение 16.3]), поэтому Sub cs($\mathfrak{A}$) $\subseteq \eta_{\mathfrak{A}}$(Sub $\mathfrak{A}$), так что ν является *SCA*-изоморфизмом. □

Предложение 3.11. Если *SCA* – нормальная структурная категория, то функтор $|\text{–}| : \mathrm{cs}(SCA) \to$ **FinSet** имеет как левый сопряженный с тождественной единицей, так и правый сопряженный с тождественной коединицей.

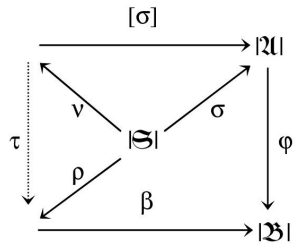
Доказательство. Любое отображение конечных множеств $\varphi : A \to B$ порождает HS-морфизм $\varphi : |\mathfrak{A}| \to |\mathrm{P}_B|$, где $\mathfrak{A} = \langle A, \mathrm{Pol}\,\{\ker \varphi\}\rangle$. Имеем Sub $\mathfrak{A} = \{\varnothing, A\}$ (поскольку все константы сохраняют отношение эквивалентности $\ker \varphi$), откуда в силу предложения 3.10 $\mathrm{cs}(\mathfrak{A}) \cong \mathrm{P}_A$, так что имеется *SCA*-морфизм $\varphi : |\mathrm{P}_A| \to |\mathrm{P}_B|$. Поэтому отображение, сопоставляющее множеству A алгебру P_A, задает полное вложение **FinSet** в *SCA*, композиция которого с рефлектором cs является искомым правым сопряженным. Левый сопряженный строится путем композиции функтора, левого сопряженного к функтору $|\text{–}| : SCA \to$ **FinSet**, с cs. □

Теорема 3.1. Существует нормальная структурная категория, содержащая подкатегорию вычислительных систем любой структурной категории. □

Доказательство теоремы основано на свойствах следующих конструкций.

Определение 3.8. Отображение $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ основных множеств конечных алгебр называется *субинъективным*, если для любого множества $S \subseteq |\mathfrak{A}|$ такого, что ограничение отображения φ на S инъективно, выполняется условие $\varphi(\mathfrak{A}[S]) \subseteq \mathfrak{B}[\varphi S]$. Если это условие выполняется для любого $S \subseteq |\mathfrak{A}|$, то φ называется *субнепрерывным*. □

Свойства субинъективности и субнепрерывности можно сформулировать на языке теории категорий в категории **FinAlg**. А именно, подмножество $S \subseteq |\mathfrak{A}|$ можно задать **FinAlg**-мономорфизмом вида $\sigma : |\mathfrak{S}| \to |\mathfrak{A}|$ для некоторой алгебры $\mathfrak{S}$ такой, что $|\mathfrak{S}| = S$. Подалгебра в $\mathfrak{A}$, содержащая S – это мономорфный гомоморфизм α такой, что существует **FinAlg**-морфизм ν, удовлетворяющий условию $\sigma = \alpha \circ \nu$. Подалгебра $\mathfrak{A}[S]$, которую мы будем обозначать через $[\sigma]$, выделяется среди таких мономорфных гомоморфизмов тем, что для любого разложения $\sigma = \alpha' \circ \nu'$, где α' – мономорфный гомоморфизм, существует **FinAlg**-морфизм θ, удовлетворяющий условию $[\sigma] = \alpha' \circ \theta$. Соответственно, **FinAlg**-морфизм $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ субнепрерывен, если для любого равенства вида $\varphi \circ \sigma = \beta \circ \rho$, где σ – мономорфизм и β – подалгебра, существует **FinAlg**-морфизм τ такой, что $\varphi \circ [\sigma] = \beta \circ \tau$ (такой морфизм τ единствен, поскольку $\beta \in$ Mono **FinAlg**). Морфизм φ субинъективен, если такой τ существует для равенств вышеуказанного вида, удовлетворяющих условию $\varphi \circ \sigma \in$ Mono **FinAlg**. Отсюда видно, что субинъективность отображения означает возможность продолжить алгебраическую структуру образов подмножеств его области, на которых оно действует разнозначно, на образы порожденных ими подалгебр.



Лемма 3.1. Композиция субинъективных отображений субинъективна. □

Лемма 3.2. Любой HS-морфизм субнепрерывен.

Доказательство. Докажем, что отображение $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ субнепрерывно тогда и только тогда, когда $\varphi^{-1}(\mathrm{Sub}\,\mathfrak{B}) \subseteq \mathrm{Sub}\,\mathfrak{A}$ (это оправдывает «топологическую» терминологию). С одной стороны, если φ субнепрерывно и $B \in \mathrm{Sub}\,\mathfrak{B}$, то, полагая $A = \varphi^{-1}(B)$, имеем $\varphi(\mathfrak{A}[A]) \subseteq \mathfrak{B}[B] = B$, откуда $\mathfrak{A}[A] \subseteq A$. А поскольку всегда $\mathfrak{A}[A] \supseteq A$, получаем, что $A = \mathfrak{A}[A] \in \mathrm{Sub}\,\mathfrak{A}$. С другой стороны, если $\varphi^{-1}(\mathrm{Sub}\,\mathfrak{B}) \subseteq \mathrm{Sub}\,\mathfrak{A}$ и $S \subseteq |\mathfrak{A}|$, то, полагая $T = \varphi^{-1}(\mathfrak{B}[\varphi S])$, имеем $T \in \mathrm{Sub}\,\mathfrak{A}$, а поскольку $\mathfrak{B}[\varphi S] \supseteq \varphi S$, $T \supseteq S$. Отсюда вытекает, что $\mathfrak{A}[S] \subseteq T$, откуда $\varphi(\mathfrak{A}[S]) \subseteq \mathfrak{B}[\varphi S]$.

Утверждение леммы вытекает из того, что доказанному выше критерию субнепрерывности удовлетворяют и все гомоморфизмы, и все структурные инъекции. □

Введем обозначения: CS – категория, состоящая из всех полупримальных алгебр и всех субинъективных отображений их основных множеств; CM – категория, порожденная классом Mor HS ∪ Mor CS. Покажем, что CM удовлетворяет условию теоремы 3.1 (и тем самым является наименьшей среди категорий, удовлетворяющих этому условию).

Лемма 3.3. MStr ∘ Str = Str. □

Лемма 3.4. Для любых конечной алгебры $\mathfrak{A}$ и полупримальной алгебры $\mathfrak{B}$ любое субинъективное отображение $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ является структурным.

Доказательство. Предположим противное: пусть для некоторой k-местной функции $t \in \mathrm{Clo}\,\mathfrak{A}/\varphi$, множества $B \in \mathrm{Sub}\,\mathfrak{B}\restriction_\varphi$ и кортежа $(x_1, \ldots, x_k) \in B^k$ выполняется условие $(\mathrm{id}\,\varphi)(t)(x_1, \ldots, x_k) \in \varphi(|\mathfrak{A}|) \setminus B$. Выберем какое-нибудь отображение $\psi : \varphi(|\mathfrak{A}|) \to |\mathfrak{A}|$ такое, что $\varphi\psi x = x$ для любого $x \in \varphi(|\mathfrak{A}|)$. Существует такая k-местная функция $t' \in \mathrm{Clo}\,\mathfrak{A}$, что $t'(\psi x_1, \ldots, \psi x_k) \notin \varphi^{-1}(B)$, следовательно, $\varphi(\mathfrak{A}[S]) \not\subseteq B$, где $S = \{\psi x_1, \ldots, \psi x_k\}$. Но это противоречит определению 3.8, поскольку $\mathfrak{B}[\varphi S] \subseteq B$. □

Лемма 3.5. Для любых полупримальных алгебр $\mathfrak{A}$ и $\mathfrak{B}$ отображение $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ является субинъективным тогда и только тогда, когда $\{\varphi\} \circ \mathrm{Mor\,HS} \subseteq \mathrm{Str}$.

Доказательство. Необходимость следует из лемм 3.1, 3.2 и 3.4. Для доказательства достаточности предположим, что для некоторого множества $S \subseteq |\mathfrak{A}|$ такого, что $\varphi\restriction_S$ инъективно, существует элемент $p \in \mathfrak{A}[S] \setminus \varphi^{-1}(\mathfrak{B}[\varphi S])$. Положим $P = S \cup \{p\}$, $\mathfrak{P} = \langle P, \mathrm{Pol}\,\{A \cap P \mid$

$A \in \mathrm{Sub}\,\mathfrak{A}\}\rangle$, $\theta : |\mathfrak{P}| \hookrightarrow |\mathfrak{A}|$ – включение. Тогда $\theta \in \mathrm{MStr} \subseteq \mathrm{Mor\,HS}$, но $\varphi \circ \theta \notin \mathrm{Str}$, поскольку $\mathrm{Clo}\,\mathfrak{P}$ содержит функцию, принимающую значение p на аргументах из S, а $\varphi \circ \theta$ инъективно. □

Лемма 3.6. CM является нормальной структурной категорией.

Доказательство. В силу лемм 3.1 и 3.2 любой CM-морфизм φ, не принадлежащий Mor HS, можно представить в виде $\varphi = \mu \circ \psi \circ \theta$, где $\mu, \theta \in \mathrm{Mor\,HS}$ и $\psi \in \mathrm{Mor\,CS}$. Поскольку любая полупримальная алгебра проста (это вытекает из предложения 3.5: семейство IFS_A не сохраняет ни одно нетривиальное отношение эквивалентности на A), имеем $\mu \in \mathrm{MStr}$. Отсюда в силу леммы 3.3 $\varphi \in \mathrm{Str}$, поскольку по лемме 3.5 $\psi \circ \theta \in \mathrm{Str}$. Таким образом, CM является структурной категорией; она является нормальной по лемме 3.2 и предложению 3.10. □

Утверждение теоремы 3.1 вытекает из лемм 3.1-3.6.

Покажем, что категория CM находится достаточно близко к нулю нижней полурешетки структурных категорий:

Предложение 3.12. Класс всех структурных категорий, содержащихся в CM, является множеством мощности $\mathfrak{c}$ и содержит континуальную булеву решетку. Класс всех структурных категорий, содержащих CM, содержит булеву решетку мощности $2^{\mathfrak{c}}$ и не является интервалом.

Доказательство. Положим $\pi_n : |\mathrm{P}_n| \to |\mathrm{P}_n| : x \mapsto x \bmod 2$ для любого $n > 2$, так что $\pi_n \in \mathrm{Mor\,CS} \setminus \mathrm{Mor\,HS}$. Для любого $M \subseteq \omega$ обозначим через SC^{π}_M категорию, порожденную классом $\mathrm{HAlg} \cup \mathrm{MStr} \cup \{\pi_{m+3} \mid m \in M\}$. По теореме 3.1 $\mathrm{SC}^{\pi}_M \in \mathcal{S}$. Поскольку $\mathrm{SC}^{\pi}_M \subseteq \mathrm{SC}^{\pi}_K$ тогда и только тогда, когда $M \subseteq K$, соответствие $M \mapsto \mathrm{SC}^{\pi}_M$ задает изоморфизм между континуальной булевой решеткой 2^{ω} и совокупностью всех категорий вида SC^{π}_M, рассматриваемой как решетка по включению. Вместе с тем, ввиду предложения 3.4 совокупность всех полупримальных алгебр, принадлежащих скелету любой структурной категории, образует счетное множество, поэтому $|\{SCA \mid SCA \in \mathcal{S}, SCA \subseteq \mathrm{CM}\}| \leq \mathfrak{c}$.

Далее, положим $N = E_8 \setminus E_4$, существует множество $C \subseteq \mathrm{CL}_N$ мощности $\mathfrak{c}$, состоящее из попарно неэквивалентных клонов (см. [42]). Для любого $K \subseteq C$ обозначим через CM_K категорию, порожденную классом $\mathrm{Mor\,CM} \cup \{\varphi^X \mid X \in K\}$, где

$\varphi^X : |\mathrm{R}^X{}_8| \to |\mathrm{Q}^X{}_8| : x \mapsto \mathrm{If}^1(x, 3, x)$,

$\mathrm{R}^X{}_8 = \langle E_8, \{r\} \cup \{t \mid t \in \mathrm{P}_8, t(E_8, \ldots, E_8) \subseteq N, t\restriction_N \in X\}\rangle$,

$\mathrm{Q}^X{}_8 = \langle E_8, (\mathrm{Pol}\,\{E_8 \times E_8 \setminus \{(3, 0), (3, 2)\}\}) \cap \{t \mid t \in \mathrm{Pol}\,\{N\}, t\restriction_N \in X\}\rangle$,

$r : x \mapsto g(x, (x + 1) \bmod 4)$,

$g : (x, y) \mapsto \mathrm{If}^0([x/4], y, x)$.

Ясно, что $\varphi^X \in \mathrm{Str} \setminus \mathrm{Mor}\,\mathrm{CM}$, и φ^X субинъективно (даже субнепрерывно), поскольку $\mathrm{Sub}\,\mathrm{Q}^X{}_8 = \mathrm{Sub}\,\mathrm{R}^X{}_8 \subseteq 2^N \cup \{E_8\}$. Покажем, что алгебра $\mathrm{Q}^X{}_8$ проста. Действительно, отношение эквивалентности $\{(3, 3)\} \bigcup (E_8 \setminus \{3\} \times E_8 \setminus \{3\})$ не может быть ее конгруэнцией, поскольку в ее клоне операций есть функция $x \mapsto g(x, 3)$. Для любого другого нетривиального отношения эквивалентности $R \subseteq E_8 \times E_8$ выберем его класс эквивалентности A такой, что $|A| > 1$, различные элементы $a_1, a_2 \in A$, и элемент $a_3 \in E_8 \setminus (\{3\} \bigcup A)$. Если существует такое $j \in \{1, 2\}$, что $a_j = 3$, то положим $m = j$, в противном случае положим $m = 1$. Функция $(x, y) \mapsto g(x, \mathrm{If}^{a}{}_{m}(y, a_{3-m}, a_3))$ принадлежит $\mathrm{Clo}\,\mathrm{Q}^X{}_8$ и не сохраняет R. Из простоты алгебры $\mathrm{Q}^X{}_8$ вытекает, в частности, что $\varphi^X \notin \mathrm{Mor}\,\mathrm{CM}_{C\setminus\{X\}}$. Поэтому $\mathrm{CM}_K \subseteq \mathrm{CM}_J$ тогда и только тогда, когда $K \subseteq J$, и соответствие $K \mapsto \mathrm{CM}_K$ задает изоморфизм между булевой решеткой 2^C мощности $2^{\mathfrak{c}}$ и совокупностью всех категорий вида CM_K, рассматриваемой как решетка по включению.

Докажем, что для произвольного $K \subseteq C$ любой CM_K-морфизм $\tau : |\mathfrak{A}| \to |\mathfrak{B}|$ является структурным. Применим индукцию по числу вхождений отображений вида φ^X в его разложение в композицию, состоящее из них и CM-морфизмов. Если оно равно 0, то $\tau \in \mathrm{Str}$ по лемме 3.6. В общем случае τ можно представить в виде $\tau = \mu \circ \psi \circ \nu \circ \varphi^X \circ \theta$, где $\mu, \nu \in \mathrm{MStr}$, $\psi \in \mathrm{Mor}\,\mathrm{CS} \bigcup \{1_{\mathfrak{X}} \mid \mathfrak{X} \in \mathrm{Ob}\,\mathrm{CM}\}$, $\theta \in \mathrm{Str} \bigcap \mathrm{Mor}\,\mathrm{CM}_K$. Ввиду лемм 3.3 и 3.4 требуется показать, что $\varphi^X \circ \theta \in \mathrm{Str}$. Если $\theta \in \mathrm{MStr}$, то это условие проверяется непосредственно. Следовательно, оно справедливо и при $\theta \in \mathrm{Mor}\,\mathrm{HS}$, поскольку $\mathrm{Str} \circ (\mathrm{HAlg} \bigcap \mathrm{Epi}\,\mathrm{HS}) = \mathrm{Str}$. Остается рассмотреть случай $\theta = \gamma \circ \rho$, где $\gamma : |\mathfrak{C}| \to |\mathrm{R}^X{}_8| \in \mathrm{MStr}$ и $\rho : |\mathfrak{A}| \to |\mathfrak{C}|$ для некоторой полупримальной алгебры $\mathfrak{C}$. Это возможно только, если $\gamma(|\mathfrak{C}|) \subseteq N$, но тогда $\varphi^X \restriction_{\theta(|\mathfrak{A}|)}$ – биекция и $\mathrm{Clo}\,\mathrm{Q}^X{}_8 \restriction_{\varphi^X \circ \theta} = \mathrm{Clo}\,\mathrm{R}^X{}_8 \restriction_{\theta}$, поэтому $\varphi^X \circ \theta \in \mathrm{Str}$. Таким образом, мы доказали, что CM_K – структурная категория, поэтому класс $\{SCA \mid SCA \in \mathcal{S}, SCA \supseteq \mathrm{CM}\}$ содержит булеву решетку мощности $2^{\mathfrak{c}}$.

Чтобы показать, что этот класс не является интервалом, построим структурную категорию $\mathrm{CM}' \supseteq \mathrm{CM}$ такую, что класс $\mathrm{Mor}\,\mathrm{CM}' \bigcup \mathrm{Mor}\,\mathrm{CM}_{\{\mathrm{P}_N\}}$ порождает неструктурную категорию. Положим

$$\varphi' : |\mathrm{Q}^{\mathrm{P}_N}{}_8| \to |\mathrm{Q}'_8| : x \mapsto \mathrm{If}^2(x, 0, x),$$

$$\mathrm{Q}'_8 = \langle E_8, \mathrm{Pol}\,\{E_8 \times E_8 \setminus \{(3, 0)\}\}\rangle,$$

так что $\varphi' \in \mathrm{Str} \setminus \mathrm{Mor}\,\mathrm{CM}$ и φ' субнепрерывно, поскольку $\mathrm{Sub}\,\mathrm{Q}'_8 = \{\varnothing, E_8\}$. Пусть CM' – категория, порожденная классом $\mathrm{Mor}\,\mathrm{CM} \bigcup \{\varphi'\}$. Простота алгебры Q'_8 доказывается так же, как для $\mathrm{Q}^X{}_8$. Утверждение, что любой CM'-морфизм является структурным, доказывается так же, как для CM_K-морфизма (индукцией по числу вхождений морфизма φ' в его разложение), за исклю-

чением рассмотрения случая $\theta = \gamma \circ \rho$, где $\gamma : |\mathfrak{C}| \to |Q^{P_N}{}_8| \in$ MStr и $\rho : |\mathfrak{A}| \to |\mathfrak{C}|$ для некоторой полупримальной алгебры $\mathfrak{C}$. Здесь он возможен только, если $\{0, 3\} \not\subseteq \varphi'\gamma(|\mathfrak{C}|)$, поскольку $\mathrm{If}^s(s, 3, s) = 3$ и $\mathrm{If}^s(3, 3, s) = s$ для любого $s \in \{0, 2\}$. А это условие влечет $\mathrm{Clo}\, Q'_8 \restriction_{\varphi' \circ \theta} = P_{\varphi'\theta(|\mathfrak{A}|)}$, поэтому $\varphi' \circ \theta \in$ Str. Таким образом, CM' – структурная категория. Положим $\psi = \varphi' \circ \varphi^{P_N}$, имеем $\psi \notin$ Str, поскольку $r \in \mathrm{Pol}\, \{\ker \psi\}$, но функция $(\mathrm{id}\, \psi)(r/\ker \psi) : x \mapsto g(x, 3 - x)$ не может быть продолжена до функции из $\mathrm{Clo}\, Q'_8$. □

Согласно результатам гл.2, процедуры синтеза и анализа вычислительных систем формализуются пределами и копределами в категориях их моделей. Однако категория CS не обладает достаточным набором универсальных конструкций: можно проверить, что она обладает суммами пар и уравнителями, но в ней в общем случае не определены ни произведения пар, ни коуравнители. Действительно, основные множества вершин и ребра (ко)пределов в CS ввиду предложения 3.11 строятся по тем же правилам, что и в **FinSet**. Так, сумма полупримальных алгебр $\mathfrak{A}$ и $\mathfrak{B}$ имеет основное множество $|\mathfrak{A}| \coprod |\mathfrak{B}|$ и множество подалгебр $\{A \coprod B \mid A \in \mathrm{Sub}\, \mathfrak{A}, B \in \mathrm{Sub}\, \mathfrak{B}\}$. Далее, положим $D_5 = \langle E_5, \mathrm{Pol}\, \{\{0\}, \{1\}, \{2, 3\}\}\rangle$. Если $\rho : |D_5| \to |\mathfrak{X}|$ – коуравнитель CS-морфизмов $\varphi : |\langle E_2, C_2\rangle| \hookrightarrow |D_5| : x \mapsto x$ и $\psi : |\langle E_2, C_2\rangle| \to |D_5| : x \mapsto x + 2$, то $\mathfrak{X} \cong P_3$, однако не существует такого CS-морфизма ν, что $\theta = \nu \circ \rho$, где $\theta : |D_5| \to |C^{\{0\}}{}_2| : x \mapsto [x/4] \in$ Mor CS. Пример пары алгебр, не имеющих произведения в CS, будет приведен в доказательстве теоремы 3.2. Уравнитель пары $\varphi, \psi : |\mathfrak{A}| \to |\mathfrak{B}|$ имеет вид $|\langle Q, \mathrm{Pol}\, \{A \cap Q \mid A \in \mathrm{Sub}\, \mathfrak{A}\}\rangle| \hookrightarrow |\mathfrak{A}|$, где $Q = \{x \in |\mathfrak{A}| \mid \varphi x = \psi x\}$. Отметим, что CS обладает пересечениями: любая пара CS-мономорфизмов вида $\mu : |\mathfrak{A}| \hookrightarrow |\mathfrak{B}| \hookleftarrow |\mathfrak{C}| : \nu$ имеет декартов квадрат, его вершиной является алгебра $\langle \mu(|\mathfrak{A}|) \cap \nu(|\mathfrak{C}|), \mathrm{Pol}\, \{\mu(A) \cap \nu(C) \mid A \in \mathrm{Sub}\, \mathfrak{A}, C \in \mathrm{Sub}\, \mathfrak{C}\}\rangle$.

Заметим, что существует функтор из CS в категорию **FinTop** всех конечных топологических пространств и всех их непрерывных отображений, имеющий левый сопряженный с тождественной единицей и коединицей, состоящей из биекций множеств. Он переводит алгебру $\mathfrak{A}$ в топологическое пространство с носителем $|\mathfrak{A}|$ и базой $\mathrm{Sub}\, \mathfrak{A}$. Этот факт показывает, что категория CS близка по своим свойствам к топологическим категориям (см. [136, определение 21.1]), которые естественным образом возникают в задачах моделирования систем [174]. В частности, ввиду предложения 2.11 технологии, формализуемые топологическими категориями, предоставляют наиболее широкие возможности для комплексирования систем. В связи с этим возникает вопрос о нахождении топологической категории, содержащей достаточное количество структурных отображений полупримальных алгебр. Покажем, что для этого можно использовать наименьшую нормальную категорию, которую мы обозначим через CN. Ее существование

вытекает из предложения 3.10, в силу которого класс всех нормальных категорий образует полную нижнюю подполурешетку в $\mathcal{S}$. Отметим, что он не является решеткой, поскольку категории $\mathrm{CM}_{\{\mathrm{P}_N\}}$ и CM', построенные в доказательстве предложения 3.12, являются нормальными, но объединение их классов морфизмов порождает неструктурную категорию.

Теорема 3.2. Подкатегория вычислительных систем нормальной структурной категории эквивалентна некоторой топологической категории над **FinSet** тогда и только тогда, когда она совпадает с cs(CN).

Доказательство. Пусть CT – категория, состоящая из всех полупримальных алгебр и всех субнепрерывных отображений их основных множеств. Докажем, что класс Mor HS ∪ Mor CT порождает категорию CN. В силу лемм 3.2 и 3.6 он порождает нормальную структурную категорию, поэтому достаточно проверить, что Mor CT ⊆ cs(Mor HS) ∘ MStr. Действительно, любой CT-морфизм $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ разлагается в композицию $\varphi = \varphi' \circ \varepsilon : |\mathfrak{A}| \leftrightarrow |\mathfrak{A}'| \to |\mathfrak{B}|$, где $\mathfrak{A}' = \langle |\mathfrak{A}|, \mathrm{Pol}\, \varphi^{-1}(\mathrm{Sub}\, \mathfrak{B})\rangle$, $\varepsilon \in$ MStr – тождественное отображение множества $|\mathfrak{A}|$ на себя, φ' действует так же, как φ. Обозначим через $\mathfrak{A}''$ алгебру $\langle |\mathfrak{A}|, \mathrm{Pol}\, (\varphi^{-1}(\mathrm{Sub}\, \mathfrak{B}) \cup \{\ker \varphi\})\rangle$; имеем $\mathrm{Sub}\, \mathfrak{A}'' = \mathrm{Sub}\, \mathfrak{A}'$, поскольку $\mathrm{Clo}\, \mathfrak{A}''$ содержит для любого множества $S \in 2^{|\mathfrak{A}|} \setminus \varphi^{-1}(\mathrm{Sub}\, \mathfrak{B})$ функцию

$$f_S(x_1, \ldots, x_k) = \begin{cases} b, & \{\varphi x_1, \ldots, \varphi x_k\} = \varphi S, \\ x_1 & \text{иначе}, \end{cases}$$

где $k = |\varphi S|$, b – произвольный элемент множества $\mathfrak{A}'[S] \setminus S$. Поэтому φ' совпадает (с точностью до изоморфизма) с cs(φ''), где $\varphi'' : |\mathfrak{A}''| \to |\mathfrak{B}| \in$ Mor HS действует так же, как φ.

Проверим по определению, что cs(CN) = CT эквивалентна топологической категории. Рассмотрим произвольный набор отображений конечных множеств вида $\{\varphi_i : A \to |\mathfrak{B}_i| \mid i \in I\}$, где все $\mathfrak{B}_i$ – полупримальные алгебры. Согласно [136, предложение 21.5] (ср. пояснения, приведенные перед предложением 2.11), должна существовать полупримальная алгебра $\mathfrak{A}$, удовлетворяющая следующим условиям: (i) $|\mathfrak{A}| = A$; (ii) $\{\varphi_i : |\mathfrak{A}| \to |\mathfrak{B}_i| \mid i \in I\} \subseteq$ Mor CT; (iii) для любых набора $\{\psi_i : |\mathfrak{C}| \to |\mathfrak{B}_i| \mid i \in I\} \subseteq$ Mor CT и отображения множеств $\chi : |\mathfrak{C}| \to A$ из равенств $\varphi_i \circ \chi = \psi_i$, $i \in I$, вытекает, что $\chi : |\mathfrak{C}| \to |\mathfrak{A}| \in$ Mor CT. Ясно, что в качестве $\mathfrak{A}$ можно выбрать алгебру $\langle A, \mathrm{Pol} \bigcup_{i \in I} \varphi_i^{-1}(\mathrm{Sub}\, \mathfrak{B}_i)\rangle$.

Осталось доказать достаточность условия теоремы. Пусть *SCA* – нормальная структурная категория, содержащая морфизм $\varphi : |\mathfrak{A}| \to |\mathfrak{B}| \in$ Mor CS \ Mor CN, тогда существует множество $B \in \mathrm{Sub}\, \mathfrak{B}$ такое, что $\varphi^{-1}B \notin \mathrm{Sub}\, \mathfrak{A}$. Отображение $\chi : |\mathfrak{B}| \to |\mathrm{C}^{\{0\}}{}_2|$, действующее на $|\mathfrak{B}|$ как характеристическая функция множества $|\mathfrak{B}| \setminus B$, является субнепрерывным (напомним, что через $\mathrm{C}^{\{0\}}{}_2$ обозначается алгебра $\langle E_2, \mathrm{Pol}\, \{\{0\}\}\rangle$). Положим $A = \varphi^{-1}B$, $A' = |\mathfrak{A}| \setminus A$, $m = |A|$, $m' = |A'|$,

$n = \max(m, m')$, $n' = |m - m'| + 1$, q – произвольный элемент меньшего из множеств A и A' (любого, если они равномощны), $\mathfrak{G}$ – результат «приклеивания» алгебры $\mathrm{P}_{n'}$ с отмеченным нулем к алгебре $\mathfrak{A}$ с отмеченной точкой q (т.е. вершина кодекартова квадрата пары морфизмов $|\mathfrak{A}| \leftarrow |\langle\{q\}, {I_1}^1\rangle| \to |\mathrm{P}_{n'}| : q \mapsto 0$ в CT). Существует канонический CT-эпиморфизм $\rho : |\mathfrak{G}| \to |\mathfrak{A}|$, переводящий все точки приклеенной алгебры в q, а остальные точки – в себя. Построим биекцию $\iota : |\mathfrak{G}| \leftrightarrow |E_2 \times E_n|$ такую, что $\iota(\rho^{-1}A) = {E^0}_n$, где ${E^0}_n = \{0\} \times E_n$. Положим $\mathfrak{G}' = \langle E_2 \times E_n, \mathrm{Pol}\, \iota(\mathrm{Sub}\, \mathfrak{G})\rangle$.

Докажем, что в cs(*SCA*) не существует произведения ${\mathrm{C}^{\{0\}}}_2 \times \mathrm{P}_n$ (тогда как любая топологическая категория над **FinSet** конечно полна). Предположим противное, обозначим произведение через $\mathfrak{X}$. Ввиду наличия пары CT-морфизмов $\varphi_1 : |{\mathrm{C}^{\{0\}}}_2| \leftarrow |\langle E_2 \times E_n, \mathrm{Pol}\, \{{E^0}_n\}\rangle| \to |\mathrm{P}_n| : \varphi_2$, действующих на $E_2 \times E_n$ как проекции произведения множеств, имеем $\mathrm{Sub}\, \mathfrak{X} \subseteq \{\varnothing, {E^0}_n, E_2 \times E_n\}$. Однако существуют cs(*SCA*)-морфизм $\varphi'_1 = \chi \circ \varphi \circ \rho \circ \iota^{-1} : |\mathfrak{G}'| \to |{\mathrm{C}^{\{0\}}}_2|$ и CT-морфизм $\varphi'_2 : |\mathfrak{G}'| \to |\mathrm{P}_n|$, также действующие на $E_2 \times E_n$ как проекции. Отсюда $\mathfrak{X} \cong \mathrm{P}_{|\mathfrak{X}|}$, поскольку ${E^0}_n \notin \mathrm{Sub}\, \mathfrak{G}'$. Но тогда проекция $\pi : |\mathfrak{X}| \to |{\mathrm{C}^{\{0\}}}_2|$ не субинъективна, следовательно по лемме 3.5 *SCA* не может быть структурной категорией, что противоречит ее выбору. Теорема доказана. □

Следствие 3.2.1. Категория cs(CN) конечно полна и конечно кополна. □

Охарактеризуем положение категории CN в нижней полурешетке $\mathcal{S}$.

Предложение 3.13. Класс всех структурных категорий, содержащихся в CN, является множеством мощности $\mathfrak{c}$ и содержит континуальную булеву решетку. Интервал между CN и CM является множеством мощности $\mathfrak{c}$ и содержит континуальную булеву решетку, состоящую из нормальных категорий.

Доказательство. Все категории ${\mathrm{SC}^{\pi}}_M$, построенные в доказательстве предложения 3.12 и образующие континуальную булеву решетку по включению, содержатся в CN. Пусть теперь $\mathrm{V}_n = \langle E_n, \mathrm{Pol}\, \{\{x\} \mid x \in E_n\}\rangle$, $\pi'_n : |\mathrm{V}_n| \to |\mathrm{V}_n| : x \mapsto x \bmod 2$ для любого $n > 2$, так что $\pi'_n \in \mathrm{Mor\ CS} \setminus \mathrm{Mor\ CN}$. Все категории вида ${\mathrm{SC}^{\pi'}}_M$ являются нормальными и образуют континуальную булеву решетку по включению. □

Заметим, что нетрудно построить структурную категорию, не содержащуюся ни в какой нормальной категории. Ввиду предложения 3.10 и леммы 3.6, такими «ненормализуемыми» являются в точности все структурные категории, содержащие несубинъективные отображения. Примером служит категория, порожденная классом $\mathrm{Mor\ HS} \cup \{\psi\}$, где $\psi : |\langle E_3, \square\rangle| \to |\langle E_2, {I_1}^1\rangle| : x \mapsto [x/2]$, $\square x = (x + 1) \bmod 3$.

На базе категории CT = cs(CN) можно построить полноценную формальную технологию проектирования распределенных вычислительных систем. Конфигурациям систем отвечают все конечные диаграммы, так что композиционные возможности вычислительных компонентов ничем не ограничены – любая конфигурация распределенной вычислительной системы, не разрушающая структуру операций, является допустимой. Трансформациями моделей вычислений, заданных полупримальными алгебрами, являются в точности все структурные биекции (HS-эпимономорфизмы) – модификации сигнатур алгебр и их интерпретаций, обогащающие (либо сохраняющие) клоны их термальных операций. Поэтому технология проектирования вычислительных систем обладает свойством однородности.

Предложение 3.14. Четверка

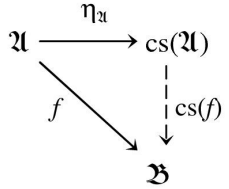
$\mathrm{CSD} = \langle \mathrm{CT}, \mathrm{Ob}\,(\mathbf{FinCat} \downarrow \mathrm{CT}), |\text{–}| : \mathrm{CT} \to \mathbf{FinSet}, (\mathrm{Ob\ CT}, \mathrm{Epi\ CT} \cap \mathrm{Mono\ CT})\rangle$

является структурируемой скоординированной однородной формальной технологией проектирования.

Доказательство. Ввиду теоремы 3.2 и предложения 2.11, тройка $\langle \mathrm{CT}, \mathrm{Ob}\,(\mathbf{FinCat} \downarrow \mathrm{CT}), |\text{–}|\rangle$ является скоординированной формальной технологией специфицирования. Теорема 3.2 также гарантирует, что $\mathrm{Epi\ CT} \cap \mathrm{Mono\ CT} = \mathrm{Init}^{\uparrow}$, поэтому из предложения 2.18 вытекает свойство однородности. Структурируемость вытекает из того, что CSD поддерживает параллелизм: как обычно в технологиях над **Set**, сопоставление полупримальной алгебре $\mathfrak{A}$ множества компонентов разбиения, которое является инициальным объектом в $\mathit{Sums} \downarrow \mathfrak{A}$, задает функтор дискретной структуры, действующий из CT в **FinSet**. □

Класс всех трансформаций в этой технологии совпадает с классом всех перегрузок. Она поддерживает разбиения на примальные алгебры: наиболее мелкое (многокомпонентное) разбиение такого рода состоит из одноэлементных алгебр, а наиболее крупное – из примальных алгебр над классами эквивалентности отношения $R_{\mathfrak{A}} = \{(x, y) \mid \mathfrak{A}[\{x\}] = \mathfrak{A}[\{y\}]\}$. Все такие разбиения являются неразрушающими.

Рефлектор cs в категории CN можно рассматривать как формальное представление процедуры порождения моделей вычислений. Напомним, что для произвольных конечной алгебры $\mathfrak{A}$ и полупримальной алгебры $\mathfrak{B}$ он порождает биекцию множества $\mathrm{Mor}(\mathfrak{A}, \mathfrak{B})$ на $\mathrm{Mor}(\mathrm{cs}(\mathfrak{A}), \mathfrak{B})$ при помощи единицы рефлексии η:



Однако он не в полной мере отражает работу, которую необходимо проделать для получения практически полезных вычислительных систем из результатов формального ограничения

арифметических операций на конечные вычислительные ресурсы. Прежде всего, в целях согласования вычислительных компонентов по данным выделяют структурные отображения, сохраняющие все константы (заметим в связи с этим, что гомоморфизмы алгебр сохраняют все сигнатурные константы, но в общем случае не сохраняют операции, не зависящие существенным образом от своих аргументов). Также требуется предусмотреть обнаружение и обработку ошибок, возникающих при выходе результатов вычислений за рамки поддерживаемой совокупности чисел (см. условие (С3) из разд.3.1). Для этого к основному множеству модели вычислений добавляют *флаг* – специальную константу, возвращаемую при переполнениях и округлениях [121]. При этом следят, чтобы не появилось существенно новых операций, кроме необходимых для обработки флага. При помощи флага можно задавать частичные структурные отображения основных множеств алгебр, представляющие интеграцию вычислительных компонентов в условиях неполного кодирования данных: незакодированным числам сопоставляется флаг. Идею введения флага можно увидеть в доказательстве предложения 3.1.

Наличие флага позволяет проинтерпретировать арифметические предикаты: предикату сопоставляется функция, возвращающая флаг в точности на тех значениях аргументов, на которых предикат является истинным. Структурные отображения, переводящие флаг в себя, сохраняют истинность поддерживаемых предикатов (аналогично гомоморфизмам в теоретико-модельном смысле). Если в модели вычислений имеются функции, пригодные в качестве интерпретаций пропозициональных логических связок, то она приобретает характер арифметико-логической. Примеры таких моделей рассмотрены ниже в разд.3.5.

Изложим эти соображения при помощи введенных выше конструкций. Напомним, что подалгебры – это все мономорфные гомоморфизмы.

Определение 3.9. CN-морфизм φ называется *центральным*, если для любого мономорфного гомоморфизма μ с кообластью $\operatorname{codom}\varphi$ существует CN-морфизм ν такой, что $\varphi = \mu \circ \nu$. CN-морфизм φ называется *главным*, если морфизм $\varphi \circ \gamma$ является центральным для любого центрального морфизма γ с кообластью $\operatorname{dom}\varphi$. □

Ясно, что cs(CN)-морфизм $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$ является главным тогда и только тогда, когда он сохраняет все константы в том смысле, что $\varphi(\operatorname{Cen}\mathfrak{A}) \subseteq \operatorname{Cen}\mathfrak{B}$. Однако нам этого недостаточно: нужны морфизмы, сохраняющие флаг. Пусть $\mathfrak{A}$ – произвольная конечная алгебра, $\langle\{*\}, I_1^{\,1}\rangle$ – одноэлементная алгебра; будем говорить, что центральный морфизм $\gamma : |\langle\{*\}, I_1^{\,1}\rangle| \hookrightarrow |\mathfrak{A}|$ предопределяет константу $*$ в $\mathfrak{A}$, если он действует как включение множеств. Обозначим через CN_* подкатегорию в CN, состоящую из всех конечных алгебр и всех таких главных морфизмов $\varphi : |\mathfrak{A}| \to |\mathfrak{B}|$, что морфизм $\varphi \circ \gamma$ предопределяет константу $*$ в $\mathfrak{B}$ для любого морфизма γ, пред-

определяющего ее в $\mathfrak{A}$. Пусть CS_* – полная подкатегория в CN_*, объектами которой являются все полупримальные алгебры, в которых предопределена константа $*$.

Предложение 3.15. CS_* эквивалентна топологической категории над категорией **PFinSet** всех конечных множеств и всех их частичных отображений.

Доказательство. Проверяется по определению топологической категории (аналогично второму абзацу доказательства теоремы 3.2). □

Предложение 3.16. Для любой константы $*$ четверка

$CSD_* = \langle CS_*, Ob\ (\mathbf{FinCat} \downarrow CS_*), |\text{–}| \setminus \{*\} : CS_* \to \mathbf{PFinSet}, (Ob\ CS_*, Epi\ CS_* \bigcap Mono\ CS_*)\rangle$

является структурируемой скоординированной однородной формальной технологией проектирования.

Доказательство. Вытекает из предложений 3.15, 2.11, 2.18, аналогично предложению 3.14. □

В отличие от CSD, вместо разбиения на примальные алгебры технология CSD_* поддерживает (неразрушающие) разбиения на алгебры с клонами вида P_X и $C^{\{*\}}{}_X$: наиболее мелкое разбиение CS_*-объекта $\mathfrak{A}$ такого рода состоит из алгебр над множествами $\{x, *\}$, $x \in |\mathfrak{A}| \setminus \{*\}$, а наиболее крупное, которое мы будем называть *базовым разбиением* – из алгебр $\langle X \bigcup \{*\}, Pol\ \{\{*\}\}\rangle$, по одной для каждого класса эквивалентности X отношения $R_{\mathfrak{A}}$, не содержащего константу $*$ (т.е. отличного от $Cen\ \mathfrak{A}$), к которым при $|Cen\ \mathfrak{A}| > 1$ добавляется алгебра $P_{Cen\ \mathfrak{A}}$.

Предложение 3.17. CS_* рефлективна в CN_*.

Доказательство. Если константа $*$ предопределена в алгебре $\mathfrak{A}$, то действие рефлектора на $\mathfrak{A}$ совпадает с действием функтора cs. В противном случае он переводит $\mathfrak{A}$ в алгебру $\langle |\mathfrak{A}| \coprod \{*\}, Pol\ \{A \coprod \{*\} \mid A \in Sub\ \mathfrak{A} \setminus \{\varnothing\}\}\rangle$, а единица рефлексии действует как включение первой компоненты в раздельное объединение множеств (оно является субнепрерывным, следовательно, структурным ввиду леммы 3.4). □

Определение 3.10. *Вычислительным расширением* конечной алгебры $\mathfrak{A}$ с флагом $*$ называется алгебра $cs_*(\mathfrak{A})$, где $cs_* : CN_* \to CS_*$ – рефлектор. □

Результаты разд.3.2-3.4 позволяют сконструировать функтор, отвечающий отображению вычислительных задач на архитектуру распределенной вычислительной среды (строго говоря, получается целый класс однотипных функторов). Поскольку задачи изначально специфицируются посредством конструкций арифметических АТД, задаваемых аксиоматически, в качестве области функтора целесообразно выбрать категорию LS_σ всех теорий некоторой сигнатуры σ,

определенную в разд.2.3. Его кообластью должна выступать категория вида CS_* с подходящим флагом, в роли которого может выступать пустое множество:

Предложение 3.18. Для произвольных сигнатуры σ и конечной подсигнатуры $\sigma_0 \subseteq \sigma$ существует функтор из категории LS_σ в $\mathrm{CS}_\varnothing$, переводящий любую непротиворечивую теорию сигнатуры σ в вычислительное расширение ее частичной интерпретации в сигнатуре σ_0 с флагом $\varnothing$.

Доказательство. Будем обозначать искомый функтор через cspi. Пусть T – произвольная теория сигнатуры σ. Если T противоречива либо σ_0 не содержит константных символов, то положим $\mathrm{cspi}(T) = \langle\{\varnothing\}, {I_1}^1\rangle$. В противном случае пусть $\mathfrak{A}$ – частичная интерпретация теории T в сигнатуре σ_0, построенная как в доказательстве предложения 3.1. Обозначим через $\mathfrak{A}^+$ результат обогащения алгебраической системы $\mathfrak{A}$ следующими символами: одной константой, интерпретируемой как $\varnothing$, и для каждого предиката $P \in \sigma_0 \setminus \{\equiv\}$ по одной функции F_P такой же местности, что и P, принимающей значение $\varnothing$ на тех и только тех кортежах аргументов, на которых P истинен (и произвольное значение из $|\mathfrak{A}| \setminus \{\varnothing\}$ на остальных кортежах). В результате можно считать $\mathfrak{A}^+$ алгеброй: при проверке выполнимости в ней формул сигнатуры σ можно заменять в них все вхождения предиката $P(\ldots)$ формулами вида $F_P(\ldots) \equiv \varnothing$. Поскольку $\mathrm{Cen}\,\mathfrak{A}^+ = |\mathfrak{A}|$, имеем $\mathrm{cs}_\varnothing(\mathfrak{A}^+) \cong \mathrm{P}_{|\mathfrak{A}|}$, так что положим $\mathrm{cspi}(T) = \mathrm{P}_{|\mathfrak{A}|}$. Отметим, что алгебру $\mathrm{P}_{|\mathfrak{A}|}$ можно получить из $\mathfrak{A}^+$ путем обогащения сигнатуры (например добавлением штриха Шеффера в $\mathrm{P}_{|\mathfrak{A}|}$), так что $\mathrm{cspi}(T)$ – частичная интерпретация теории T в сигнатуре σ_0. Если S, T – две произвольные непротиворечивые теории сигнатуры σ такие, что $T \vdash S$, то единственному LS_σ-морфизму из S в T сопоставляется отображение $\varphi : |\mathrm{cspi}(S)| \to |\mathrm{cspi}(T)|$, которое строится следующим образом. Положим $\varphi(\varnothing) = \varnothing$, а для любого $x \in |\mathrm{cspi}(S)| \setminus \{\varnothing\}$ положим $\varphi(x)$ равным интерпретации в $\mathrm{cspi}(T)$ константы, интерпретируемой в $\mathrm{cspi}(S)$ как x. Ясно, что $\varphi(x)$ не зависит от выбора конкретной константы такого рода, поскольку если $S \vdash c_1 \equiv c_2$, то и $T \vdash c_1 \equiv c_2$. Кроме того, ввиду определения 3.8 имеем $\varphi \in \mathrm{Mor}\,\mathrm{CS}_\varnothing$, так что действительно имеется функтор $\mathrm{cspi} : LS_\sigma \to \mathrm{CS}_\varnothing$. □

Отметим, что функтор cspi не индуцирует морфизм формальной технологии аксиоматизации спецификаций TS_σ в технологию проектирования вычислительных систем $\mathrm{CSD}_\varnothing$ даже на уровне технологий конфигурирования, поскольку в состав конфигураций в TS_σ входят бесконечные диаграммы. В общем случае нельзя построить с его помощью и **CONF**-морфизм из «финитной» подтехнологии $\boldsymbol{conf}(\boldsymbol{spec}(FTS_\sigma))$ (см. разд.2.6) в $\boldsymbol{conf}(\boldsymbol{spec}(\mathrm{CSD}_\varnothing))$, поскольку он не сохраняет копределы: копроизведение двух неодноэлементных примальных алгебр в $\mathrm{CSD}_\varnothing$ не примально. Это свидетельствует о глубоком концептуальном разрыве между постановкой вычислительных задач и их реализацией на компьютерах с фоннеймановской архитектурой.

3.5 Архитектура арифметики и логика Лукасевича

В современных компьютерах чаще всего используется модель целочисленных вычислений по mod n, $n > 1$, операции которой заимствуются из кольца $\mathbf{Z}/n\mathbf{Z}$. Значения из старшей половины его основного множества трактуются как отрицательные числа, упорядоченные по убыванию абсолютной величины от минимального значения $-[n/2]$ до -1, а младшая интерпретирует начальный отрезок множества натуральных чисел от 0 до $[(n-1)/2]$. В качестве флага используется добавочное значение n, называемое переносом (carry) и неотличимое от нуля при сложении и умножении. Предусматривается унарное отрицание, переводящее нуль во флаг (дополнительный код [17]) и операция ветвления потока вычислений по флагу (предикация [121]). Получается алгебраическая система следующего вида (см. [68]):

$$\mathrm{MA}_{n+1} = \langle E_{n+1}, 0, 1, \ldots, [(n-1)/2], -[n/2], \ldots, -1, (=), (+), (-), (\times), (\mathrm{Carry})\rangle,$$
$$x\,(=)\,y \Leftrightarrow x \equiv y \bmod n,$$
$$x\,(+)\,y = (x+y) \bmod n,$$
$$(-)x = n - x,$$
$$x\,(\times)\,y = xy \bmod n,$$
$$(\mathrm{Carry})(x, y, z) = \mathrm{If}^n(x, y, z).$$

Ясно, что MA_{n+1} верифицирует все предложения из проекции теории $\mathrm{T}(\mathbf{Z})$ арифметики целых чисел на подсигнатуру, не содержащую предикацию:

$$\mathrm{MA}_{n+1} \models \mathrm{T}(\mathbf{Z}) \downarrow (\sigma(\mathrm{MA}_{n+1}) \setminus \{(\mathrm{Carry})\}).$$

Функции этой системы (не считая констант) порождают клон вычислительного расширения кольца $\mathbf{Z}/n\mathbf{Z}$ не при всех значениях n. Чтобы точно сформулировать и доказать этот факт, а также указать варианты выбора недостающих функций, мы привлекаем аппарат конечнозначной логики Лукасевича [67]. Напомним, что логической матрицей Лукасевича [54] называется алгебраическая система

$$Ł_{n+1} = \langle E_{n+1}, \sim, \to, D_n\rangle,$$
$$\sim x = n - x,$$
$$x \to y = \min(n, n - x + y),$$
$$D_n(x) \Leftrightarrow (x = n).$$

В дальнейшем мы будем обозначать тем же символом $Ł_{n+1}$ алгебру, получающуюся из матрицы Лукасевича путем обеднения до сигнатуры $\{\sim, \to\}$ (а также ее клон). Заметим, что обеднение не приводит к ослаблению множества истинных предложений матрицы, поскольку $D_n(x) \Leftrightarrow (x \to x = x)$. Центроидом алгебры $Ł_{n+1}$ является множество $\{0, n\}$, он является сильным, поскольку сужение ее операций на него образует матрицу классической двузначной логики (с n

в качестве значения «истина»). Любой предикат, определенный на множестве E_{n+1}, может быть выражен в $Ł_{n+1}$ функцией, принимающей значение n на наборах аргументов, на которых предикат истинен, и 0 в противном случае.

Теорема 3.3. Вычислительное расширение кольца $\mathbf{Z}/n\mathbf{Z}$ с флагом n изоморфно $Ł_{n+1}$.

Доказательство. Как показано в [54, с.108], известный критерий Мак-Нотона выразимости функций через операции матрицы Лукасевича эквивалентен утверждению, что клон $Ł_{n+1}$ имеет вид Pol $\{V(d) \mid d \setminus n\}$, где

$$V(d) = E_{n+1} \cap |d\mathbf{Z}|.$$

Поэтому $n \in \mathrm{Clo}\, Ł_{n+1}$, и $\{S \setminus \{n\} \mid S \in \mathrm{Sub}\, Ł_{n+1}\} = \mathrm{Sub}\, \mathbf{Z}/n\mathbf{Z}$. Согласно конструкции вычислительного расширения, описанной в доказательстве предложения 3.17, получаем, что $\mathrm{cs}_n(\mathbf{Z}/n\mathbf{Z}) \cong Ł_{n+1}$. (Конечно, мы рассматриваем $\mathbf{Z}/n\mathbf{Z}$ как кольцо без единицы, т.е. обозначаем этим символом обеднение кольца вычетов до сигнатуры кольца без единицы, поскольку кольцо вычетов с единицей вообще не имеет нетривиальных подколец). □

Базовое разбиение алгебры $Ł_{n+1}$ состоит из алгебр с основными множествами вида $\{x \mid x \in E_{n+1}, \mathrm{НОД}(x, n) = d\} \cup \{n\}$, по одному для каждого $d \setminus n$ (здесь полагаем $\mathrm{НОД}(0, n) = n$). Примером нетривиального субнепрерывного отображения матриц Лукасевича, сохраняющего флаг, служит $\varphi_{n,k} : |Ł_{n+1}| \to |Ł_{kn+1}| : x \mapsto kx$ для произвольного $k > 1$.

Положим

$$\mathrm{МŁ}_{n+1} = \{(+), (-), (\mathrm{Carry})\},$$

$$\mathrm{МСŁ}_{n+1} = \begin{cases} \mathrm{МŁ}_{n+1}, & n = 2, \\ \mathrm{МŁ}_{n+1} \cup \{(\times)\} & \text{иначе}, \end{cases}$$

обозначим через П множество всех конечных произведений попарно различных простых чисел.

Теорема 3.4. Множество функций $\mathrm{МСŁ}_{n+1}$ образует базис в $Ł_{n+1}$ тогда и только тогда, когда $n \in$ П либо $n = 2m$ для некоторого $m \in$ П. В противном случае $\mathrm{МСŁ}_{n+1}$ образует базис в клоне квазипримальной алгебры с предопределенной константой n, вычислительное расширение которой с флагом n изоморфно $Ł_{n+1}$. □

Следствие 3.4.1. Множество констант и функций алгебраической системы $\mathrm{МА}_{n+1}$ полно в P_{n+1}. □

Справедливость теоремы 3.4 вытекает из нижеследующих лемм. В их доказательствах через $k * x$ обозначается сложение $k > 0$ экземпляров переменной x по модулю n (полагаем $1 * x = x$). Также используется одноместная функция

$$p_{n+1}(x) = \mathrm{If}^n(x, x, \mathrm{If}^0(x, x, {\sim}x)),$$

которая при $n > 2$ является нетривиальной перестановкой множества E_{n+1}.

Лемма 3.7. Множество $\mathrm{МСŁ}_{n+1}$ независимо.

Доказательство.

(i) $\text{МСЛ}_{n+1} \setminus \{(+)\}$ сохраняет множество $\{1, n-1\}$.

(ii) $\text{МСЛ}_{n+1} \setminus \{(-)\}$ сохраняет множество $\{0\}$.

(iii) $\text{МСЛ}_{n+1} \setminus \{(\text{Carry})\}$ сохраняет отношение эквивалентности (=).

(iv) $\text{МСЛ}_{n+1} \setminus \{(\times)\}$ самодвойственно относительно p_{n+1}. □

Лемма 3.8. $\{\sim, \text{If}^n, \text{If}^0, 0, n, (=), d_3\} \subseteq \text{Clo}\,\langle E_{n+1}, \text{МЛ}_{n+1}\rangle$.

Доказательство. Имеем

$$\sim x = (-)x,$$

$$\text{If}^n(x, y, z) = (\text{Carry})(x, y, z),$$

$$\text{If}^0(x, y, z) = \text{If}^n(\sim x, y, z),$$

$$0 = n * x,$$

$$n = \sim 0,$$

$$x\ (=)\ y = \text{If}^0(x\ (+)\ \sim y, n, 0),$$

$$d_3(x, y, z) = \text{If}^0(x\ (+)\ \sim y, \text{If}^n(x, \text{If}^0(y, x, z), \text{If}^0(x, \text{If}^0(y, z, x), z)), x).$$ □

Лемма 3.9. $\text{Sub}\,\langle E_{n+1}, \text{МЛ}_{n+1}\rangle = \text{Sub}\,\langle E_{n+1}, \text{МСЛ}_{n+1}\rangle = \text{Sub}\,\text{Ł}_{n+1}$.

Доказательство. С одной стороны, по теореме 3.3 $\text{МЛ}_{n+1} \subseteq \text{МСЛ}_{n+1} \subseteq \text{Ł}_{n+1}$, поэтому $\text{Sub}\,\langle E_{n+1}, \text{МЛ}_{n+1}\rangle \supseteq \text{Sub}\,\langle E_{n+1}, \text{МСЛ}_{n+1}\rangle \supseteq \text{Sub}\,\text{Ł}_{n+1}$. С другой стороны, любая подалгебра $A \in \text{Sub}\,\langle E_{n+1}, \text{МЛ}_{n+1}\rangle$ вместе с любым элементом a содержит все элементы вида $k * a$, включая 0 (когда $k = n$), а значит, и $n = (-)0$. Используя алгоритм Евклида, мы получаем, что $A = V(\text{НОД}\,A \setminus \{0\}) \in \text{Sub}\,\text{Ł}_{n+1}$. □

Лемма 3.10. Множество МСЛ_{n+1} полно в Ł_{n+1} тогда и только тогда, когда $n \in \text{П} \cup 2\text{П}$.

Доказательство. Полнота множества МСЛ_{n+1} в Ł_{n+1} равносильна возможности выразить функции If^s, $s \in E_{n+1} \setminus \{0, n, [n/2]\}$, через его элементы. Действительно, если такая возможность имеется, то полнота вытекает из лемм 3.8, 3.9 и утверждения (ii) предложения 3.5. В частности, МСЛ_3 полно в Ł_3.

При $n > 2$ сначала докажем достаточность условия леммы. Ввиду п. (iv) доказательства леммы 3.7, если существует число $d > 2$ такое, что $d \setminus n$ и сужение умножения по модулю n на множество $V(n/d)$ самодвойственно относительно перестановки p_{n+1}, то МСЛ_{n+1} сохраняет нетривиальное (не содержащееся в диагонали) бинарное отношение $\{(x, p_{n+1}(x)) \mid x \in V(n/d)\}$. Иначе говоря, в этом случае все сигнатурные функции алгебры $\langle E_{n+1}, \text{МСЛ}_{n+1}\rangle$ сохраняют автоморфизм ее подалгебры $V(n/d)$, индуцированный функцией p_{n+1}. В то же время функция $\text{If}^{n/d}$ не сохраняет его, откуда вытекает неполнота МСЛ_{n+1} в Ł_{n+1}. Если $n \equiv 8 \bmod 16$, то можно выбрать $d = 4$, поскольку в этом случае $(n/4)^2 \equiv n/2 \bmod n$. Иначе либо можно выбрать $d > 2$ такое, что $d^2 \setminus n$ (так что $(n/d)^2 \equiv 0 \bmod n$), либо n имеет вид, указанный в условии леммы.

Для доказательства необходимости условия будем искать выражение для If^s, исследуя сравнение $x(x - s) \equiv 0 \bmod n$. Введем обозначения $u = \mathrm{НОД}(s, n)$, $Q = \{q_1, \ldots, q_r\}$ – множество простых делителей числа n/u.

Пусть сначала $n \in \Pi$. Докажем справедливость тождества

$$\mathrm{If}^s(x, y, z) = \mathrm{If}^0((x\ (\times)\ x)\ (+)\ ((n - s) * x),$$
$$\mathrm{If}^n(x, z, \mathrm{If}^0((n/q_1) * x, z, \ldots, \mathrm{If}^0((n/q_r) * x, z, y)\ldots)),$$
$$z).$$

Предположим, что для некоторого s оно не выполняется, тогда существует решение исследуемого сравнения $r \in E_{n+1} \setminus \{0, s\}$, не делящееся ни на одно q_j. Тогда r не делится на $\prod_j q_j = n/u$, поэтому $r - s = kn/u$ для некоторого целого $k \neq 0$. Но тогда

$$r(r - s) = (kn/u + s)kn/u = k^2(n/u)^2 + (s/u)kn.$$

Отсюда $k^2(n/u)^2 = vn$ для некоторого $v > 0$, т.е. $k^2n/u = vu$. Поскольку $\mathrm{НОД}(n/u, u) = 1$, то k^2 кратно u, а значит, $k = tu$ для некоторого целого $t \neq 0$. Но тогда $r = tun/u + s = tn + s$, что противоречит выбору r. Поэтому при $n \in \Pi$ вышеуказанное тождество для If^s выполняется.

Пусть теперь $n = 4\prod_i p_i$, где все p_i попарно различны и нечетны (здесь же рассматриваем случай $n = 4$). Если s нечетно, то приведенные рассуждения остаются в силе, если заменить во множестве Q число 2 числом 4, поскольку r и $r - s$ обладают разной четностью. Если s делится на 4, то поиск значений r приводит к числу $r' = (s + n/2) \bmod n$. Однако оно не делится на 4, поэтому тождество для If^s останется в силе, если заменить в нем вхождение y термом $\mathrm{If}^0((n/4) * x, y, z)$. Наконец, если $s \equiv 2 \bmod 4$ (причем $s \neq n/2$ – см. первую фразу доказательства), то сравнение $2x \equiv 2s \bmod n$ имеет два решения s и r', причем последнее делится на 4. Поэтому $\mathrm{If}^s(x, y, z) = \mathrm{If}^{2*s}(2 * x, \mathrm{If}^0((n/4) * x, z, y), z)$. □

Утверждение теоремы 3.4 непосредственно вытекает из лемм 3.7-3.10. Отметим, что, поскольку по лемме 3.8 алгебра $\langle E_{n+1}, \mathrm{МСЛ}_{n+1}\rangle$ квазипримальна при любом $n > 1$, ее клон состоит из всех функций, сохраняющих все автоморфизмы всех ее подалгебр (ср. доказательство предложения 3.4). Лемма 3.10 дает критерий того, что все эти автоморфизмы тривиальны. Конечно, он совпадает с критерием отсутствия нетривиальных автоморфизмов подколец в $\mathbf{Z}/n\mathbf{Z}$.

Для случая, когда n не удовлетворяет условию теоремы 3.4, возникает вопрос о нахождении базиса в $Ł_{n+1}$, естественного с точки зрения синтеза моделей вычислений по $\bmod n$. Для этого можно привлечь, в частности, операцию вычисления максимума двух чисел относительно любого линейного порядка на множестве E_{n+1}, поскольку она сохраняет все его подмножества (ее значение всегда совпадает со значением одного из аргументов) и не сохраняет нетождественные перестановки любого подмножества (ни одна из которых не монотонна). Рассмотрим числовое упорядочение констант арифметики $\mathrm{МА}_{n+1}$, располагающее элементы E_{n+1} в следую-

щей последовательности: $[(n + 1)/2]$, …, n, 0, …, $[(n – 1)/2]$. Заметим, что (при $n > 2$) сложение по mod n не является монотонным относительно этого порядка. Обозначим через (max) бинарную операцию, возвращающую значение аргумента, максимального относительно него.

Предложение 3.19. При $n > 2$ множество функций $\text{МЛ}_{n+1} \cup \{(\max)\}$ образует базис в Л_{n+1}.

Доказательство. Независимость доказывается точно так же, как в лемме 3.7, с заменой умножения функцией (max). Полнота вытекает из лемм 3.8 и 3.9. □

Положим

$$\text{ММЛ}_{n+1} = \begin{cases} \text{МСЛ}_{n+1}, & n \in \Pi \cup 2\Pi, \\ \text{МЛ}_{n+1} \cup \{(\max)\} & \text{иначе.} \end{cases}$$

Из предложения 3.19 и теоремы 3.4 вытекает следующее утверждение.

Следствие 3.4.2. Множество функций ММЛ_{n+1} образует базис в Л_{n+1}. □

Таким образом, соотношение $\text{cs}_n(\mathbf{Z}/n\mathbf{Z}) \cong \text{Л}_{n+1}$ является показательным примером нетривиального вычислительного расширения. Показано, что наибольшую эффективность имеет машинная арифметика по mod 2 (ее можно построить даже без аппаратной реализации умножения), а наименьшую – по модулю, который делится на квадрат нечетного простого числа либо на 8 (для нее недостаточно кольцевых операций и предикации). Последний вывод интересен, в частности, с точки зрения реализации цифрового шифра RSA, требующей вычислений по mod pq, где p и q – различные простые числа [114].

Теперь кратко рассмотрим машинную арифметику с переполнением. Здесь сложение и умножение неотрицательных целых чисел реализуется так, что любой результат, превосходящий некоторый пороговое значение $n > 1$, принудительно приводится к этому значению. Поэтому основное множество этой арифметики образует представление начального отрезка E_{n+1}. В качестве вычитания на нем задается правая разность (см. [123]). Ввиду своего аномального поведения при вычислениях значение n выступает также в качестве флага. Получается алгебраическая система следующего вида (см. [68]):

$$\text{ОА}_{n+1} = \langle E_{n+1}, 0, 1, \ldots, n, =, [+], [-], [\times], (\text{Carry})\rangle,$$
$$x\ [+]\ y = \min(n, x + y),$$
$$x\ [-]\ y = \max(0, x - y),$$
$$x\ [\times]\ y = \min(n, xy).$$

Ясно, что ОА_{n+1} верифицирует все предложения из проекции теории $\text{Т}(\mathbf{Z})$ на подсигнатуру, не содержащую флаг:

$$\text{ОА}_{n+1} \models \text{Т}(\mathbf{Z}) \downarrow (\sigma(\text{ОА}_{n+1}) \setminus \{n\}).$$

По аналогии с теоремой 3.3, установим, что алгебра, сигнатура которой состоит из вычислительных операций системы OA_{n+1} и флага, изоморфна логической матрице Лукасевича. Поскольку флаг изначально присутствует в основном множестве этой алгебры (случай (i) предложения 3.3), функтор вычислительного расширения не изменяет ее основное множество и совокупность подалгебр. Положим

$$\mathrm{OŁ}_{n+1} = \{[+], [-], [\times]\},$$

$$\mathrm{OAŁ}_{n+1} = \{n, [-]\}.$$

Теорема 3.5. Множество функций $\mathrm{OAŁ}_{n+1}$ образует базис в $\mathrm{Ł}_{n+1}$.

Доказательство. Полнота следует из тождеств $x\,[-]\,y = {\sim}(x \to y)$ (так что $[-] \in \mathrm{Ł}_{n+1}$), ${\sim}x = n\,[-]\,x$ и $x \to y = n\,[-]\,(x\,[-]\,y)$, а независимость – из соотношения $[-] \in \mathrm{C}^{\{0\}}{}_{n+1}$. □

Следствие 3.5.1. Множество констант и функций алгебраической системы OA_{n+1} полно в P_{n+1}. □

Следствие 3.5.2. Множество функций $\mathrm{OŁ}_{n+1} \setminus \{[\times]\}$ образует базис в клоне $\mathrm{Ł}_{n+1} \cap \mathrm{C}^{\{0\}}{}_{n+1}$, содержащем умножение с переполнением, и является предполным в $\mathrm{Ł}_{n+1}$.

Доказательство. Независимость вытекает из соотношений $[+] \in \mathrm{C}^{\{n\}}{}_{n+1}$, $[-] \in \mathrm{C}^{\{0,1\}}{}_{n+1}$. Соотношение $\mathrm{OŁ}_{n+1} \subseteq \mathrm{Ł}_{n+1}$ вытекает из критерия Мак-Нотона: сложение и умножение с переполнением сохраняют все множества $V(d)$, $d \setminus n$ (отметим, что $x\,[+]\,y = {\sim}x \to y$). Кроме того, очевидно, что $\mathrm{OŁ}_{n+1} \subseteq \mathrm{C}^{\{0\}}{}_{n+1}$. Для доказательства полноты множества $\mathrm{OŁ}_{n+1} \setminus \{[\times]\}$ в клоне $\mathrm{Ł}_{n+1} \cap \mathrm{C}^{\{0\}}{}_{n+1}$ рассмотрим произвольную k-местную функцию f из этого клона (считаем $k > 0$, добавляя, если нужно, фиктивные аргументы). Пусть $f_{[-]}$ – $(k+1)$-местная функция, полученная путем замены каждого вхождения n в разложение f в суперпозицию функций из базиса $\{n, [-]\}$ вхождением дополнительной $(k+1)$-й переменной. Имеем

$$f(x_1, \ldots, x_k) = f_{[-]}(x_1, \ldots, x_k,\ \overline{Z}(x_1\,[+]\,\ldots\,[+]\,x_k)),$$

где

$$\overline{Z} : x \mapsto x\,[+]\,\ldots\,[+]\,x\ (n \text{ раз}).$$

Этим доказана полнота множества функций $\mathrm{OŁ}_{n+1} \setminus \{[\times]\}$ в клоне $\mathrm{Ł}_{n+1} \cap \mathrm{C}^{\{0\}}{}_{n+1}$. Его предполнота в $\mathrm{Ł}_{n+1}$ следует из рассуждений, приведенных в доказательстве предложения 3.4 (поскольку $\mathrm{Sub}\,\langle E_{n+1}, [+], [-]\rangle \setminus \mathrm{Sub}\,\mathrm{Ł}_{n+1} = \{\{0\}\}$), а также непосредственно из того, что для всякой функции $f \in \mathrm{Ł}_{n+1} \setminus \mathrm{C}^{\{0\}}{}_{n+1}$ справедливо соотношение $f(x\,[-]\,x, \ldots, x\,[-]\,x) = n$, $x \in E_{n+1}$. □

Заметим, что множества функций $\mathrm{OŁ}_{n+1} \setminus \{[+]\}$ и $\mathrm{OŁ}_{n+1} \setminus \{[-]\}$ не предполны в $\mathrm{Ł}_{n+1}$, поскольку первое содержится в клоне $\mathrm{C}^{\{0,1\}}{}_{n+1}$, второе – в клоне $\mathrm{C}^{\{n\}}{}_{n+1}$, и множества $\{0, 1\}$ и $\{n\}$ не являются подалгебрами в $\mathrm{Ł}_{n+1}$.

В вычислительной практике арифметика с переполнением редко используется как полноценная самостоятельная модель вычислений. Операции с переполнением входят в арифмети-

ку порядков действительных чисел с плавающей точкой стандарта IEEE-754 [204]. Определенный интерес к арифметике с переполнением имеется в области разработки нетрадиционных многозначных арифметико-логических устройств (МЗАЛУ), основанных на представлении чисел уровнями значений таких физических величин, как электрический ток или частота светового излучения. Природа этих величин позволяет достаточно просто сконструировать преобразователи, реализующие операции алгебраической системы OA_{n+1}. Например, в [230] описан вычислитель ограниченной разности на двух токовых КМОП (current-mode CMOS) транзисторах, позволяющий построить полнофункциональное МЗАЛУ ввиду теоремы 3.5 (с учетом наличия шины с уровнем тока, соответствующим значению n). В работе [12] описано оптоэлектронное МЗАЛУ, реализующее алгебру Аллена-Живона, связки которой выражаются в логике Лукасевича. Существуют вычислительные устройства, аппаратно реализующие связки логики Лукасевича, например полимерные процессоры [224].

Приведем простой пример применения вышеизложенных конструкций при анализе эффективности потоковых (dataflow) вычислений [213]. К числу часто используемых алгоритмов реального времени относится фильтрация, задаваемая разностным уравнением

$$y_i = \sum_{l=0}^{N} b_l\, x_{i-l} - \sum_{m=1}^{M} a_m\, y_{i-m},$$

где x_i, $i = 0, 1, \ldots$ – поток числовых значений входного сигнала, y_i – выходной поток, b_l – коэффициенты входного сигнала, a_m – коэффициенты обратной связи. При наличии хотя бы одного ненулевого a_m вычислительный узел, реализующий этот алгоритм, называется БИХ-фильтром, или фильтром с бесконечной импульсной характеристикой [118, с.11]. В потоковом вычислителе, поддерживающем операции матрицы Лукасевича, он реализуется следующими функциями:

$$y(z_0, \ldots, z_{N+M}) = [(+)_{l\,=\,0,\,\ldots,\,N}\,(b_l * z_l)]\ (+)\ [(-)\,(+)_{m\,=\,1,\,\ldots,\,M}\,(a_m * z_{N+m})],$$

$$o(z_0, \ldots, z_{N+M}) = (\mathrm{Carry})([|+|_{l\,=\,0,\,\ldots,\,N}\,(b_l \triangleleft z_l)],\ n,\ (\mathrm{Carry})([|+|_{m\,=\,1,\,\ldots,\,M}\,(a_m \triangleleft z_{N+m})],\ n,\ 0)),$$

где через $k \triangleleft x$ обозначается сложение k экземпляров переменной x с переполнением, o – двузначный (булевый) сигнал переполнения. Для анализа корректности и сложности вычисления этих функций заданным устройством, операции арифметики по модулю и с переполнением выражаются в базисе клона алгебры, отвечающей его модели вычислений. Аналогичным образом анализируются схожие вычислительные задачи, часто возникающие при автоматизации управления в других технических предметных областях. Так, в разд.5.5 будет приведен пример расчета показателей объекта топливно-энергетического комплекса.

Результаты настоящего раздела позволяют предложить новую интерпретацию природы логики Лукасевича [71]. Напомним, что изначально она была создана для формализации рассуждений в условиях неопределенности и недостаточности информации [54]. Теперь можно значительно расширить область ее применения, поскольку она оказывается адекватным форма-

лизованным языком описания инструментария, предназначенного для решения вычислительных задач. Можно заключить, что она определяет формы и ограничения, лишь в рамках которых конечные информационные системы способны работать с таким инфинитным объектом, как множество целых чисел. Логика Лукасевича оказывается своего рода «дистиллятом» тех явлений информационной реальности, которые порождают в человеческом сознании брауэровскую «изначальную интуицию натурального числа».

В частности, трактовка выделенного истинностного значения n как флага означает, что класс тавтологий логики Лукасевича состоит из всех арифметических термов, значения которых не могут быть вычислены в машинной арифметике, поддерживающей n различных чисел. Этот факт открывает путь к использованию техники доказательств логики Лукасевича (в том числе автоматизированной [147]) в задачах верификации вычислительных программ на предмет отсутствия переполнения: выражение A не дает переполнения тогда и только тогда, когда $(\mathrm{Carry})(A, 0, n)$ – тавтология $Ł_{n+1}$. Отметим в связи с этим, что слабость логики как класса теорем оборачивается ее выразительной силой как класса числовых функций – с ростом n выражений с неопределимым значением становится все меньше, так что $Ł_{n+1}$ как класс теорем ослабевает (например, согласно условию Линденбаума [54], класс тавтологий логики $Ł_{kn+1}$ строго содержится в классе тавтологий логики $Ł_{n+1}$ при любом $k > 1$).

Глава 4. АСПЕКТНО-ОРИЕНТИРОВАННОЕ РАСШИРЕНИЕ МОДУЛЬНЫХ ТЕХНОЛОГИЙ ПРОЕКТИРОВАНИЯ

4.1 Семантика аспектно-ориентированного подхода

Рассмотрим семантику аспектно-ориентированного подхода, следуя [70]. Аспектно-ориентированное программирование (АОП) было создано Г.Кишалесом и его коллегами в конце 1990-х годов [208] для поддержки реализации классов задач, рассеивающихся по системе и пересекающих границы единиц модульной архитектуры (crosscutting concerns). Такие задачи глубоко погружены в контекст своего исполнения, поэтому их реализацию трудно оформлять единицами модульной архитектуры, контекст которых передается через фиксированный интерфейс («обобщенными процедурами», в терминологии Г.Кишалеса). Приходится дублировать программный код, реализующий рассеянные задачи, во всех местах обращения к ним. Дублирование кода не только чрезвычайно повышает трудоемкость модификации программы, но и мешает составлять логически корректные суждения о назначении и функциях ее фрагментов, необходимые для документирования [74]. Даже если искусственно оформить рассеянные задачи процедурами, то перемежающиеся обращения к ним, нагруженные передачей контекста, сделают текст программы трудным для понимания и сопровождения.

В рамках АОП предлагается оформлять реализации рассеянных задач в виде аспектов – особых программных единиц, места обращения к которым задаются в отдельной спецификации, внешней по отношению к вызывающей их («базовой») программе. Сборка системы из аспектов заключается во вставке их программного кода в базовый код в этих местах, в результате чего они получают полный доступ к контексту, обходя ограничения модульного интерфейса доступа. Эта процедура выходит за рамки традиционной компоновки модулей (linking), не позволяющей модифицировать поток исполнения программы, поэтому она называется связыванием (weaving). Совокупность мест обращения к аспектам в базовой программе называется срезом (pointcut), а совокупность вставляемых в них аспектов – советом (advice). Связывание может выполняться как на этапе компиляции, путем генерации программного кода со вставками согласно спецификации мест обращения к аспектам, так и в процессе исполнения, путем вызова предварительно скомпилированных аспектов при достижении соответствующих мест.

Этот подход отражен в самом известном определении АОП, данном Р. Филманом и Д. Фридманом [176]: aspect-oriented programming is quantification and obliviousness (аспектно-ориентированное программирование – это квантификация и забывчивость). Здесь квантификация означает возможность строить спецификацию связывания из конструкций, влияющих на

произвольное множество мест базовой программы (не объясняется, почему выбран такой термин: подразумевается ли аналогия с кванторными формулами исчисления предикатов, либо с квантовой механикой пространственно распределенных частиц, либо с чем-то еще). Забывчивость означает внешний характер спецификации связывания – отсутствие необходимости вносить в базовую программу какую-либо информацию о связываемых с ней аспектах. Проверяя наличие этих возможностей, авторы определения выявили степень правомерности отнесения к аспектно-ориентированным ряда технологий и инструментов программирования, таких как композиционные фильтры (composition filters [154]), AspectJ (аспектно-ориентированное расширение языка Java [166]), событийное программирование (event-based programming [222]), порождающее программирование (generative programming [130] – технология-предшественник разработки, управляемой моделями).

Однако это определение не содержит ответы на ряд важных методологических вопросов. Например, неясно, как выделяются возможные места обращения к аспектам (точки соединения, join points). В тексте программы они могут быть недвусмысленно заданы только на синтаксическом уровне – как точки, находящиеся непосредственно перед или после операторов определенного вида (присваивание, вызов процедуры, декларация переменной и т.п.). Для задания множеств таких точек можно дополнительно предусмотреть ограничения на наименования операндов, порядок следования операторов, даже на содержимое стека вызова процедур – и все же правила вызова аспектов диктуются организацией программы, а не ее предметной области. Этим вызвано удручающее однообразие типичных областей применения АОП: они не выходят за рамки программно-технических задач журналирования, кэширования и т.п. Для указания мест привязывания аспектов по предметно-ориентированным правилам можно добавить в текст базовой программы разнообразные аннотации [209], однако это нарушает принцип забывчивости, и большое количество аннотаций в базовой программе запутывает ее не меньше, чем перемешивание кода [254].

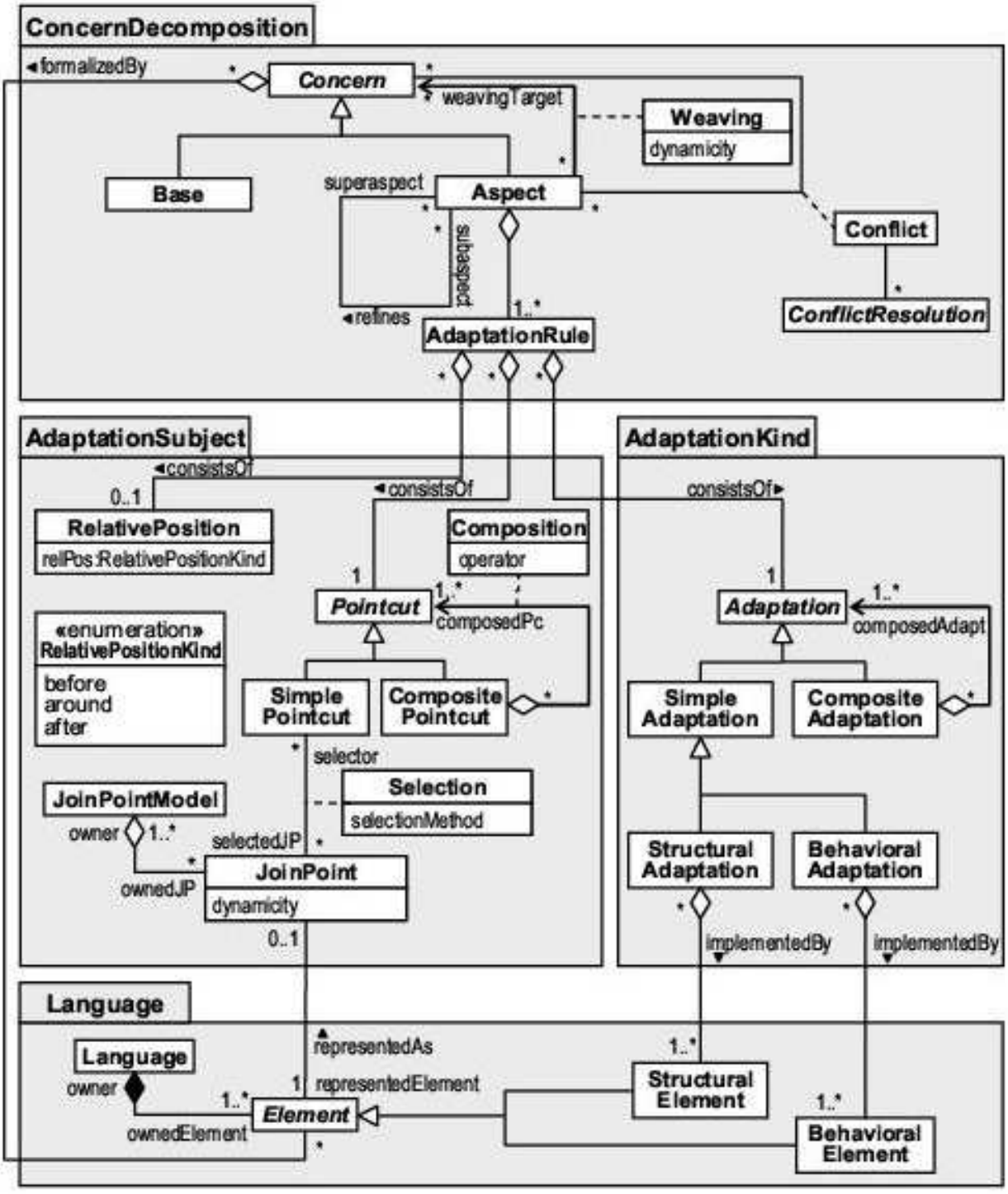
Неясными остаются и значительно более важные вопросы: откуда вообще берутся аспекты? каким фрагментам автоматизируемой предметной области следует сопоставлять аспекты, а каким – традиционные модули? как наиболее рационально расширить имеющуюся технологию разработки базовых программ, чтобы можно было задавать и связывать с ними аспекты? Для ответа на них необходимо выйти за рамки программирования и распространить аспектно-ориентированный подход на весь жизненный цикл системы – от формирования требований до сопровождения готового изделия [144]. Действительно, рассеяние задач присутствует практически во всех артефактах жизненного цикла: в характеристиках качества, спецификациях, моделях архитектуры, программном коде, массивах данных, тестах, электронных документах, экранных формах. Первопричиной этого является извечный конфликт между цельностью

внешнего мира и фрагментарностью целеполагания, присущей постигающему его человеческому сознанию. Так что аспекты естественным образом появляются в самом начале жизненного цикла, в виде классов задач (concerns), присутствующих одновременно во многих требованиях. Такие аспекты называются ранними (early aspects) [240]. По своей природе они вполне соответствуют значению слова «аспект» в обыденном языке: «(лат. aspectus – вид) точка зрения, с которой рассматривается какое-либо явление, понятие, перспектива» [19]. Для их идентификации и связывания применяются конструкции естественного языка и структурные элементы документов – параграфы, ссылки, таблицы. Например, в работе [163] предлагается строить таблицу композиции требований (Requirement Composition Table, RCT), в столбцах которой указываются базовые варианты использования проектируемой системы, а в строках – задачи, рассеивающиеся по ним. Ячейки таблицы образуют их матрицу инцидентности: если обращение к рассеянной задаче CC происходит в варианте использования UC, то в ячейке (CC, UC) указывается 1, а иначе – 0.

В этом подходе идентификация рассеянных задач происходит сугубо эвристически: аналитики строят таблицу композиции требований путем размышлений над описаниями вариантов использования. Предлагаются разнообразные подходы к частичной формализации и автоматизации этой деятельности: автоматический синтаксический разбор описаний требований [165], формальное представление требований в виде дерева целей, в котором аспекты сопоставляются частично достижимым целям (softgoals) [263] и др. К сожалению, эвристическая составляющая остается доминирующей: разные специалисты могут выделить в одной и той же системе совершенно разные наборы ранних аспектов. В аспектно-ориентированной инженерии требований не хватает единой базовой семантической модели [238]. Эта проблема распространяется и на дальнейшую разработку аспектов в ходе жизненного цикла системы, где они могут трансформироваться не только в «настоящие» аспекты (единицы связывания АОП), но и в обобщенные процедуры (объекты или функции), и в проектные решения (фрагменты документации, не выразимые непосредственно в программном коде) [240]. Современные технологии аспектно-ориентированной разработки программного обеспечения (aspect-oriented software development, AOSD) не подчинены единым методологическим правилам управления трансформациями аспектов – они слабо совместимы друг с другом на концептуальном уровне. Как отмечается в работе [247], они «имеют разное происхождение и преследуют различные цели при поддержании характерных возможностей аспектно-ориентированного подхода» (мы приводили эту цитату в разд.1.3).

В указанной работе делается попытка выделить ключевые понятия, присущие любой аспектно-ориентированной технологии, и унифицировать их в рамках единой базовой архитектуры (reference architecture) аспектно-ориентированного моделирования. За основу взята система

понятий классического АОП, поэтому предложенная архитектура наследует его главный недостаток – чисто синтаксический характер приемов аспектно-ориентированной сборки систем. В языке базовой модели выделяются элементы, к которым аспекты привязываются с указанием относительного положения (RelativePosition) – перед или после языковой конструкции (для графических языков моделирования, по-видимому, подразумевается более богатая система «синтаксических координат»). Предлагаемая базовая архитектура выглядит следующим образом [247].



Видно, что возможности привязки аспектов определяются допустимыми техническими манипуляциями с синтаксическими конструкциями базовых моделей, а не семантическими требованиями моделирования предметных областей. Это приводит к проблеме композиционной хрупкости (composition fragility) [238]: комплексная многоаспектная модель может радикально измениться и даже развалиться на части при незначительном изменении синтаксиса базы, хотя семантика остается неизменной. В то же время концептуальные вопросы аспектно-ориентированного подхода, сформулированные выше, по-прежнему остаются без ответа. Напрашивается вывод, что аспекты вообще не могут появляться в моделях предметных областей, будучи сущностями «второго порядка» над ними, выразимыми только посредством метамодельных (в частности, метапрограммных) конструкций [253].

В работе [239] делается попытка опровергнуть этот вывод многочисленными примерами из области разработки ранних аспектов. Однако чрезмерное разнообразие аспектно-ориентированных технологий снижает ценность приводимых аргументов. Ни один из примеров не способен претендовать на роль канона аспектно-ориентированного моделирования – они выглядят частными мнениями специалистов, моделирующих частные системы. Позитивным выводом работы является выделение фундаментальных «технологических» свойств, которыми должны обладать модели рассеянных классов задач: абстрактность (abstraction), модульность (modularity), комплексируемость (composability). Этот тезис безусловно справедливо характеризует цель применения аспектно-ориентированных технологий, однако ничего не дает для понимания способа ее достижения.

Методологический эффект от подходов наподобие предложенного в [247] трудно получить также из-за отсутствия правил обнаружения аналогов выделенных понятий в конкретных технологиях: неясно, как быть, если в рамках некоторой аспектно-ориентированной технологии вместо терминов «аспект» или «точка соединения» используются какие-либо другие. Традиционно для выполнения смысловой идентификации терминов независимо от их звучания, обеспечивающей однозначность понимания сути технологии всеми категориями ее потребителей, строится ее формальная семантика [78]. Аспектно-ориентированный подход и здесь отличается чрезмерным разнообразием частных решений и отсутствием общего канона. Его семантические модели строились с привлечением почти всех формализмов теоретического программирования: алгебры процессов [140], лямбда-исчисление [202], преобразования графов [260], проверка на моделях [206], теория инвариантов поведения программ [249], языки описания архитектуры [234] и др. Кроме того, предлагаются многочисленные аспектно-ориентированные расширения языка объектно-ориентированного моделирования UML [261 и др.] и формализованных языков моделирования процессов [161, 162]. Каждый из этих видов моделей имеет значительные кон-

цептуальные отличия от других и может применяться только в рамках совместимой с ним парадигмы программирования.

Здесь уместно провести аналогию с развитием методологической базы другой, значительно более зрелой парадигмы разработки программного обеспечения – объектно-ориентированного подхода. На этапе программирования объектно-ориентированное расширение процедурного языка заключается в предоставлении следующих двух главных возможностей. Во-первых, при декларации структур данных можно указывать в них не только поля, но и сигнатуры процедур (в результате чего у процедур появляется неявный дополнительный параметр – ссылка на эту структуру, обозначаемая ключевым словом `this`). Во-вторых, можно определять новые структуры как расширения уже определенных, указывая только отличия от них (в т.ч. можно подменять реализации процедур, декларированных в базовых структурах). Руководства по объектно-ориентированным языкам для программистов (например [120]) не дают целостного представления о том, в каких случаях следует декларировать такие процедуры и структуры, и каким образом они облегчают составление программ. Картина проясняется только при рассмотрении объектно-ориентированного подхода как полноценной парадигмы разработки программного обеспечения. Расширенные структуры данных возникают в результате особого способа анализа автоматизируемой предметной области – разбиения (декомпозиции) на локальные автономные подобласти (объекты), содержащие данные (атрибуты) и функции (методы), и связанные иерархическим классификационным отношением «абстрактное–конкретное» [20]. Описание этого подхода в терминах теоретического программирования приводит к компактному и в целом приемлемому определению: object-oriented programming is abstract data types and inheritance (объектно-ориентированное программирование – это абстрактные типы данных и наследование) [254]. Для записи результатов объектно-ориентированного анализа и проектирования в виде, не зависящем от выбора языка их программной реализации, разработан специализированный формализованный диаграммный язык UML (Universal Modeling Language [92 и др.]). Он позволил унифицировать на концептуальном уровне ключевые понятия объектного подхода и отношения между ними (класс, объект, наследование и т.д.).

На момент написания настоящей работы объектно-ориентированный подход остается доминирующей парадигмой разработки программного обеспечения. Программисты стремятся представить в виде объектов любые фрагменты предметной области. И здесь главным препятствием оказываются рассеянные задачи, не поддающиеся локализации в жестких рамках объектов. Так что появление аспектно-ориентированного подхода – вполне закономерный шаг в развитии инженерии программного обеспечения. Этот подход имеет важное методологическое отличие от других парадигм программирования: от него требуется не навязать собственные правила анализа предметной области, а предложить общие методы моделирования и реализации

рассеянных задач, затрудняющих декомпозицию каким-либо заранее выбранным способом. Он не нуждается в собственном языке для моделирования артефактов жизненного цикла: вместо него нужны единые правила рационального расширения существующих языков конструкциями, отражающими основные приемы поддержки аспектов – модуляризацию, идентификацию, связывание (ср. приведенный выше вывод работы [239]).

Таким образом, для обеспечения рационального использования аспектно-ориентированных технологий необходима универсальная семантическая модель, основанная на их назначении, а не на механизме связывания. Обращаясь к началу раздела, заметим, что одна из основных проблем, для решения которых было предложено АОП, состоит в стремлении аспектов «раствориться» в своем контексте из-за рассеяния. Поэтому основным назначением аспектно-ориентированного подхода в целом правомерно считать сохранение идентичности аспектов – оснащение моделей систем разметкой, идентифицирующей классы задач, на решение которых направлены составляющие их единицы. Технологии аспектно-ориентированной разработки отличаются способами разметки, но сходятся в стремлении обеспечить сквозную трассируемость, т.е. возможность однозначно установить класс задач, в целях решения которых в систему включена та или иная единица. Трассируемость страдает от перемешивания классов задач больше, чем другие показатели качества систем. Недостаточная поддержка трассирования, предоставляемая большинством распространенных инструментов моделирования и составления программ [205], является главным мотивом привлечения аспектно-ориентированных технологий [152, 238].

Например, в классическом АОП аспекты оформляются именованными блоками кода и проходят полный цикл разработки (включая разнообразные операции трассирования) отдельно от базовой программы. Квантификация и забывчивость являются «всего лишь» теми минимальными отступлениями от общих принципов традиционного программирования, которые позволяют реализовать такой режим обращения с аспектами. Другие примеры реализации аспектно-ориентированного подхода приведены в следующей таблице (Таблица 6). Конечно, их перечень далек от полноты, но во всяком случае они прошли процесс отчуждения от своих создателей и применяются во время написания настоящей работы.

Таблица 6

Технология	**Пример**	**Аспекты**	**Идентификация**	**Связывание**
Аспектно-ориентированное программирование	AspectJ [166]	Блоки кода	Именование блоков кода	Генерация перемешанного кода
Событийное программирование	RAPIDE [222]	Обработчики событий	Пометка (labeling) событий	Мониторинг событий с вызовом обработчиков
Порождающее программирование	openArchitectureWare [184]	Характеристики (features)	Именование характеристик	Генерация моделей (текстов и диаграмм)

Таблица 6 демонстрирует разнородность технологических контекстов, в рамках которых целесообразно привлекать аспектно-ориентированный подход. Приведенные в ней примеры приемов АОП требуют отражения в универсальной семантической модели, чтобы возникла концептуальная основа для их совместного использования в жизненном цикле систем. В основе семантики, разработанной нами, как указывалось в разд.1.3, лежит наиболее прямой и экономный способ обеспечения трассируемости – «запоминание» трансформаций вместе с моделями, порожденными ими из классов задач. Поскольку перемешивание задач критическим образом затрудняет интеграцию компонентов в системы, достаточно запоминать действие трансформаций на уровне интерфейсов, описывающих их возможности интегрироваться в модели систем. Тогда оно выглядит как дополнительная структура модели – разметка классами задач. Разнообразные помеченные структуры часто фигурируют в литературе в качестве формальных моделей программ (см. например [245]), однако назначение меток и способы их синтеза обычно не рассматриваются. В условиях применения аспектно-ориентированного подхода разметка служит для идентификации аспектов путем трассирования – каждому классу задач соответствует реализующий его аспект. Поэтому совокупность классов задач, образующая интерфейс источника трансформации, породившей модель, представляет собой ее аспектную структуру. В результате разметки в модели появляется явный ответ на вопрос, откуда взялись составляющие ее аспекты.

Аспектно-ориентированные приемы описываются как расширения модульных приемов комплексирования моделей, согласованные с их разметкой классами задач. Например, связывание совета с базой представляет собой их соединение (в смысле разд.2.1) со спецификацией мест связывания в роли «клея», выполняемое по одной и той же структурной схеме на уровне модульных основ и аспектных структур связываемых моделей. Другим важным приемом служит выявление моделей, которые ведут себя в актах интеграции как единицы модульной архитектуры, что позволяет оперировать с ними без привлечения специфических аспектно-ориентированных средств. В качестве канонического критерия, выделяющего такую модель, нами рассматривается возможность «поднять» трансформацию, породившую ее из классов ее задач, с уровня ее интерфейса на уровень модульной основы. Действительно, путем трассирования классов задач вдоль такой трансформации, как правило, можно извлечь из модели составляющие ее аспекты в виде модульных единиц, т.е. произвести на модульном уровне полное разделение ответственности (separation of concerns). Поэтому такая трансформация названа нами экспликацией аспектной структуры. Наша семантика аспектно-ориентированного подхода позволила предложить новый способ разделения ответственности при помощи экспликации, который (насколько нам известно) не был описан в литературе.

Для формализации и верификации нашей семантики мы привлекаем аппарат теории категорий, руководствуясь соображениями, обстоятельно изложенными в гл.2. За основу взята

формальная технология проектирования систем – категорная конструкция, введенная в разд.2.3, в рамках которой описываются основные шаги процесса создания систем: комплексирование, выделение интерфейсов, трансформация. Добавление поддержки аспектов в технологию описывается нами как формальное преобразование в расширенную технологию, главный эффект которого состоит в появлении специального функтора, выделяющего аспектную структуру из размеченных моделей артефактов. Аспектно-ориентированные приемы формализуются универсальными категорными конструкциями – разнообразными пределами и копределами, естественными относительно действия этого функтора.

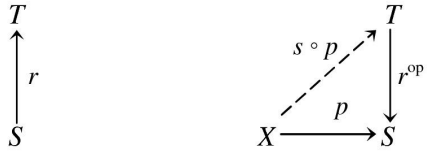
В качестве приложения нашей семантики в гл.5 построена и теоретически обоснована концепция технологии совместного аспектно-ориентированного проектирования моделей данных и сценариев исполнения процессов – ключевых составляющих информационно-управляющих систем. Также приводится нестандартное приложение, составляющее краеугольный камень разработки, управляемой моделями (MDE) – синтез специализированых технологий специфицирования систем (разд.4.5). Показано, что функтор выделения интеграционных интерфейсов, дополняющий технологию конфигурирования систем до технологии специфицирования, представляет собой в точности эксплицируемую разметку ее формальной модели. Конструкция аспектно-ориентированного расширения модульных технологий проектирования совместима с аспектной структурой, порождаемой интерфейсами, в следующем смысле. Если в некоторой технологии проектирования сборка систем не разрушает аспектную структуру, то ее аспектно-ориентированное расширение на уровне формальных технологий специфицирования представляет собой эксплицируемую трансформацию.

4.2 Формальные технологии аспектно-ориентированного проектирования

Основой для формализации аспектно-ориентированного подхода служит семантика трассирования, которую в настоящей работе мы строим из конструкций в формальных технологиях проектирования. Зафиксируем произвольную формальную технологию проектирования $AR = \langle c\text{-}DESC, Conf, sig, r\text{-}DESC \rangle$, обозначим через ε коединицу сопряжения $sig^* \dashv sig$. Ясно, что трассируемость сильно нарушается при трансформациях (хрестоматийным примером является реализация алгебраической спецификации программы на алгоритмическом языке программирования). При интеграции, напротив, обычно обеспечивается хотя бы частичное трассирование (здесь примером служит прямая сумма множеств в категории **Set**, при построении которой элементы каждой компоненты снабжаются ее меткой). Поэтому возможность трассирования результата трансформации $r : S \to T$ к источнику (в направлении от T к S) означает, что

обращение ее направления, т.е. теоретико-категорная дуализация, превращает ее в действие по интеграции – в *c-DESC*-морфизм $r^{\mathrm{op}} : T \to S$ (ср. разд.2.5).

Возможность совместного трассирования трансформации и сборки систем наиболее прозрачным образом обеспечивается, если трасса r^{op} обратима справа. Действительно, существование *c-DESC*-морфизма $s : S \to T$ такого, что $r^{\mathrm{op}} \circ s = 1_S$, эквивалентно существованию для любого *c-DESC*-морфизма $p : X \to S$ действия по интеграции X в T, совместимого с трассированием трансформации r в том смысле, что композиция трассы r^{op} с этим действием дает p: таким действием служит $s \circ p$, поскольку $r^{\mathrm{op}} \circ (s \circ p) = p$.



В свою очередь, для любого *c-DESC*-морфизма $q : T \to Y$ морфизм $q \circ s$ описывает действие по интеграции S в Y, совместимое с трассированием в несколько ином смысле: выбор $q = r^{\mathrm{op}}$ превращает его в тождественный морфизм. Заметим, что s является регулярным мономорфизмом [136, предложение 7.59(1)] – теоретико-категорным аналогом операции включения источника трансформации в результат, не разрушающей его структуру (ср. понятие неразрушающего включения, введенное в разд.2.4). На практике его построение может быть довольно трудоемким, но он требуется не всегда, поскольку трассированию вдоль процессов интеграции систем подлежат в первую очередь интеграционные требования, предъявляемые к интерфейсам моделей. Как указывалось в разд.1.3, следуя подходу к трассируемости, основанному на значимости (value-based requirements traceability) [171], достаточно потребовать обратимости справа для *SIG*-морфизма $sig(r^{\mathrm{op}})$, представляющего действие трассы на уровне интерфейсов. Реализация его обращения обычно не требует значительных затрат, поскольку интерфейсы проектируются так, чтобы интегрировать их было «проще», чем сами модели.

Такой подход можно кратко выразить в терминах модусов (вариантов использования) отношения «часть–целое», выделение которых согласно [131] восходит к Аристотелю. Трассируемая трансформация – это обращение частно-видового отношения, порождающее частно-родовое отношение между интерфейсами. Отсюда видно, что ввиду условия (vii) определения 2.3 тривиальным примером трассируемой (причем обратимой) трансформации является *c-DESC*-изоморфизм (напомним, что морфизм, двойственный к изоморфизму, отождествляется с обратным к нему и также является изоморфизмом [136]). Нетривиальные примеры возникают следующим образом. Рассмотрим совокупность всех общих подкатегорий в *c-DESC* и *r-DESC*$^{\mathrm{op}}$, содержащих все *c-DESC*-изоморфизмы. Она является полной нижней полурешеткой по включению. Обозначим через *cr-DESC* пересечение всех ее максимальных элементов.

Определение 4.1. *cr-DESC*-морфизм t называется (*sig*-)*трассой*, если *SIG*-морфизм *sig*(t) является ретракцией (т.е. имеет правый обратный). *sig*-образ трассы называется (*sig*-)*разметкой*. *r-DESC*-морфизм, двойственный к трассе, называется *трассируемой трансформацией*. Трансформация называется *обратимой*, если она трассируема и двойственна к *c-DESC*-ретракции. Формальная технология *AR поддерживает трассирование*, если все трансформации в ней обратимы. □

Обозначим через *tr-DESC* пару, состоящую из класса всех *c-DESC*-объектов и класса всех трасс.

Предложение 4.1. *tr-DESC* является подкатегорией в *c-DESC*, обладающей следующими свойствами.

(i) Iso *c-DESC* $\subseteq$ Mor *tr-DESC* $\subseteq$ Epi *c-DESC*.

(ii) Iso *SIG* $\subseteq$ *sig*(Mor *tr-DESC*) $\subseteq$ Retract *SIG*.

Доказательство. Любая ретракция является эпиморфизмом, а любой унивалентный функтор отражает эпиморфизмы. □

В формальной технологии над **Set** разметки являются сюръективными отображениями, поскольку в силу аксиомы выбора любой **Set**-эпиморфизм является ретракцией. Как указывалось в конце разд.2.5, здесь действие трансформации $t^{\mathrm{op}} : X \to Y$ можно описать как раскрытие (expansion) всех точек множества $|X|$, обозначающих классы задач, в их реализации – прообразы относительно t. Примером служит трансформация сценариев, разбивающая свой результат на подмножества, хорошо упорядоченные (well-ordered) в том смысле, что для любых $x, y, z, u \in Y$ таких, что $t(x) = t(y) \neq t(z) = t(u)$, условие $x \leq z$ влечет $y \leq u$ (см. [77]). Класс всех разметок сценариев состоит из всех сюръективных отображений множеств. Технология *SM* поддерживает трассирование.

Желательно иметь универсальный источник трансформаций, из которого можно получать нетривиальные модели, применяя трассируемые трансформации, определяемые своими результатами однозначно. Такой источник представляет собой класс всех возможных задач, рассматриваемый как одно целое, без какой-либо внутренней структуры. Любая модель может быть формально интегрирована в него, причем единственным способом – путем полного «сокрытия» своей структуры. Поэтому его формальным аналогом служит терминальный *c-DESC*-объект (при наличии подходящих трансформаций). В свою очередь, морфизм вида $e : \mathbf{1} \to S$, где $\mathbf{1}$ – терминальный объект, задает элемент модели S (см. [30]). В технологии над **Set** терминальный объект обычно обладает одноточечным основным множеством, отвечающим атомарному классу задач, и при интеграции в любую модель он становится ее атомарным фрагментом – точкой основного множества. Например, в технологии моделирования сценариев *SM* из него можно получить путем трансформации любой непустой сценарий.

Определение 4.2. Формальная технология *AR* называется *элементарной*, если категория *c-DESC* обладает терминальным объектом. □

Семантика трассирования подсказывает способ расширения модульной технологии проектирования, приводящий к достижению цели аспектно-ориентированного подхода. А именно, нужно обогащать модели трассируемыми трансформациями, порождающими их из классов задач. Поскольку в контексте аспектно-ориентированного подхода интерес представляет в первую очередь влияние трансформаций на интеграционные возможности моделей, достаточно присоединить к моделям действия трансформаций на уровне интерфейсов, т.е. в точности разметки. (В разд.4.4 мы увидим, что возможность «поднятия» трансформации на уровень моделей является основным условием модуляризации классов задач – представления аспектов модулями). Интеграция и трансформация таких обогащенных моделей должна согласованно выполняться на двух уровнях: на уровне их модульных основ и аспектных разметок. Как указано в [180, разд.7], в теории категорий имеется специальная конструкция, предназначенная для естественного присоединения действий к объектам – категория запятой (см. разд.2.2). За основу мы возьмем категорию запятой $sig \downarrow SIG$ (строго говоря, $sig \downarrow 1_{SIG}$). Ее объектами являются все пары вида $\langle A, l : sig(A) \to L\rangle$, где $A \in \mathrm{Ob}\ c\text{-}DESC$ и $l \in \mathrm{Mor}\ SIG$. Морфизмом объекта $\langle A_1, l_1 : sig(A_1) \to L_1\rangle$ в $\langle A_2, l_2 : sig(A_2) \to L_2\rangle$ является любая пара $\langle f : A_1 \to A_2,\ b : L_1 \to L_2\rangle$ такая, что $b \circ l_1 = l_2 \circ sig(f)$.

$$\begin{array}{lccc} \langle A_1, & sig(A_1) & \xrightarrow{\ l_1\ } & L_1\rangle \\ f\downarrow & sig(f)\downarrow & & \downarrow b \\ \langle A_2, & sig(A_2) & \xrightarrow{\ l_2\ } & L_2\rangle \end{array}$$

Определение 4.3. *Аспектно-ориентированной моделью* (АО-моделью) называется любой $(sig \downarrow SIG)$-объект $\langle A, l : sig(A) \to L\rangle$ такой, что l является sig-разметкой. *c-DESC*-объект A называется (*модульной*) *основой* АО-модели, *SIG*-морфизм l – ее (*аспектной*) *разметкой*, *SIG*-объект L – ее *аспектной структурой*. □

Будем обозначать через *AO* полную подкатегорию в $sig \downarrow SIG$, класс объектов которой состоит из всех АО-моделей. Поясним эту конструкцию на примере формальной технологии над **Set**. Напомним, что здесь разметка l – сюръективное отображение, сопоставляющее каждой точке множества $|A|$ элемент множества L, обозначающий класс задач, для решения которого она предназначена. В частности, аспектом естественно называть любую АО-модель, в которой L состоит из одного элемента (см. предложение 4.6). По существу (с точностью до изоморфизма), разметка является отношением эквивалентности на множестве $|A|$, классы эквивалентности которого отвечают отдельным аспектам. Любое отношение эквивалентности превращает A в

корректную АО-модель, так что аспектная структура в общем случае никак не связана с модульной. *AO*-морфизмом является в точности любой *c-DESC*-морфизм, сохраняющий это отношение эквивалентности. Поскольку канонический забывающий функтор $|\text{–}| : \mathbf{Equ} \to \mathbf{Set}$, где **Equ** – категория всех множеств с отношением эквивалентности и всех их гомоморфизмов, является топологическим, *AO* во многих случаях эквивалентна топологической категории над *c-DESC*. Кроме того, как мы увидим далее, *AO* может выступать в качестве категории моделей подходящей формальной технологии над **Set**. Например, в технологии моделирования сценариев *SM* АО-моделью является в точности любое частично-упорядоченное мультимножество (pomset) аспектов. Такой подход к моделированию сценариев был предложен еще в 1980-х годах [236], однако природа меток и способы их синтеза оставались неясными, поскольку они не рассматривались в контексте АОП.

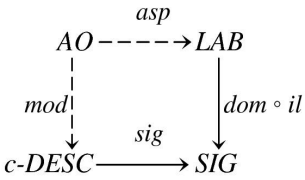
Вернемся к рассмотрению произвольных формальных технологий. Как указывалось в разд.2.2, категория запятой – сложная конструкция, составляющие которой можно извлекать посредством подходящих «забывающих» функторов. При построении некоторых из них используется категория стрелок $C^{\mathbf{2}}$, где C – произвольная категория. Напомним, что класс объектов категории $C^{\mathbf{2}}$ состоит из всех C-морфизмов (диаграмм вида $\mathbf{2} \to C$), а морфизмом из f в g является любая пара C-морфизмов $\langle u, v\rangle$ такая, что $v \circ f = g \circ u$. Контравариантный функтор $C^- : \mathbf{CAT}^{\mathrm{op}} \to \mathbf{CAT} : D \mapsto C^D$ сопоставляет постоянным функторам $0, 1 : \mathbf{1} \hookrightarrow \mathbf{2}$ функторы $dom, codom : C^{\mathbf{2}} \to C$, переводящие любой C-морфизм в его область и кообласть, соответственно. Эти функторы являются правым сопряженным (с тождественной коединицей) и левым сопряженным (с тождественной единицей) к полному вложению $1_- : C \hookrightarrow C^{\mathbf{2}} : A \mapsto 1_A, f \mapsto \langle f, f\rangle$, соответственно. Любую диаграмму $\Delta : X \to C^{\mathbf{2}}$ стандартный изоморфизм $(C^{\mathbf{2}})^X \cong (C^X)^{\mathbf{2}}$ взаимно однозначно переводит в естественное преобразование $\beta^\Delta : dom \circ \Delta \to codom \circ \Delta \in \mathrm{Mor}\, C^X$ с компонентами $\beta^\Delta_I = \Delta(I)$, $I \in X$. Если диаграммы $dom \circ \Delta$ и $codom \circ \Delta$ имеют копределы, то коконус $\langle \mathrm{colim}\, dom \circ \Delta, \mathrm{colim}\, codom \circ \Delta\rangle : \Delta \to \ulcorner colim(\langle \beta^\Delta, 1_X\rangle)\urcorner$ является копределом диаграммы Δ. Двойственно, если диаграммы $dom \circ \Delta$ и $codom \circ \Delta$ имеют пределы, то конус $\langle \lim dom \circ \Delta, \lim codom \circ \Delta\rangle : \ulcorner lim(\langle \beta^\Delta, 1_X\rangle)\urcorner \to \Delta$ является пределом диаграммы Δ. Следовательно, функтор $\langle dom, codom\rangle : C^{\mathbf{2}} \to C \times C$ поднимает пределы и копределы всех $C^{\mathbf{2}}$-диаграмм.

Пусть *LAB* – полная подкатегория в $SIG^{\mathbf{2}}$, класс объектов которой состоит из всех *sig*-разметок, $il : LAB \hookrightarrow SIG^{\mathbf{2}}$ – вложение. Мы будем использовать следующие «забывающие» функторы, связанные с категорией $sig \downarrow SIG$:

- $mod : AO \to c\text{-}DESC : \langle A, l\rangle \mapsto A, \langle f, b\rangle \mapsto f$;
- $asp : AO \to LAB : \langle A, l\rangle \mapsto l, \langle f, b\rangle \mapsto \langle sig(f), b\rangle$;

- $int = sig \circ mod = dom \circ il \circ asp : AO \to SIG : \langle A, l\rangle \mapsto sig(A), \langle f, b\rangle \mapsto sig(f)$;
- $str = codom \circ il \circ asp : AO \to SIG : \langle A, l\rangle \mapsto \mathrm{codom}\ l, \langle f, b\rangle \mapsto b$.

Мы будем называть *фундаментальными* функторы 1_{AO}, *mod*, *asp* и *int*. С их помощью можно построить *AO* как универсальную конструкцию в категории **CAT** (ср. диаграмму, определяющую категорию запятой, из разд.2.2): непосредственно проверяется, что равенство $sig \circ mod = (dom \circ il) \circ asp$, определяющее функтор *int*, задает декартов квадрат (предел) пары функторов $sig : c\text{-}DESC \to SIG \leftarrow LAB : dom \circ il$, обладающий вершиной *AO*. Следовательно, обогащение моделей аспектными разметками можно выполнить единственным образом (с точностью до изоморфизма).



Фундаментальные функторы имеют большое значение для формализации аспектно-ориентированного проектирования: ниже мы докажем (теорема 4.1), что они позволяют извлекать из АО-моделей различные интеграционные интерфейсы в смысле определения 2.3, т.е. порождают формальные технологии проектирования. Различные интерфейсы требуются для выработки различных видов проектных решений: модульный интерфейс, извлекаемый функтором *mod*, играет ключевую роль при модуляризации аспектов (см. разд.4.4), аспектный, извлекаемый функтором *asp* – при синтезе разметок, исходный, извлекаемый функтором *int* – при специфицировании интеграционных требований к моделям без детализации их модульной или аспектной структуры. В общем случае возможны и другие виды интерфейсов, «уточняющие» исходные интерфейсы из категории *SIG*, т.е. естественным образом отображающиеся в них (порождая трансформацию на уровне формальных технологий специфицирования интерфейсов – см. следствие 4.4.6).

В то же время, как мы увидим далее (предложение 4.2), аспектная структура, выделяемая функтором *str*, не может служить интерфейсом для нетривиальных АО-моделей, а в некоторых случаях выступает в качестве их дискретной структуры (предложение 4.3). Функтор *str* формально выражает «квинтэссенцию» аспектно-ориентированного расширения технологий проектирования систем, не сводимую к понятиям модульного подхода. Примечательно, что он сюръективен, т.е. любая аспектная структура реализуется в подходящей АО-модели (следствие 4.1.5). Как мы увидим в разд.4.3, для формального определения ключевых понятий аспекта и связывания требуется только этот функтор. Также отметим, что функтор *asp* индуцирует естественное преобразование $\beta^{il \circ asp} : int \to str$, сопоставляющее любой АО-модели ее разметку.

Зададим правила построения конфигураций и трансформаций АО-моделей из модульного «материала». Трансформации естественным образом получаются как двойственные к трассам (нетрассируемые трансформации технологии *AR* полностью игнорируются): положим

$$tr\text{-}AO = (\mathrm{Ob}\ AO,\ mod^{-1}(\mathrm{Mor}\ tr\text{-}DESC)).$$

В силу предложения 4.1 *tr-AO* является категорией.

В качестве конфигураций АО-моделей целесообразно выбрать *AO*-диаграммы, копределы которых согласованно вычисляются на уровне модульных основ и аспектных структур, и которые расширяют модульные конфигурации. Дополнительно нужно обеспечить свойство кооднородности, поскольку оно гарантирует неразрушаемость при покомпонентных трансформациях систем (разд.2.5).

Определение 4.4. *AO*-диаграмма Δ называется *аспектно детерминированной*, если она обладает копределом и функтор $\langle mod, str\rangle : AO \to c\text{-}DESC \times SIG$ детерминирует ее копределы. Пусть *ai* – произвольный функтор, действующий из *AO* в произвольную категорию *INT*. Класс *INT*-диаграмм *AIDia* называется *аспектно замкнутым* (относительно *ai*), если класс *AO*-диаграмм $\mathbf{D}ai^{-1}(AIDia)$ состоит из аспектно детерминированных диаграмм, содержится в классе $\mathbf{D}mod^{-1}(Conf)$ и замкнут относительно накачек *tr-AO*-морфизмами. □

Хотя бы один аспектно замкнутый класс существует для любого функтора *ai* – это пустое множество. Будем обозначать через $AOInt_{ai}$ объединение всех аспектно замкнутых классов *INT*-диаграмм. Ясно, что оно также является аспектно замкнутым классом.

Определение 4.5. Функтор *ai* с областью *AO* называется *порождающим аспектно-ориентированную формальную технологию* (АО-технологию) над *AR*, если четверка

$$AO_{ai}(AR) = \langle AO,\ \mathbf{D}ai^{-1}(AOInt_{ai}),\ ai,\ tr\text{-}AO^{\mathrm{op}}\rangle$$

является формальной технологией проектирования и существует функтор *si* такой, что $si \circ ai = int$. □

Теорема 4.1. Любой фундаментальный функтор порождает АО-технологию над *AR*.

Доказательство. Условия (i), (v)-(viii) определения 2.3 выполняются для четверки $AO_{ai}(AR)$ при любом выборе функтора *ai* с областью *AO* (в частности, любая АО-технология над *AR* кооднородна). Проверим условия (ii)-(iv) для функторов *mod* и *asp*: для функтора 1_{AO} они выполняются ввиду предложения 2.12, а для функтора *int* – ввиду их устойчивости относительно композиции функторов (ср. конструкцию трансформации в технологии синтеза технологий конфигурирования ***SCONF***).

(ii). Функтор $dom \circ il$ унивалентен: если $\langle f, b\rangle, \langle f, b'\rangle : l \to k$ – два произвольных *LAB*-морфизма, то $b' \circ l = k \circ f = b \circ l$, откуда $b' = b$, поскольку l обратим справа. Поэтому функтор *mod*, будучи ребром декартова квадрата в **CAT**, параллельным $dom \circ il$, унивалентен. Аналогично, функтор *asp*, параллельный унивалентному функтору *sig*, унивалентен.

(iii). Функтор дискретной разметки

$$mod^* : c\text{-}DESC \to AO : A \mapsto \langle A, 1_{sig(A)}\rangle, f \mapsto \langle f, sig(f)\rangle$$

является левым сопряженным к *mod* с тождественной единицей и коединицей $\langle 1_A, l\rangle : \langle A, 1_{sig(A)}\rangle \to \langle A, l\rangle, \langle A, l\rangle \in \mathrm{Ob}\, AO$. Функтор

$$asp^* : LAB \to AO : l \mapsto \langle sig^*(\mathrm{dom}\, l), l\rangle, \langle p, q\rangle \mapsto \langle sig^*(p), q\rangle$$

является левым сопряженным к *asp* с тождественной единицей и коединицей $\langle \varepsilon_A, 1_{\mathrm{codom}\, l}\rangle : \langle sig^*(sig(A)), l\rangle \to \langle A, l\rangle, \langle A, l\rangle \in \mathrm{Ob}\, AO$.

(iv). Покажем, что функтор *mod* поднимает копределы всех аспектно детерминированных *AO*-диаграмм. Выберем произвольно аспектно детерминированную диаграмму Ξ и копредел ρ диаграммы $mod \circ \Xi$. Согласно определению 4.4, *SIG*-диаграмма $str \circ \Xi$ обладает копределом, и диаграмма Ξ имеет копредел ξ, удовлетворяющий условиям $mod \circ \xi = \rho$ и $str \circ \xi = \mathrm{colim}\, str \circ \Delta$.

Чтобы показать, что функтор *asp* поднимает копределы всех конфигураций, рассмотрим произвольную диаграмму $\Delta \in \mathbf{D}asp^{-1}(AOInt_{asp})$ и копредел $\sigma : asp \circ \Delta \to \ulcorner k \urcorner$. Выберем произвольно копредел $\delta : \Delta \to \langle A, l\rangle$. Ввиду определения 4.4 $str \circ \delta$ – копредел диаграммы $str \circ \Delta$ и $mod \circ \delta$ – копредел диаграммы $mod \circ \Delta$, а поскольку согласно предложению 2.6 функтор *sig* сохраняет копределы конфигураций, $int \circ \delta$ – копредел диаграммы $int \circ \Delta$. Поэтому l – объект копредела диаграммы $il \circ asp \circ \Delta$ в категории стрелок SIG^2, а поскольку l – *sig*-разметка, имеется копредел $asp \circ \delta : asp \circ \Delta \to \ulcorner I \urcorner$. Поскольку копредел определен однозначно с точностью до изоморфизма, существует *LAB*-изоморфизм i такой, что $\sigma = \ulcorner i \urcorner \circ (asp \circ \delta)$, поэтому $dom \circ il \circ \sigma = \ulcorner dom(il(i)) \urcorner \circ (int \circ \delta)$ – копредел диаграммы $int \circ \Delta$. А поскольку функтор *sig* поднимает копределы конфигураций, существует копредел $\theta : mod \circ \Delta \to \ulcorner B \urcorner$ такой, что $sig \circ \theta = dom \circ il \circ \sigma$. Тогда $\langle \theta, codom \circ il \circ \sigma\rangle : \Delta \to \ulcorner \langle B, k\rangle \urcorner$ – копредел. □

Следствие 4.1.1. Произвольный функтор $ai : AO \to INT$ порождает AO-технологию над *AR* тогда и только тогда, когда он удовлетворяет следующим условиям.

(i) Функтор *ai* обладает левым сопряженным с тождественной единицей.

(ii) $int \circ ai^* \circ ai = int$, где ai^* – левый сопряженный из условия (i).

(iii) Любой аспектно замкнутый класс *INT*-диаграмм состоит из *ai*-предконфигураций.

Доказательство. Необходимость вытекает из определения 4.5 с учетом того, что если *si* – функтор, удовлетворяющий условию $si \circ ai = int$, то $si = si \circ (ai \circ ai^*) = int \circ ai^*$. Достаточность вытекает из того, что условия (i), (v)-(viii) определения 2.3 выполняются при любом выборе функтора *ai* (ср. доказательство теоремы 4.1) а условие (ii) гарантируется унивалентностью

функтора *int*, поскольку функтор *ai* является первой компонентой его разложения в композицию (см. [136, предложение 3.30(2)]). □

Следствие 4.1.2. Формальная технология $AO_{ai}(AR)$ элементарна тогда и только тогда, когда *AR* элементарна.

Доказательство. Необходимость вытекает из наличия левого сопряженного *mod** к функтору *mod* (ввиду чего последний переводит терминальный *AO*-объект в терминальный *c-DESC*-объект), а достаточность – из непосредственно проверяемого факта, что *mod** в свою очередь переводит терминальный *c-DESC*-объект в терминальный *AO*-объект. □

Следствие 4.1.3. Любая конфигурация технологии $AO_{int}(AR)$ является конфигурацией в любой AO-технологии над *AR*. В свою очередь, любая конфигурация любой AO-технологии над *AR* является конфигурацией в $AO_{1_{AO}}(AR)$.

Доказательство. Пусть $ai : AO \to INT$ – произвольный функтор, порождающий AO-технологию над *AR*, $si : INT \to SIG$ – функтор, удовлетворяющий условию $si \circ ai = int$. Выберем произвольно *AO*-диаграмму $\Delta \in \mathbf{D}int^{-1}(AOInt_{int})$, положим $\Theta = int \circ \Delta$, $\Xi = ai \circ \Delta$. Любая *AO*-диаграмма $\Delta' \in \mathbf{D}ai^{-1}(\{\Xi\})$ аспектно детерминирована и содержится в классе $\mathbf{D}mod^{-1}(Conf)$, поскольку $int \circ \Delta' = si \circ \Xi = \Theta \in AOInt_{int}$. Обозначим через $ai \rightrightarrows \Omega$ класс *ai*-образов всех накачек *AO*-диаграммы Ω *tr-AO*-морфизмами. Любая *AO*-диаграмма $\Delta'' \in \mathbf{D}ai^{-1}(ai \rightrightarrows \Delta')$ также аспектно детерминирована и содержится в $\mathbf{D}mod^{-1}(Conf)$, поскольку $int \circ \Delta'' \in si \circ (ai \rightrightarrows \Delta') \subseteq int \rightrightarrows \Delta' \subseteq AOInt_{int}$. Рассуждая по индукции, определим последовательность классов *INT*-диаграмм *PDia*, полагая $PDia_0 = \{\Xi\}$, $PDia_{n+1} = \bigcup_{\Omega \in \mathbf{D}ai^{-1}(PDia_n)} ai \rightrightarrows \Omega$. Класс $\bigcup_{n \geq 0} PDia_n$ аспектно замкнут, а поскольку он содержит Ξ, имеем $\Delta \in \mathbf{D}ai^{-1}(AOInt_{ai})$. Первое утверждение доказано.

Второе утверждение вытекает непосредственно из того, что класс $\mathbf{D}ai^{-1}(AOInt_{ai})$ аспектно замкнут относительно функтора 1_{AO}. □

Следствие 4.1.4. Класс Mor *tr-AO* состоит из всех трасс технологии $AO_{int}(AR)$. □

Следствие 4.1.5. Функтор *str* обратим справа.

Доказательство. Функтор $1_{\_} : SIG \hookrightarrow LAB : A \mapsto 1_A$ является правым обратным к функтору $codom \circ il$, а функтор *asp* обратим справа по теореме 4.1. □

Следствие 4.1.6. Категория *AO* вместе с любым фундаментальным функтором образует подвижную конкретную категорию тогда и только тогда, когда *c-DESC* подвижна.

Доказательство. Унивалентность фундаментальных функторов вытекает из теоремы 4.1. Как и в ее доказательстве, для проверки достаточности условия нужно установить подвижность конкретных категорий (*AO*, *mod*) и (*AO*, *asp*), поскольку свойство подвижности тривиаль-

ным образом выполняется для тождественного функтора и устойчиво относительно композиции функторов.

Проверим подвижность категории (*AO*, *mod*). Пусть $\langle A, l : sig(A) \to L\rangle \in \mathrm{Ob}\, AO$, $i : A \to X \in \mathrm{Iso}\, c\text{-}DESC$. Положим $\hat{\imath} = sig(i)$. Пусть $t : U \to V$ – трасса, удовлетворяющая условию $sig(t) = l$. Из него вытекает, что $sig(U) = sig(A)$, так что по условию существует *c-DESC*-изоморфизм $j : U \to Y$ такой, что $sig(j) = \hat{\imath}$. В силу предложения 4.1 морфизм $t \circ j^{-1}$ является трассой, причем $sig(t \circ j^{-1}) = l \circ \hat{\imath}^{-1}$. Поэтому имеется *AO*-изоморфизм $\langle i, 1_L\rangle : \langle A, l\rangle \to \langle X, l \circ \hat{\imath}^{-1}\rangle$.

Подвижность конкретной категории (*AO*, *asp*) проверяется проще. Пусть $\langle A, l\rangle \in \mathrm{Ob}\, AO$, $\langle i, i'\rangle : l \to k \in \mathrm{Iso}\, LAB$. Поскольку $i \in \mathrm{Iso}\, SIG$, по условию существует *c-DESC*-изоморфизм $j : A \to Y$ такой, что $sig(j) = i$. Поэтому имеется *AO*-изоморфизм $\langle j, i'\rangle : \langle A, l\rangle \to \langle Y, k\rangle$.

Осталось проверить необходимость условия. Предположим, что категория *AO* вместе с любым фундаментальным функтором, в частности пара (*AO*, *int*), подвижна. Пусть $A \in \mathrm{Ob}\, c\text{-}DESC$, $i : sig(A) \to X \in \mathrm{Iso}\, SIG$. Ввиду теоремы 4.1, по условию существует *AO*-изоморфизм $j : mod^*(A) \to Y$ такой, что $int(j) = i$. Поэтому имеется *c-DESC*-изоморфизм $mod(j) : A \to mod(Y)$, *sig*-образ которого совпадает с i. □

Существуют технологии проектирования, аспектно-ориентированное расширение которых не привносит ничего существенно нового. Покажем формально, что они характеризуются отсутствием трассируемых трансформаций, способных порождать нетривиальную аспектную структуру: любая разметка в них оказывается изоморфизмом. Этот критерий эквивалентен ряду других, таких как способность функтора *str* служить для выделения интерфейсов АО-моделей.

Определение 4.6. Формальная технология проектирования *AR* называется *аспектно тривиальной*, если функтор *mod* является эквивалентностью категорий *AO* и *c-DESC*. □

Предложение 4.2. Следующие утверждения эквивалентны для любой формальной технологии *AR*.

(i) Формальная технология *AR* аспектно тривиальна.

(ii) Любая разметка является изоморфизмом.

(iii) Функтор $1_{\_} : SIG \hookrightarrow LAB$ является эквивалентностью категорий.

(iv) Функтор *int* естественно изоморфен функтору *str*.

(v) Функтор *str* унивалентен.

(vi) Функтор *str* переводит любой *tr-AO*-морфизм в изоморфизм.

Доказательство. Мы будем пользоваться свойствами эквивалентности категорий, установленными в [93, разд.4.4]. Обозначим через μ коединицу сопряжения $mod^* \dashv mod$. Зафиксируем произвольную *sig*-разметку $l : X \to L$.

(i) $\Rightarrow$ (ii). В силу п.(iii) доказательства теоремы 4.1 имеем $l = str(\mu_{asp^*(l)})$. По предположению, μ состоит из изоморфизмов, следовательно l – изоморфизм.

(ii) $\Rightarrow$ (iii). Функтор 1_- является полным вложением, и если $l \in \mathrm{Iso}\, SIG$, то $\langle l, 1_L \rangle : l \to 1_L \in \mathrm{Iso}\, LAB$.

(iii) $\Rightarrow$ (i). Функтор $dom \circ il : LAB \to SIG$ сопряжен справа к функтору 1_-, поэтому по предположению он является полным. Следовательно, функтор *mod*, будучи ребром декартова квадрата в **CAT**, параллельным ему, также полон. А поскольку ввиду теоремы 4.1 он унивалентен и сюръективен на объектах, то он является эквивалентностью категорий.

(ii) $\Rightarrow$ (iv). Естественное преобразование функтора *int* в *str*, состоящее из разметок всех АО-моделей, по предположению состоит из изоморфизмов.

(iv) $\Rightarrow$ (v). Любой функтор, естественно изоморфный унивалентному, унивалентен.

(v) $\Rightarrow$ (ii). Рассмотрим *AO*-морфизм $l^* = asp^*(\langle l, 1_L \rangle)$. Поскольку $str(l^*) = 1_L$, по предположению $l^* \in \mathrm{Mono}\, AO$ (любой унивалентный функтор отражает мономорфизмы). Тогда $l = int(l^*) \in \mathrm{Mono}\, SIG$ (любой функтор, имеющий левый сопряженный, сохраняет все пределы, в частности мономорфизмы). Поскольку l обратим справа, получаем, что $l \in \mathrm{Iso}\, SIG$.

(iv) $\Rightarrow$ (vi). Для любого *tr-AO*-морфизма t *SIG*-морфизм $int(t)$ по определению является *sig*-разметкой. Ввиду уже доказанной импликации (iv) $\Rightarrow$ (ii), по условию $int(t) \in \mathrm{Iso}\, SIG$, а значит и $str(t) \in \mathrm{Iso}\, SIG$.

(vi) $\Rightarrow$ (ii). В силу определения коединицы имеем $mod(\mu_P) = 1_{mod(P)}$ для любой АО-модели P, поэтому $\mu_P \in \mathrm{Mor}\, tr\text{-}AO$. В свою очередь, как отмечалось выше, $l = str(\mu_{asp^*(l)})$, так что по предположению $l \in \mathrm{Iso}\, SIG$. □

Примеры аспектно тривиальных технологий были построены в разд.3.4 для формализации проектирования вычислительных систем, где, как известно, аспектно-ориентированный подход не применяется (хотя на уровне инфраструктуры вычислительных сред существуют рассеяннные программно-технические задачи, для реализации которых может применяться АОП) [103]. Другим примером служит любая формальная технология, категория интерфейсов которой представляет собой частичный порядок: в ней любая разметка является тождественным морфизмом, поэтому функтор *mod* представляет собой изоморфизм. В частности, аспектно-ориентированные методы не удается применять в аксиоматическом подходе к разработке спецификаций, которому отвечает формальная технология TS_σ из разд.2.3. Для систем аксиом фиксированной сигнатуры конструкции, естественным образом отвечающие разметке и связыванию, отсутствуют.

Целесообразно также рассмотреть свойства АО-технологий над формальными технологиями, представляющими противоположный «предельный» случай выбора интерфейсов, когда функтор их выделения является изоморфизмом (ср. предложение 2.12).

Предложение 4.3. Следующие утверждения справедливы для любой формальной технологии *AR*, в которой функтор *sig* является изоморфизмом.

(i) Функтор *asp* задает изоморфизм между категорией АО-моделей *AO* и категорией разметок *LAB*.

(ii) $AO_{mod}(AR) \cong AO_{int}(AR)$.

(iii) АО-технология над *AR*, порожденная любым фундаментальным функтором, структурируема, причем в технологии $AO_{int}(AR)$ функтором дискретной структуры служит *str*.

Доказательство.

(i). В декартовом квадрате в **CAT**, определяющем категорию *AO*, функтор *asp* является ребром, параллельным функтору *sig*.

(ii). Тройка $\langle 1_{AO}, sig, 1_{tr\text{-}AO^{op}} \rangle$ представляет собой **ARCH**-изоморфизм, действующий из $AO_{mod}(AR)$ в $AO_{int}(AR)$.

(iii). Структурируемость технологии $AO_{asp}(AR)$ вытекает из уже доказанного утверждения (i) и предложения 2.12. Из него же легко вывести структурируемость технологии $AO_{int}(AR)$, поскольку функтор $dom \circ il$ обладает левым сопряженным с тождественной единицей (это $1_{\_} : SIG \hookrightarrow LAB$), который в свою очередь обладает левым сопряженным с тождественной коединицей – это $codom \circ il$. Имеем $int^{**} = (dom \circ il \circ asp)^{**} = codom \circ il \circ asp^{**} = codom \circ il \circ asp = str$. Отсюда и из уже доказанного утверждения (ii) вытекает структурируемость технологии $AO_{mod}(AR)$. □

4.3 Аспекты и связывание

Рассмотрим процедуры идентификации и связывания аспектов согласно работе [75]. Рассеяние задач приводит к повышению затрат на проектирование и эксплуатацию систем во многом из-за того, что сборка системы может разрушить аспектную структуру ее компонентов. Как указывалось выше, избежать этого можно путем применения действий, которые на уровне аспектных структур обладают обратимостью слева – возможностью идентифицировать аспектную структуру компонента в составе системы путем трассирования. Наилучшим же с точки зрения АОП является действие, не вызывающее существенных изменений в аспектной структуре. Эти соображения служат мотивировкой для следующего определения.

Определение 4.7. *AO*-морфизм f называется *аспектным*, если $str(f)$ является сечением (т.е. имеет левый обратный), и *изоаспектным*, если $str(f)$ является изоморфизмом. □

В АО-технологиях, порожденных формальными технологиями над **Set**, аспектными являются в точности все отображения с непустой областью, переходящие в инъекции под действием функтора *str*. Они не «склеивают» метки и потому допускают однозначное трассирование на уровне аспектной структуры.

Поскольку трансформации АО-моделей хорошо трассируются, для них неразрушающее преобразование аспектной структуры эквивалентно ее сохранению:

Предложение 4.4. Любой *tr-AO*-морфизм является аспектным тогда и только тогда, когда он является изоаспектным.

Доказательство. Для любого *tr-AO*-морфизма $\langle t, b\rangle : \langle A, l\rangle \to \langle B, k\rangle$ ввиду равенства $b \circ l = k \circ sig(t)$ *SIG*-морфизм b обратим справа, поскольку таковыми являются k и $sig(t)$. В свою очередь, любой морфизм, обратимый справа и слева, является изоморфизмом. □

Аспектом (aspect) называется строительный блок аспектно-ориентированной программы, реализующий отдельный класс задач. Как указывалось в разд.4.1, технология АОП нацелена на сохранение идентичности аспектов в составе программы, поэтому их аспектная структура не может быть разрушена при интеграции в любую систему. Это свойство и составляет формальное определение аспекта. Подчеркнем, что оно выходит за рамки классического АОП, где аспект обязательно должен вставляться в некоторую базовую программу посредством связывания: предлагаемая семантика позволяет избавиться от заложенного классиками неявного порочного круга. Например, мы докажем, что в элементарной формальной технологии АОП аспекты – это в точности все АО-модели, аспектная структура которых элементарна.

Определение 4.8. АО-модель A называется *аспектом*, если любой *AO*-морфизм с областью A является аспектным. □

Предложение 4.5. Следующие утверждения эквивалентны для любой АО-модели A.

(i) A является аспектом.

(ii) A изоморфно аспекту.

(iii) A может быть трансформировано в аспект.

(iv) Существует изоаспектный морфизм, направленный из некоторого аспекта в A.

(v) Существует сечение, направленное из A в некоторый аспект.

Доказательство.

(i) $\Rightarrow$ (ii). Тождественный морфизм $1_A : A \to A$ является изоморфизмом.

(ii) $\Rightarrow$ (iii). Вытекает из условия (vii) определения 2.3.

(iii) $\Rightarrow$ (iv). Вытекает из предложения 4.4.

(iv) ⇒ (i). Если B – аспект и $u : B \to A$ – *AO*-морфизм, то согласно определению 4.8 $str(f \circ u)$ является сечением для любого *AO*-морфизма f с областью A. В частности, если u изоаспектен, то *SIG*-морфизм $str(f) = str(f \circ u) \circ str(u)^{-1}$ является сечением.

(i) ⇒ (v). Тождественный морфизм 1_A является сечением.

(v) ⇒ (iv). Рассмотрим сечение $s : A \to B$, где B – аспект. Пусть $s' : B \to A$ – *AO*-морфизм такой, что $s' \circ s = 1_A$. *SIG*-морфизм $str(s')$ имеет как правый обратный (это $str(s)$), так и левый обратный (определение 4.8), следовательно он является изоморфизмом, т.е. s' изоаспектен. □

Предложение 4.6. Если формальная технология *AR* элементарна, то следующие утверждения эквивалентны для любой АО-модели A.

(i) A является аспектом.

(ii) $str(A)$ является терминальным *SIG*-объектом.

(iii) Существует аспектный морфизм, направленный из A в некоторый аспект.

Доказательство. Напомним, что терминальный объект традиционно обозначается через **1**. Заметим, что в любой категории: (a) любой морфизм вида $i : \mathbf{1} \to X$ является сечением (морфизм $!_X : X \to \mathbf{1}$ является левым обратным к нему), поэтому (b) любое сечение вида $!_X : X \to \mathbf{1}$ является изоморфизмом.

(i) ⇒ (ii). Если A – аспект, то ввиду (b) морфизм $str(!_A)$ является изоморфизмом.

(ii) ⇒ (i). Если $str(A)$ – терминальный объект, то в силу (a) любой морфизм с областью A является аспектным.

(i) ⇒ (iii). Тождественный морфизм 1_A является аспектным.

(iii) ⇒ (ii). Если B – аспект (т.е., как уже доказано, $str(B)$ – терминальный объект) и $u : A \to B$ – аспектный *AO*-морфизм, то $str(u)$ представляет собой сечение, направленное в терминальный объект. Поэтому ввиду (b) $str(A)$ также является терминальным объектом. □

Покажем, что импликация (iii) ⇒ (i) предложения 4.6 выполняется не для любой формальной технологии *AR*, несмотря на то, что в ее формулировке не фигурирует терминальный объект. Воспользуемся категорией $\boldsymbol{V}$ из разд.2.2 (это частично упорядоченное множество $\{1' \leftarrow 0 \to 1\}$), рассмотрим функтор $vv : \boldsymbol{V} \to \boldsymbol{2} : 0 \mapsto 0, 1 \mapsto 1, 1' \mapsto 0$. В категории АО-моделей над технологией $\langle \boldsymbol{V}, \varnothing, vv, \boldsymbol{V}^{\mathrm{op}} \rangle$ имеется аспект $mod^*(1')$ и (изо)аспектный морфизм $mod^*(0 \to 1')$, однако АО-модель $mod^*(0)$ не является аспектом, поскольку морфизм $mod^*(0 \to 1)$ не аспектен. В то же время существуют неэлементарные технологии, для которых предложение 4.6 выполняется: очевидным примером служит $\langle \boldsymbol{V}, \varnothing, !_{\boldsymbol{V}} : \boldsymbol{V} \to \boldsymbol{1}, \boldsymbol{V}^{\mathrm{op}} \rangle$.

Рассмотрим процедуру связывания системы из аспектов. Напомним, что в классическом АОП оно состоит в подключении программы W, называемой советом (advice), к базовой программе (base) B в заданных местах, называемых точками соединения (join point) [144]. Каждый

раз, когда при исполнении базовой программы встречается точка соединения, вызывается совет. Поэтому аспект обычно выглядит как блок программного кода, охраняемый (guarded) условием, идентифицирующим точку соединения; начало блока служит точкой вызова совета (entry point). Пример такого блока на языке AspectJ был приведен в разд.1.3. Таким образом, инструмент связывания (weaver) принимает на вход две спецификации:

- описание точек соединения в базовой программе, или срез (pointcut);
- описание точек вызова совета в точках соединения.

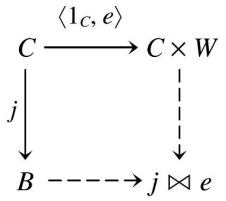
При связывании сначала (виртуально) создается достаточное количество копий совета, по одной на каждую точку соединения, с маркировкой соответствующих им точек вызова. Далее эти точки «склеиваются» друг с другом так, чтобы не разрушить аспектную структуру базы и совета. Для формальной записи правил связывания привлекается дополнительная АО-модель C, называемая связкой (connector, см. [234]), которая интегрируется с базой в точках соединения, а с советом – в точках вызова. В технологиях типа AspectJ в роли связки выступает регулярное выражение, выделяющее в тексте базовой программы синтаксические единицы, образующие срез (конструкция `pointcut`). Соответствие точек соединения точкам вызова задается *AO*-мегамоделью – парой *AO*-морфизмов $j : B \leftarrow C \rightarrow W : e$ (здесь наглядно проявляется отличие связывания от модульной компоновки, формализуемой одношаговым действием вида $l : M \rightarrow S$, где M – модуль, S – система).

Как легко видеть на примере формальной АО-технологии моделирования сценариев, первый шаг связывания может быть формализован как построение произведения $C \times W$, а второй – кодекартова квадрата (стандартная конструкция склеивания элементов множества). Эти операции должны быть естественными относительно вычисления аспектной структуры, чтобы получилось связывание разметок: в соответствие с соображениями, изложенными в гл.2, указанные универсальные конструкции должны детерминироваться функтором *str*.

В литературе рассмотрены два частных случая вычисления связывания как кодекартова квадрата того или иного вида: когда метки трактуются как роли [229] и когда аспекты задаются как инварианты поведения программ, описанных алгебраическими спецификациями [250]. В отличие от них, наше определение имеет общий характер, и, как мы сейчас покажем, оно отражает ряд интуитивно ожидаемых свойств связывания (например возможность трассируемого включения базы в результат).

Определение 4.9. *Аспектным связыванием* пары *AO*-морфизмов $j : B \leftarrow C \rightarrow W : e$, где B называется базой, W – советом, C – связкой, называется кодекартов квадрат пары $j : B \leftarrow C \rightarrow C \times W : \langle 1_C, e \rangle$ (*схемы связывания*), если он существует (в частности, существует произведение $C \times W$) и функтор *str* детерминирует как произведение $C \times W$, так и этот кодекар-

тов квадрат. *Результатом связывания* называется вершина кодекартова квадрата, обозначается через $j \bowtie e$. □



Предложение 4.7. Для любой пары $j : B \leftarrow C \rightarrow W : e$, обладающей связыванием, справедливы следующие утверждения.

(i) Результат связывания определяется единственным образом (с точностью до изоморфизма).

(ii) Существует обратимое слева (в частности, аспектное) вложение $b : B \hookrightarrow (j \bowtie e)$.

(iii) Если результат связывания является аспектом, то и база является аспектом.

(iv) Если j – изоморфизм, то $j \bowtie e \cong B \times W$.

Доказательство.

(i). Вытекает из универсальности (ко)пределов.

(ii). Имеем $\pi_C \circ \langle 1_C, e\rangle = 1_C$, где $\pi_C : C \times W \rightarrow C$ – проекция, так что $\langle 1_C, e\rangle$ – сечение. В свою очередь, как легко проверить, ребро кодекартова квадрата, параллельное сечению, само является сечением (см. двойственное утверждение в [136, предложение 11.18]).

(iii). Вытекает из утверждения (ii) и предложения 4.5 (эквивалентность (i) ⇔ (v)).

(iv). Ребро кодекартова квадрата, параллельное изоморфизму, само является изоморфизмом. □

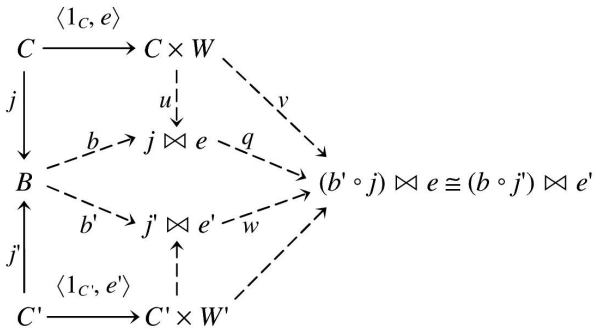
Подчеркнем, что мы не требуем от спецификации или схемы связывания вхождения в класс конфигураций АО-технологии, поскольку он отражает только возможности модульной компоновки АО-моделей. Так что путем связывания можно собирать в одно целое даже совершенно разнородные артефакты жизненного цикла системы, если они входят в число АО-моделей. Например, можно формализовать процедуру комментирования текста программы как связывание с ним фрагментов документации: один фрагмент приписывается ко всем блокам программного кода, реализующим его. Если все же схема связывания существует и принадлежит классу $\mathbf{D}ai^{-1}(AOInt_{ai})$, то критерием существования связывания служит способность функтора *str* детерминировать произведение $C \times W$, поскольку согласно определению 4.4 функтор *str* детерминирует копределы всех конфигураций АО-моделей.

Утверждение (ii) предложения 4.7 позволяет описать многошаговое связывание взаимно независимых советов с общей базой: если имеются связываемые пары $j : B \leftarrow C \rightarrow W : e$ и

$j' : B \leftarrow C' \rightarrow W' : e'$, то их можно естественным образом собрать в одно целое путем связывания пары $b \circ j' : (j \bowtie e) \leftarrow C' \rightarrow W' : e'$. Покажем, что результат здесь не зависит от порядка привязки советов.

Предложение 4.8. Если пары $j : B \leftarrow C \rightarrow W : e$, $j' : B \leftarrow C' \rightarrow W' : e'$, $b \circ j' : (j \bowtie e) \leftarrow C' \rightarrow W' : e'$, $b' \circ j : (j' \bowtie e') \leftarrow C \rightarrow W : e$ обладают связыванием, то $(b \circ j') \bowtie e' \cong (b' \circ j) \bowtie e$.

Доказательство. Пусть *AO*-морфизмы u, v, w таковы, что соотношения $u \circ \langle 1_C, e \rangle = b \circ j$ и $v \circ \langle 1_C, e \rangle = w \circ (b' \circ j)$ задают кодекартовы квадраты. Тогда существует морфизм $q : (j \bowtie e) \rightarrow (b' \circ j) \bowtie e$ такой, что $q \circ u = v$ и $q \circ b = w \circ b'$. В силу общего утверждения, двойственного к известной лемме о декартовых квадратах [136, предложение 11.10(2)], второе из этих равенств определяет кодекартов квадрат пары $b : (j \bowtie e) \leftarrow B \rightarrow (j' \bowtie e') : b'$ с вершиной $(b' \circ j) \bowtie e$. Рассуждая аналогично, получаем, что объект $(b \circ j') \bowtie e'$ является вершиной кодекартова квадрата той же пары. □



Эффект перемешивания классов задач при связывании ярко проявляется, если совет является достаточно мелкой единицей аспектной архитектуры, например аспектом (в смысле определения 4.8). База связывания в некоторой степени поглощает такую единицу, затрудняя ее идентификацию в составе системы. Для формальной характеризации таких единиц введем следующее понятие. Будем называть объект некоторой категории *частично терминальным*, если из любого объекта в него имеется не более одного морфизма. Объект, изоморфный частично терминальному, сам является частично терминальным. Ясно, что любой терминальный объект является частично терминальным. В любой категории предпорядка все объекты частично терминальны. В категории **Set** частично терминальными объектами являются только пустое и одноэлементное множество. Для формальных моделей свойство частичной терминальности однозначно устанавливается на уровне интерфейсов: *c-DESC*-объект T является частично терминальным тогда и только тогда, когда таковым является *SIG*-объект $sig(T)$. Действительно, с од-

ной стороны, ввиду существования сопряжения $sig^* \dashv sig$ имеем $|\mathrm{Mor}(I, sig(T))| = |\mathrm{Mor}(sig^*(I), T)| \leq 1$ для любых *SIG*-объекта I и частично терминального *c-DESC*-объекта T. С другой стороны, если частично терминальным является *SIG*-объект $sig(T)$ для некоторого *c-DESC*-объекта T, то ввиду унивалентности функтора sig имеем $|\mathrm{Mor}(X, T)| \leq |\mathrm{Mor}(sig(X), sig(T))| \leq 1$ для любого *c-DESC*-объекта X.

Предложение 4.9. Для любой пары $j : B \leftarrow C \rightarrow W : e$, обладающей связыванием, справедливы следующие утверждения.

(i) Если $mod(W)$ – частично терминальный *c-DESC*-объект, то $j \bowtie e \cong B$.

(ii) Если $str(W)$ – частично терминальный *SIG*-объект, то $str(j \bowtie e) \cong str(B)$.

Доказательство. Сначала заметим, что если X – частично терминальный объект в произвольной категории и в ней существует морфизм $f : Y \rightarrow X$, то объект Y по определению представляет собой произведение $Y \times X$ с проекциями 1_Y и f (с точностью до изоморфизма).

(i). Если $mod(W)$ – частично терминальный *AO*-объект, то, поскольку по теореме 4.1 функтор *mod* служит для выделения интерфейсов из АО-моделей, W является частично терминальным *AO*-объектом. Следовательно, как было только что установлено, π_C – изоморфизм, поэтому $\langle 1_C, e \rangle$ – изоморфизм, а значит, вложение $b : B \hookrightarrow (j \bowtie e)$, параллельное ему в кодекартовом квадрате, само является изоморфизмом.

(ii). Поскольку $str(C \times W) = str(C) \times str(W)$, аналогично предыдущему получаем, что вложение b – изоаспектный *AO*-морфизм. □

Проверка того факта, что функтор *str* детерминирует универсальные конструкции, возникающие в ходе связывания, упрощается для случая, когда категория моделей исходной модульной формальной технологии проектирования является подвижной (напомним, что в настоящей работе рассматриваются в основном именно такие технологии). Аналогично предложению 2.10, здесь достаточно проверить тот факт, что функтор *str* сохраняет эти конструкции (именно такое требование указано в определении связывания в [75]).

Предложение 4.10. Если категория моделей формальной технологии проектирования *AR* подвижна, то следующие условия эквивалентны для любой *AO*-диаграммы Δ:

(i) Диаграмма Δ обладает копределом и функтор *str* сохраняет ее копределы.

(ii) *SIG*-диаграмма $str \circ \Delta$ обладает копределом и функтор *str* поднимает копределы диаграммы Δ.

Аналогичная эквивалентность имеет место для пределов диаграммы Δ.

Доказательство.

(i) ⇒ (ii). Сначала установим, что функтор *str* удовлетворяет следующему «условию подвижности»: для любых АО-модели $\langle A, l : sig(A) \to L\rangle$ и *SIG*-изоморфизма $i : L \to X$ существуют АО-модель *Y* и *AO*-изоморфизм $j : \langle A, l\rangle \to Y$ такие, что $str(j) = i$. Действительно, пусть $t : U \to V$ – трасса, удовлетворяющая условию $sig(t) = l$. Ввиду подвижности конкретной категории *c-DESC*, существует *c-DESC*-изоморфизм $\hat{\imath}$ такой, что $\mathrm{dom}\ \hat{\imath} = V$ и $sig(\hat{\imath}) = i$. В силу предложения 4.1 морфизм $\hat{\imath} \circ t$ является трассой, причем $sig(\hat{\imath} \circ t) = i \circ l$, поэтому имеется *AO*-изоморфизм $\langle 1_A, i\rangle : \langle A, l\rangle \to \langle A, i \circ l\rangle$, который можно взять в качестве *j*.

С учетом этого факта, импликация (i) ⇒ (ii) доказывается так же, как аналогичная импликация предложения 2.10.

(ii) ⇒ (i). Вытекает из предложения 2.2. □

В [70] показано, что операцию связывания можно формально задать функтором, действующим из подходящей категории спецификаций связывания в *AO*. Пусть *WSP* – полная подкатегория в категории функторов AO^V, класс объектов которой состоит из всех диаграмм, нетождественные морфизмы которых образуют пару, обладающую связыванием.

Предложение 4.11. Существует функтор $weav : WSP \to AO$, сопоставляющий каждому *WSP*-объекту результат связывания его нетождественных морфизмов.

Доказательство. Обозначим через *WSC* полную подкатегорию в AO^V, класс объектов которой состоит из всех диаграмм, представляющих схемы связывания. Пусть $wsc : WSP \to WSC$ – функтор, сопоставляющий каждому *WSP*-объекту схему связывания его нетождественных морфизмов, а произвольному естественному преобразованию произвольного *WSP*-объекта $j : B \leftarrow C \to W : e$ в $j' : B' \leftarrow C' \to W' : e'$ с компонентами $q : B \to B'$, $c : C \to C'$, $w : W \to W'$ – естественное преобразование схем связывания с компонентами *q*, *c*, $c \times w$ (этот набор морфизмов действительно задает естественное преобразование схем, поскольку $(c \times w) \circ \langle 1_C, e\rangle = \langle c, w \circ e\rangle = \langle c, e' \circ c\rangle = \langle 1_{C'}, e'\rangle \circ c$). Имеется функтор

$$weav : WSP \to AO : (j : B \leftarrow C \to W : e) \mapsto j \bowtie e,\ \alpha \mapsto colim(\langle wsc(\alpha), 1_V\rangle).\ \square$$

4.4 Экспликация и модуляризация аспектов

Наиболее явным способом пометки аспекта является его модуляризация, т.е. оформление в виде отдельной единицы модульной архитектуры: объекта, таблицы в базе данных и т.д. Модуляризация всех аспектов, составляющих АО-модель, называется разделением ответственности (separation of concerns) и рассматривается как одна из важнейших задач АОП [255]. Модуляризированные аспекты можно собирать в системы посредством компоновки, не прибегая к связыванию, что позволяет снизить затраты на сборку за счет применения широко доступных

«модульных» технологий инженерии систем. В связи с этим многие технологии АОП, в том числе AspectJ, реализованы в виде расширений технологий модульного программирования так, чтобы программы, скомпонованные из модулей, тривиальным образом разделялись на модуляризируемые аспекты.

Модуляризируемые АО-модели выделяются среди прочих тем, что при интеграции с модулями они ведут себя так же, как модульные единицы. Возможности интеграции модулей в АО-модель определяются ее модульным интерфейсом. В свою очередь, возможности интеграции АО-модели в модули определяются ее аспектной структурой, поскольку при интеграции в модуль аспекты, составляющие АО-модель, выступают в качестве элементарных единиц комплексирования. Формально, модуляризируемые модели образуют полную подкатегорию в *AO* (которую мы будем обозначать через *m-AO*) такую, что существует аспектно-ориентированное расширение (АО-расширение) модульной технологии – вложение $am : c\text{-}DESC \hookrightarrow m\text{-}AO$, полностью воспроизводящее интеграционные возможности модулей в следующем смысле. С одной стороны, все способы интеграции модуля $M \in \mathrm{Ob}\ c\text{-}DESC$ в АО-модель $A \in \mathrm{Ob}\ m\text{-}AO$ задаются множеством морфизмов $\mathrm{Mor}(am(M), A)$, поэтому функтор выделения модульного интерфейса *mod* (точнее, его ограничение на *m-AO*) должен устанавливать биекцию между ним и множеством $\mathrm{Mor}(M, mod(A))$. С другой стороны, все способы интеграции A в M задаются множеством морфизмов $\mathrm{Mor}(A, am(M))$, поэтому должен существовать функтор модуляризации аспектной структуры $am^* : m\text{-}AO \to c\text{-}DESC$, тривиально действующий на модули ($am^* \circ am = 1_{c\text{-}DESC}$) и устанавливающий биекцию между $\mathrm{Mor}(A, am(M))$ и множеством $\mathrm{Mor}(am^*(A), M)$, причем функтор $sig \circ am^*$ должен совпадать с ограничением функтора *str* на *m-AO*.

Примером расширения, которым обладает любая формальная технология проектирования, служит изоморфизм между *c-DESC* и полной подкатегорией в *AO* с классом объектов $\{\langle A, 1_{sig(A)}\rangle \mid A \in \mathrm{Ob}\ c\text{-}DESC\}$, действующий как функтор *mod**, сопряженный слева к *mod* (см. п.(iii) доказательства теоремы 4.1). В технологиях над **Set** он порождает дискретно размеченные *AO*-объекты, которые являются самыми «аспектно-неориентированными» – в них каждый класс задач помечает только один элемент основного множества, так что никакого перемешивания не происходит. Мы назовем это АО-расширение тривиальным, и покажем, что действие любого АО-расширения по существу (с точностью до естественного изоморфизма) совпадает с его действием: модули (т.е. *c-DESC*-объекты) всегда переходят в АО-модели, история получения интерфейсов которых из классов задач путем трансформации утрачена (тривиальна). Таким образом, АО-расширение по существу однозначно определяется своей кообластью – классом всех модуляризируемых АО-моделей. Легко видеть, что конструкция АО-расширения устойчива относительно ограничения: любая полная подкатегория в *m-AO*, содержащая класс

am(Ob *c-DESC*), является кообластью АО-расширения, действующего так же, как *am*. В связи с этим интерес представляют расширения, кообласть которых достаточно «велика».

На языке теории категорий свойства АО-расширения компактно формулируются при помощи конструкции сопряжения функторов.

Определение 4.10. Функтор $am : c\text{-}DESC \to m\text{-}AO$, где *m-AO* – некоторая полная подкатегория в *AO*, называется *аспектно-ориентированным расширением* (АО-расширением) формальной технологии *AR*, если он обладает следующими сопряженными функторами:

- правый сопряженный am_* с тождественной единицей, причем $am_*(f) = mod(f)$ для любого $f \in \mathrm{Mor}\ m\text{-}AO$;
- левый сопряженный *am** с тождественной коединицей, причем $sig(am^*(f)) = str(f)$ для любого $f \in \mathrm{Mor}\ m\text{-}AO$.

АО-расширение *am* называется:

- *тривиальным*, если оно является изоморфизмом;
- *наибольшим*, если любое АО-расширение $am' : c\text{-}DESC \to m\text{-}AO'$ удовлетворяет условию $\mathrm{Ob}\ m\text{-}AO' \subseteq \mathrm{Ob}\ m\text{-}AO$;
- *полным*, если категория *m-AO* эквивалентна *AO*. □

Зафиксируем произвольное АО-расширение $am : c\text{-}DESC \to m\text{-}AO$, пусть $mao : m\text{-}AO \hookrightarrow AO$ – полное вложение.

Предложение 4.12. АО-расширение *am* является полным вложением, обратимым слева: $am_* \circ am = am^* \circ am = 1_{c\text{-}DESC}$. □

Предложение 4.13. Функтор $mao \circ am$ естественно изоморфен функтору *mod**.

Доказательство. По определению, для любых $X, Y \in am(\mathrm{Ob}\ c\text{-}DESC) \subseteq \mathrm{Ob}\ m\text{-}AO$ и любого *AO*-морфизма $f : X \to Y$ имеем $f = am(am_*(f)) = am(mod(f))$, поэтому ввиду предложения 4.12 $str(f) = sig(am^*(f)) = sig(am^*(am(mod(f)))) = int(f)$. Выберем произвольно *SIG*-объект *I*, положим $A = am(sig^*(I))$, $l = asp(A)$, имеем $mod(A) = am^*(am(sig^*(I))) = sig^*(I)$ и $str(A) = int(A) = I$, поэтому $\mathrm{dom}\ l = \mathrm{codom}\ l = I$. Для произвольного *SIG*-морфизма *m* такого, что $l \circ m = 1_I$, непосредственно проверяется существование *AO*-морфизма $\langle sig^*(m \circ l), 1_I \rangle : A \to A$, так что $1_I = sig(sig^*(m \circ l)) = m \circ l$, поэтому $l \in \mathrm{Iso}\ SIG$.

Для произвольного *c-DESC*-объекта *P* положим $k = asp(am(P))$, $P^* = sig^*(sig(P))$, $i = asp(am(P^*))$. Как мы только что доказали, $i \in \mathrm{Iso}\ SIG$, так что непосредственно проверяется существование *AO*-морфизма $\langle \varepsilon_P, k \circ i^{-1} \rangle : am(P^*) \to am(P)$, поэтому $k \circ i^{-1} = sig(\varepsilon_P) = 1_{sig(P)}$, откуда $k = i$. Следовательно, семейство *AO*-изоморфизмов $\langle 1_P, asp(am(P)) \rangle : \langle P, 1_{sig(P)} \rangle \to am(P)$, $P \in \mathrm{Ob}\ c\text{-}DESC$, образует естественный изоморфизм функтора *mod** в $mao \circ am$. □

Предложение 4.14. Формальная технология *AR* является аспектно тривиальной тогда и только тогда, когда она обладает тривиальным полным АО-расширением.

Доказательство. Вытекает из предложений 4.2 и 4.13. □

Таким образом, АО-расширение совместимо и со сборкой систем (ввиду предложения 4.13 оно сохраняет копределы всех диаграмм), и с выделением интерфейсов (ввиду предложения 4.12 имеем $int(am(-)) = sig(am_*(am(-))) = sig(-)$), и с трассированием (ввиду предложения 4.12 оно переводит трассу любой трассируемой трансформации в *tr-AO*-морфизм).

Заметим, что условие существования двух сопряженных (левого и правого), частично воспроизводящее определение 4.10, накладывалось нами в разд.2.3 на функтор дискретной реализации интерфейсов, когда мы вводили понятие структурируемой технологии проектирования. Это не простое совпадение: построим структурируемую подтехнологию в $AO_{mod}(AR)$, в которой функтор аспектно-ориентированного расширения действительно описывает дискретную реализацию модульных основ, рассматриваемых как интерфейсы АО-моделей. Положим $mtr\text{-}AO = m\text{-}AO \bigcap tr\text{-}AO$. Обозначим через *DAOExt* класс всех *c-DESC*-диаграмм $\Theta \in AOInt_{mod}$ (см. определение 4.4) таких, что любая *AO*-диаграмма из класса $mao \circ \mathbf{D}am_*^{-1}(\{\Theta\})$ имеет копредел, принадлежащий классу $mao \circ \mathbf{D}am_*^{-1}(\{\theta\})$, для любого копредела θ диаграммы Θ.

Определение 4.11. *Модуляризируемой АО-технологией* над произвольной формальной технологией проектирования *AR*, порожденной АО-расширением *am*, называется четверка

$$m\text{-}AO_{am}(AR) = \langle m\text{-}AO, \mathbf{D}am_*^{-1}(DAOExt), am_*, mtr\text{-}AO^{op}\rangle. \square$$

Предложение 4.15. Для любых формальной технологии *AR* и АО-расширения *am* четверка $m\text{-}AO_{am}(AR)$ является структурируемой кооднородной технологией проектирования, наибольшей среди подтехнологий в $AO_{mod}(AR)$, обладающих категорией моделей *m-AO* и функтором выделения интерфейсов am_*. Технология $m\text{-}AO_{am}(AR)$ совпадает с $AO_{mod}(AR)$ тогда и только тогда, когда $m\text{-}AO = AO$.

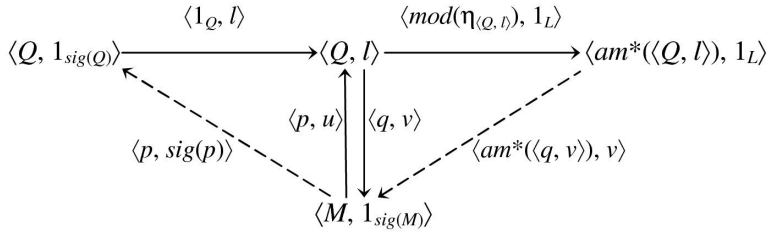
Доказательство. Условие (ii) определения 2.3 вытекает из теоремы 4.1, условие (iii) вместе со структурируемостью – из определения 4.10. Условия (i), (iv), (v) выполняются в силу предложения 2.7. Кооднородность гарантируется аспектной замкнутостью класса *DAOExt* (в смысле определения 4.4). Тройка $\langle mao, 1_{c\text{-}DESC}, mao(-^{op})^{op}\rangle$ задает **ARCH**-мономорфизм из $m\text{-}AO_{am}(AR)$ в $AO_{mod}(AR)$. В любой подтехнологии, вложение которой в $AO_{mod}(AR)$ задается такой тройкой, все конфигурации содержатся в классе $\mathbf{D}am_*^{-1}(DAOExt)$, а все трансформации – в Mor $mtr\text{-}AO^{op}$. □

При формальном описании модуляризации аспектов, составляющих АО-модели, ключевую роль играет единица сопряжения $am^* \dashv am$, которую мы будем обозначать через η.

Напомним, что она является естественным преобразованием функтора $1_{m\text{-}AO}$ в $am \circ am^*$ – семейством m-AO-морфизмов вида $\eta_S : S \to am(am^*(S))$, $S \in \mathrm{Ob}\ m\text{-}AO$, таких, что для любого m-AO-морфизма $m : A \to S$ выполняется тождество $am(am^*(m)) \circ \eta_A = \eta_S \circ m$. По определению, η_S является прообразом c-$DESC$-морфизма $1_{am^*(S)}$ при биекции $am^* : \mathrm{Mor}(S, am(am^*(S)) \cong \mathrm{Mor}(am^*(S), am^*(S))$, поэтому $str(\eta_S) = sig(am^*(\eta_S)) = 1_{sig(am^*(S))} = 1_{str(S)}$ ввиду определения 4.10, т.е. η_S изоаспектен. Поскольку c-$DESC$-объект $am^*(S)$ представляет на модульном уровне аспектную структуру АО-модели S (в частности, $sig(am^*(S)) = str(S)$), морфизм $mod(\eta_S) : mod(S) \to am^*(S)$ можно рассматривать как каноническую интеграцию модульной основы модели в ее модуляризированную аспектную структуру. Согласно доказанному ниже следствию 4.3.2, это регулярный эпиморфизм – категорный аналог сюръективного отображения, переносящего структуру своей области на кообласть. Семейство всех этих морфизмов образует естественное преобразование функтора выделения модульной основы am_* в функтор выделения аспектной структуры am^*.

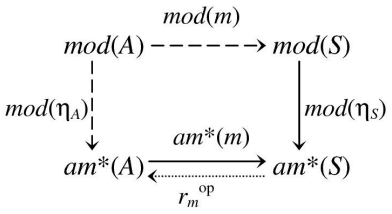
Определение 4.12. (*am*-)*модуляризацией* (аспектной структуры) произвольного m-AO-объекта S называется c-$DESC$-морфизм $mod(\eta_S)$, где η – единица сопряжения $am^* \dashv am$. *Тождеством* (*am*-)*модуляризации* произвольного m-AO-морфизма $m : A \to S$ называется равенство $am^*(m) \circ mod(\eta_A) = mod(\eta_S) \circ mod(m)$. □

Для случая, когда $mao \circ am = mod^*$, действия по извлечению модульной основы и модуляризованной аспектной структуры из произвольной модуляризируемой АО-модели $\langle Q, l : sig(Q) \to L\rangle$ («треугольные тождества» сопряжений) показаны на следующей диаграмме, где M – произвольный c-$DESC$-объект, $\langle p, u\rangle$ и $\langle q, v\rangle$ – произвольные m-AO-морфизмы.



Чтобы извлечь модуляризированные аспекты из модульной основы m-AO-объекта S, нужно трассировать их метки (обозначения классов задач) вдоль модуляризации его аспектной структуры. Если A – такой аспект, то должна существовать трассируемая модульная трансформация $r_m : am^*(A) \to am^*(S)$ его (модуляризированной) метки в (модуляризированную) аспектную структуру модели S. Идентификация метки в аспектной структуре производится посредством прямого трассирования – морфизма, обратного справа к трассе $r_m{}^{\mathrm{op}}$ (см. пояснения перед

определением 4.1). Таким морфизмом должен служить $am^*(m)$, где m – вхождение A в S. Поэтому $mod(m)$ должен выделять в $mod(S)$ полный прообраз вложения $am^*(m)$ относительно $mod(\eta_S)$. Напомним, что в произвольной категории полный прообраз объекта V, включающегося в объект W посредством мономорфизма $e : V \hookrightarrow W$, относительно морфизма $f : X \to W$ вычисляется как декартов квадрат (см. [30, разд.3.13, пример 2]) – предел диаграммы, имеющей вид пары стрелок с общим концом $e : V \hookrightarrow W \leftarrow X : f$ (точнее, прообраз выделяется ребром предела, направленным в X, причем оно также является мономорфизмом). В нашем случае декартовым квадратом должно служить именно тождество модуляризации для морфизма $m : A \to S$. Таким образом, подаспекты – это вложения аспектов, модуляризация которых имеет универсальный характер. Как и следует ожидать, аспекты являются атомарными единицами разделения ответственности – мы покажем, что они не имеют собственных подаспектов.



Понятие подаспекта имеет ряд общих черт с теоретико-категорным понятием подобъекта [30, гл.4]. Напомним, что многие известные категории (в частности, категории «над» **Set**) обладают классификатором подобъектов – выделенным объектом O с элементом $true : \mathbf{1} \hookrightarrow O$, позволяющим задавать подобъекты посредством характеристических функций. А именно, подобъект объекта X – это любой морфизм $m : Y \to X$, для которого существует единственный морфизм $x_m : X \to O$, характеристический в том смысле, что соотношение $true \circ !_Y = x_m \circ m$ выполняется и задает декартов квадрат – предел пары $true : \mathbf{1} \hookrightarrow O \leftarrow X : x_m$, ребра которого образуют пару $!_Y : \mathbf{1} \leftarrow Y \to X : m$. Например в **Set** классификатором подобъектов служит двухэлементное множество $\{true, false\}$. Как мы увидим ниже (см. доказательство предложения 4.16), для подаспекта в роли характеристического морфизма выступает η_S, а вместо терминального объекта, идентифицирующего x_m-образ подобъекта в классификаторе, используется модуляризованная метка аспекта (ср. предложение 4.6).

Непосредственно проверяется, что подобъекты являются регулярными мономорфизмами [30, разд.5.1]. Мы покажем, что таковыми являются и подаспекты. Поэтому разбиение (в смысле определения 2.8) АО-модели на подаспекты, задающее разделение ответственности, является неразрушающим. В связи с этим представляет интерес оптимизация разделения ответственно-

сти по критериям, задаваемым на модульном уровне. Например, если модульная архитектура поддерживает параллелизм, то можно эффективно создавать распределенные аспектно-ориентированные системы, в которых модуляризированные аспекты исполняются параллельно.

Определение 4.13. *m-AO*-морфизм $m : A \to S$ называется (*am*-)*подаспектом* АО-модели *S*, если он удовлетворяет следующим условиям.

(i) АО-модель *A* является аспектом.

(ii) Существует трассируемая трансформация $r_m : am^*(A) \to am^*(S)$ такая, что $r_m{}^{\mathrm{op}} \circ am^*(m) = 1_{am^*(A)}$.

(iii) Тождество *am*-модуляризации для *m* задает декартов квадрат в категории *c-DESC*.

Разделением ответственности АО-модели называется ее *int*-разбиение, каждая компонента которого является подаспектом. Формальная технология *AR аспектно отражает* оптимизируемый класс *sig*-разбиений $OPM \subseteq \mathrm{Cocone}\ c\text{-}DESC$, если любое разбиение из *OPM* является **D***mod*-образом разделения ответственности некоторой АО-модели над *AR*. □

Предложение 4.16. Любой подаспект является аспектным регулярным *AO*-мономорфизмом.

Доказательство. Покажем, что если $m : A \to S$ – подаспект, то тождество единицы $am(am^*(m)) \circ \eta_A = \eta_S \circ m$ задает декартов квадрат в категории *AO*. Действительно, функтор *mod* переводит его в тождество модуляризации, которое является декартовым квадратом по условию (iii) определения 4.13. Функтор *int* переводит его в тождество $str(m) \circ int(\eta_A) = int(\eta_S) \circ int(m)$, которое представляет собой *sig*-образ тождества модуляризации, так что задает декартов квадрат в *SIG* (ввиду условия (iii) определения 2.3 функтор *sig* сохраняет все пределы). Наконец, функтор *str* переводит тождество единицы в коммутативный квадрат в *SIG*, две параллельных стороны которого ($str(\eta_A)$ и $str(\eta_S)$) являются тождественными морфизмами, он также декартов. Используя эти факты, непосредственно проверяется, что тождество единицы задает декартов квадрат (ср. конструкцию предела в категории стрелок, описанную в разд.4.2 при введении фундаментальных функторов). Следовательно, *m* – уравнитель пары *AO*-морфизмов η_S, $(am(am^*(m) \circ r_m{}^{\mathrm{op}}) \circ \eta_S) : S \rightrightarrows am(am^*(S))$. □

Предложение 4.17. Следующие утверждения эквивалентны для любого подаспекта $m : A \to S$.

(i) *S* является модуляризируемым аспектом.

(ii) *m* является изоморфизмом.

(iii) *m* изоаспектен.

Доказательство.

(i) ⇒ (iii). Если S – аспект, то AO-морфизм $t \circ \eta_S$, где $t = am(r_m^{op})$, является аспектным. Поскольку η_S изоаспектен, $str(t)$ обратим слева, следовательно AO-морфизм t, будучи ретракцией, изоаспектен. Поэтому морфизм $w = str(am(am^*(m)))$, правый обратный к $str(t)$, является изоморфизмом. В свою очередь, применяя функтор str к тождеству единицы $am(am^*(m)) \circ \eta_A = \eta_S \circ m$, получаем, что $w = str(m)$.

(iii) ⇒ (ii). Как отмечалось в доказательстве предложения 4.16, в SIG имеется декартов квадрат, задаваемый тождеством $str(m) \circ int(\eta_A) = int(\eta_S) \circ int(m)$. По предположению, $str(m)$ – изоморфизм, поэтому $int(m)$, будучи параллельным ему ребром декартова квадрата, также является изоморфизмом, в частности эпиморфизмом. А поскольку по теореме 4.1 функтор int универсален, он отражает эпиморфизмы. В свою очередь, если m – эпиморфизм, то в силу предложения 4.16 он является изоморфизмом.

(ii) ⇒ (i). Вытекает из предложения 4.5. □

Предложение 4.18. Если АО-модель S обладает разделением ответственности с непустым основанием, то оно состоит из единственной компоненты тогда и тогда, когда S является аспектом.

Доказательство. Вытекает из предложений 4.16, 4.17 и 2.16. □

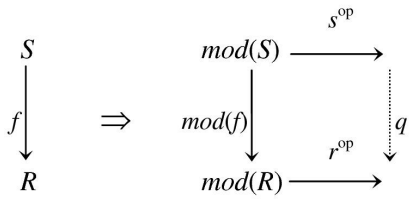
Естественным (хотя и не единственным) способом модуляризации АО-модели является восстановление трассируемой трансформации, породившей ее разметку [214]. Действительно, если оно возможно, то аспектная структура модели приобретает форму модульной единицы, состоящей из всех своих классов задач. Восстановление трансформации можно считать полноценным, если любое действие по интеграции модели в любую систему проецируется на модульный уровень, в виде пары действий по интеграции модульных основ и аспектных структур, согласованных с восстановленными трансформациями. Часто удается аналогичным образом восстановить и трансформации АО-моделей – спроецировать их на модульный уровень в виде согласованной пары трансформаций модульных основ и аспектных структур.

Мы покажем, что трансформация восстанавливается полноценно тогда и только тогда, когда ее трасса является регулярным эпиморфизмом – категорным аналогом факторизации множества (канонического отображения на фактор-множество по некоторому отношению эквивалентности). Этот факт согласуется с трактовкой разметки как обобщенного отношения эквивалентности, которым она по существу является в формальных технологиях над **Set**. В них восстановление трансформации АО-модели $\langle A, l \rangle$ возможно в случае, когда разметка в определенной степени согласована со структурой объекта A. А именно, восстановление состоит в создании структуры на множестве $l(|A|)$, превращающей отображение l в трассу. Если такую структуру удается создать, то АО-модель обычно полностью разделяется на модуляризируемые аспек-

ты (т.е. обладает разделением ответственности): их семейство имеет вид $\{\langle l^{-1}(x), l_x : y \mapsto x\rangle \mid x \in l(|A|)\}$. Ввиду предложений 4.16 и 2.14 это разделение ответственности задается однозначно (с точностью до изоморфизма). Например, трансформация помеченного сценария восстанавливается (очевидным образом) из такой и только такой разметки, которая разбивает A на хорошо упорядоченную совокупность прообразов точек. Очевидными примерами являются: любой дискретно размеченный сценарий, любой аспект, а также любой сценарий, в котором все элементы каждой компоненты связности порядка помечены одной меткой. Поскольку *SM* поддерживает трассирование, восстановление всегда полноценно (см. предложение 4.20 ниже). Любая трансформация АО-моделей сценариев, допускающих восстановление трансформаций, сама восстанавливаема (следствие 5.1.5). Как и следует ожидать, аспектное связывание в общем случае разрушает модуляризируемость: в разд.5.3 мы приведем пример связывания дискретно размеченного сценария с аспектом в системе мониторинга, порождающий немодуляризируемый сценарий.

Мы будем называть *экспликациями* действия модульного уровня, порождающие разметки, интеграцию и трансформацию АО-моделей [214]. Условие полноценности экспликации формализуется как подходящее требование универсальности. Дадим определения необходимых понятий, выявим их теоретико-категорные свойства, и построим с их помощью АО-расширение, которое будем считать каноническим и применять по умолчанию для разделения ответственности.

Определение 4.14. *Экспликацией* (аспектной структуры) АО-модели $\langle A, l\rangle$ называется трассируемая трансформация s некоторого *r-DESC*-объекта в A такая, что $sig(s^{op}) = l$. *Экспликацией действия АО*-морфизма $f : S \to R$ (вдоль экспликаций s и r АО-моделей S и R, соответственно) называется *c-DESC*-морфизм q такой, что $q \circ s^{op} = r^{op} \circ mod(f)$. Экспликация s АО-модели S называется *универсальной*, если любой *АО*-морфизм с областью S эксплицируем вдоль s и любой экспликации своей кообласти. *Аспектным ядром* формальной технологии *AR* называется полная подкатегория *c-AO* в *AO*, состоящая из всех объектов, обладающих универсальной экспликацией. Формальная технология *AR* называется *аспектно универсальной*, если ее ядро совпадает с *AO*. *Экспликацией трансформации* АО-моделей g называется трансформация некоторых *r-DESC*-объектов, обладающая трассой, эксплицирующей действие *АО*-морфизма g^{op}. □



Предложение 4.19. Следующие утверждения эквивалентны для произвольной АО-модели $S = \langle A, l : sig(A) \to L\rangle$ и ее произвольной экспликации $s : P \to A$.

(i) Экспликация s универсальна.

(ii) Трасса s^{op} является регулярным *c-DESC*-эпиморфизмом.

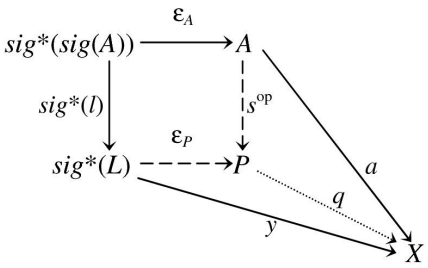
(iii) Трасса s^{op} является финальным *c-DESC*-морфизмом.

(iv) Тождество коединицы $s^{op} \circ \varepsilon_A = \varepsilon_P \circ sig^*(l)$ задает кодекартов квадрат в *c-DESC*.

Доказательство.

(i) $\Rightarrow$ (iii). Напомним [136, определение 8.10], что финальный морфизм в категории *c-DESC* (рассматриваемой вместе с функтором *sig* как конкретная категория) – это в точности инициальный морфизм в категории $c\text{-}DESC^{op}$. Таким образом, морфизм s^{op} по определению является финальным, если для любых *c-DESC*-морфизма $m : A \to X$ и *SIG*-морфизма $k : sig(P) \to sig(X)$ таких, что $k \circ l = sig(m)$, существует *c-DESC*-морфизм $k^+ : P \to X$ такой, что $sig(k^+) = k$ (так что $k^+ \circ f = m$ ввиду унивалентности функтора *sig*). Проверим, что в качестве k^+ можно выбрать экспликацию *AO*-морфизма $\langle m, k\rangle : \langle A, l\rangle \to \langle X, 1_{sig(X)}\rangle$ вдоль s и 1_X (ясно, что 1_X служит экспликацией АО-модели $\langle X, 1_{sig(X)}\rangle$). Действительно, при таком выборе ввиду определения 4.14 имеем $k^+ \circ s^{op} = m$, откуда $sig(k^+) \circ l = sig(m) = k \circ l$, а поскольку l обратим справа, имеем $sig(k^+) = k$. Таким образом, s^{op} – финальный морфизм.

(iii) $\Rightarrow$ (iv). Выберем произвольно пару *c-DESC*-морфизмов $a : A \to X \leftarrow sig^*(L) : y$ такую, что $a \circ \varepsilon_A = y \circ sig^*(l)$. Применяя функтор *sig* к этому равенству, получаем $sig(a) = sig(y) \circ l$, так что ввиду финальности существует *c-DESC*-морфизм $q : P \to X$ такой, что $sig(q) = sig(y)$, откуда $q \circ s^{op} = a$. Тогда $(q \circ \varepsilon_P) \circ sig^*(l) = q \circ s^{op} \circ \varepsilon_A = a \circ \varepsilon_A = y \circ sig^*(l)$, так что и $q \circ \varepsilon_P = y$ ввиду наличия у морфизма l правого обратного. С учетом того, что ε_P – эпиморфизм, получаем, что q – стрелка копредела диаграммы $\varepsilon_A : A \leftarrow sig^*(sig(A)) \to sig^*(L) : sig^*(l)$, задаваемого тождеством коединицы.



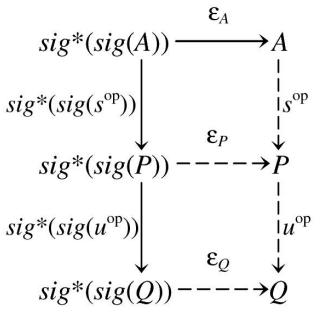
(iv) $\Rightarrow$ (ii). Ребро любого кодекартова квадрата, параллельное регулярному эпиморфизму, само является таковым (см. двойственное утверждение в [136, предложение 11.18(2)]). А поскольку все ретракции являются регулярными эпиморфизмами [136, предложение 7.75(1)],

ребро копредела s^{op}, параллельное ретракции $sig^*(l)$, представляет собой регулярный эпиморфизм.

(ii) $\Rightarrow$ (i). Предположим, что s^{op} является коуравнителем пары *c-DESC*-морфизмов $u, v : Q \rightrightarrows A$. Выберем произвольно АО-модель R с экспликацией r и АО-морфизм $\langle h, b\rangle : S \to R$, положим $p = r^{\mathrm{op}} \circ h$, так что $sig(p) = b \circ l$. Имеем $sig(p \circ u) = b \circ l \circ sig(u) = b \circ sig(s^{\mathrm{op}} \circ u) = b \circ sig(s^{\mathrm{op}} \circ v) = sig(p \circ v)$, откуда $p \circ u = p \circ v$ ввиду унивалентности функтора sig. По определению коуравнителя, существует *c-DESC*-морфизм q такой, что $q \circ s^{\mathrm{op}} = p$, он является экспликацией действия морфизма $\langle h, b\rangle$. □

Предложение 4.20. Все *r-DESC*-объекты и универсальные экспликации всех объектов ядра образуют подкатегорию в *r-DESC*, содержащую все обратимые трансформации, а также все трассируемые трансформации *sig*-дискретных объектов в произвольные.

Доказательство. Напомним, что все ретракции являются регулярными эпиморфизмами, т.е. любая обратимая трансформация s удовлетворяет условию (ii) предложения 4.19. В частности, тождественный морфизм 1_A служит универсальной экспликацией АО-модели $\langle A, 1_{sig(A)}\rangle$. Кроме того, если $s : P \to A$ и $u : Q \to P$ – универсальные экспликации (АО-моделей $\langle A, sig(s^{\mathrm{op}})\rangle$ и $\langle P, sig(u^{\mathrm{op}})\rangle$, соответственно), то $v = s \circ u$ – универсальная экспликация (АО-модели $\langle A, sig(v^{\mathrm{op}})\rangle$) ввиду импликации (iv) $\Rightarrow$ (i) предложения 4.19 и возможности строить композицию кодекартовых квадратов (см. двойственное утверждение в [136, предложение 11.10(1)]): поскольку два квадрата, из которых составлена нижеприведенная диаграмма, кодекартовы, то таковым является и внешний прямоугольник.



Осталось рассмотреть произвольную трассируемую трансформацию $s : P \to A$ такую, что $\varepsilon_P \in \mathrm{Iso}\, SIG$. Покажем, что в этом случае тождество коединицы $s^{\mathrm{op}} \circ \varepsilon_A = \varepsilon_P \circ sig^*(sig(s^{\mathrm{op}}))$ представляет собой кодекартов квадрат. Выберем произвольно пару *c-DESC*-морфизмов $a : A \to X \leftarrow sig^*(sig(P)) : y$ такую, что $a \circ \varepsilon_A = y \circ sig^*(sig(s^{\mathrm{op}}))$. Положим $q = y \circ \varepsilon_P^{-1}$. Имеем $q \circ \varepsilon_P = y$, и кроме того, $(q \circ s^{\mathrm{op}}) \circ \varepsilon_A = q \circ \varepsilon_P \circ sig^*(sig(s^{\mathrm{op}})) = y \circ sig^*(sig(s^{\mathrm{op}})) = a \circ \varepsilon_A$, откуда и $q \circ s^{\mathrm{op}} = a$, поскольку ε состоит из эпиморфизмов. Таким образом, q – стрелка копредела, задава-

емого тождеством коединицы. Снова обращаясь к предложению 4.19, получаем, что s действительно является универсальной экспликацией АО-модели $\langle A, sig(s^{op})\rangle$. □

Отсюда вытекает, что если формальная технология AR поддерживает трассирование, то любая экспликация АО-модели в АО-технологии над ней универсальна. В общем случае это не так: существуют и такие формальные технологии, ядро которых содержит объекты с необратимой универсальной экспликацией, и такие, вне ядра которых имеются эксплицируемые объекты. Для построения примеров воспользуемся категорией $\boldsymbol{V}$. Пусть $q : 0 \to 1$, $q' : 0 \to 1'$ – $\boldsymbol{V}$-морфизмы. Обозначим через $\boldsymbol{V}^+$ категорию, порожденную из $\boldsymbol{V}$ путем добавления морфизма $s : 1 \to 0$, удовлетворяющего соотношению $q \circ s = 1_1$ (он приводит к появлению следующих морфизмов в дополнение к Mor $\boldsymbol{V}$ и s: $s_0 = s \circ q : 0 \to 0$, $s_1 = q' \circ s : 1 \to 1'$ и $s_{1'} = q' \circ s_0 : 0 \to 1'$), объекта T и морфизма $t : 1' \to T$, удовлетворяющего соотношению $t \circ q' = t \circ s_{1'}$ (так что T – терминальный $\boldsymbol{V}^+$-объект). Пусть $\boldsymbol{2}^+$ – полная подкатегория в $\boldsymbol{V}^+$ с классом объектов $\{0, 1\}$, $vv^+ : \boldsymbol{V}^+ \to \boldsymbol{2}^+$ – функтор, который действует на $\boldsymbol{2}^+$ тождественно и переводит объект 1' в 0 (соответственно, морфизм q' в 1_0), а T в 1. Четверка $\langle \boldsymbol{V}^+, \varnothing, vv^+, (\boldsymbol{V}^+)^{op}\rangle$ является формальной технологией проектирования, и в АО-технологии над ней необратимая трансформация $t^{op} : T \to 1'$ служит универсальной экспликацией аспекта $\langle 1', q\rangle$. Далее, обозначим через $\boldsymbol{V}^{++}$ категорию, порожденную из $\boldsymbol{V}^+$ путем добавления объекта T' и морфизма $t' : 1' \to T'$, удовлетворяющего соотношению $t' \circ q' = t' \circ s_{1'}$. Пусть $vv^{++} : \boldsymbol{V}^{++} \to \boldsymbol{2}^+$ – функтор, который действует на $\boldsymbol{V}^+$ как vv^+ и переводит T' в 1. Четверка $\langle \boldsymbol{V}^{++}, \varnothing, vv^{++}, (\boldsymbol{V}^{++})^{op}\rangle$ является формальной технологией проектирования, но в АО-технологии над ней аспект $\langle 1', q\rangle$ обладает в точности двумя экспликациями t^{op} и t'^{op}, ни одна из которых не универсальна.

Теорема 4.2. Существует АО-расширение $ac : c\text{-}DESC \hookrightarrow c\text{-}AO : A \mapsto \langle A, 1_{sig(A)}\rangle$, $g \mapsto \langle g, sig(g)\rangle$, причем $ac^*(f)$ эксплицирует любой c-AO-морфизм f, и ac-модуляризация любого c-AO-объекта S представляет собой трассу его универсальной экспликации.

Доказательство. Ввиду предложения 4.20, пара отображений ac действительно является функтором. По теореме 4.1 правым сопряженным к нему является функтор $mod \circ cao$, где $cao : c\text{-}AO \hookrightarrow AO$ – полное вложение. Далее, пусть υ – отображение, сопоставляющее каждому c-AO-объекту какую-либо его универсальную экспликацию. Поскольку любая трасса является эпиморфизмом (предложение 4.1), экспликация q любого c-AO-морфизма $f : S \to R$ определяется равенством $q \circ \upsilon_S^{op} = \upsilon_R^{op} \circ mod(f)$ однозначно. Отображение, переводящее f в q, является функцией морфизмов функтора ac^*, левого сопряженного к ac, а семейство c-AO-морфизмов $\langle \upsilon_S^{op}, 1_{str(S)}\rangle : S \to \langle \mathrm{dom}\, \upsilon_S, 1_{str(S)}\rangle$, $S \in \mathrm{Ob}\, c\text{-}AO$ – единицей этого сопряжения. Непосредственно проверяется, что таким способом действительно задается сопряжение $ac^* \dashv ac$ с тождественной

коединицей (с учетом эквивалентности условий (i) и (iv) предложения 4.19, аналогично первой части доказательства теоремы 4.3). □

Следствие 4.2.1. Универсальная экспликация АО-модели, если она существует, определяется однозначно с точностью до изоморфизма. □

Следствие 4.2.2. АО-технология над аспектно универсальной формальной технологией, порожденная функтором *mod*, структурируема.

Доказательство. Вытекает из предложения 4.15. □

Следствие 4.2.3. Если АО-модель S обладает *ac*-подаспектом, то аспектное ядро содержит аспект с модульной основой $mod(S)$.

Доказательство. Пусть χ – единица сопряжения $ac^* \dashv ac$, $m : A \to S$ – произвольный *ac*-подаспект. Покажем, что пара $T = \langle mod(S), sig(r_m{}^{op} \circ mod(\chi_S))\rangle$ является искомым аспектом. Действительно, в силу теоремы 4.2 и предложения 4.20 она является АО-моделью с универсальной экспликацией $mod(\chi_S)^{op} \circ r_m$. Кроме того, поскольку $(r_m{}^{op} \circ mod(\chi_S)) \circ mod(m) = r_m{}^{op} \circ ac^*(m) \circ mod(\chi_A)^{op} = mod(\chi_A)^{op}$, имеется изоаспектный *AO*-морфизм $\langle mod(m), 1_{str(A)}\rangle : A \to T$, так что в силу предложения 4.5 T является аспектом. □

Определение 4.15. *Ядерной АО-технологией* над произвольной формальной технологией проектирования *AR* называется модуляризируемая АО-технология над *AR*, порожденная функтором *ac*. □

Следует предостеречь, что аспектное ядро в общем случае не замкнуто относительно изоморфизмов, т.е. могут существовать две изоморфные АО-модели, одна из которых обладает универсальной экспликацией, а другая – нет. Эта особенность отличает универсальную экспликацию от других конструкций, определенных с точностью до изоморфизма. Здесь имеется критерий, основанный на способности изоморфизмов выступать в роли как действий по интеграции, так и трансформаций. Он сводится к наличию минимально необходимого количества конфигураций (предложение 2.9).

Предложение 4.21. Для любой формальной технологии *AR* следующие утверждения эквивалентны.

(i) Аспектное ядро замкнуто относительно изоморфизмов.

(ii) Любая АО-модель $\langle A, i\rangle$ такая, что $i \in \mathrm{Iso}\, SIG$, обладает универсальной экспликацией.

(iii) Категория *c-DESC* подвижна.

Доказательство.

(i) $\Rightarrow$ (ii). Если АО-модель $\langle A, i\rangle$ удовлетворяет условию $i \in \mathrm{Iso}\, SIG$, то имеется *AO*-изоморфизм $\langle 1_A, i\rangle : \langle A, 1_{sig(A)}\rangle \to \langle A, i\rangle$. Ввиду теоремы 4.2 $\langle A, 1_{sig(A)}\rangle \in \mathrm{Ob}\, c\text{-}AO$, так что по предположению $\langle A, i\rangle \in \mathrm{Ob}\, c\text{-}AO$.

(ii) $\Rightarrow$ (iii). Выберем произвольно *c-DESC*-объект A, *SIG*-объект X и *SIG*-изоморфизм $i : sig(A) \to X$. Ввиду утверждения (ii) предложения 4.1, пара $\langle A, i\rangle$ представляет собой АО-модель. По предположению, она обладает универсальной экспликацией, обозначим ее через $s : P \to A$. Из соотношения $sig(s^{\mathrm{op}}) = i$ вытекает, что $s^{\mathrm{op}} \in \mathrm{Mono}\, c\text{-}DESC$, поскольку любой унивалентный функтор отражает мономорфизмы. В свою очередь, если регулярный эпиморфизм является мономорфизмом, то он является изоморфизмом, поэтому ввиду предложения 4.19 $s^{\mathrm{op}} \in \mathrm{Iso}\, c\text{-}DESC$. Таким образом, категория *c-DESC* подвижна.

(iii) $\Rightarrow$ (i). Выберем произвольно *c-AO*-объект $\langle A, l : sig(A) \to L\rangle$, АО-модель $\langle B, k\rangle$ и *AO*-изоморфизм $\langle i, i'\rangle : \langle A, l\rangle \to \langle B, k\rangle$ (так что $i' \circ l = k \circ sig(i)$). Пусть $s : P \to A$ – универсальная экспликация объекта $\langle A, l\rangle$. Поскольку $sig(s^{\mathrm{op}}) = l$, имеем $sig(P) = L$, так что по предположению существует *c-DESC*-изоморфизм j с областью P, удовлетворяющий условию $sig(j) = i'$. Рассмотрим *c-DESC*-морфизм $r = j \circ s^{\mathrm{op}} \circ i^{-1}$. Имеем $sig(r) = i' \circ l \circ sig(i)^{-1} = k$, поэтому ввиду предложения 4.20 r^{op} является универсальной экспликацией АО-модели $\langle B, k\rangle$. □

Например если формальная технология *AR* аспектно тривиальна, то в АО-технологии над ней любая АО-модель имеет вид, указанный в условии (ii) предложения 4.21. Поэтому аспектно тривиальная технология аспектно универсальна тогда и только тогда, когда ее категория моделей подвижна.

В пояснении перед определением 4.14 отмечалось, что эксплицирование является естественным, но не единственным способом модуляризации АО-моделей. Так, в АО-технологии моделирования сценариев класс всех АО-сценариев, допускающих частичное упорядочение множества меток, превращающее разметку в гомоморфизм, строго содержит аспектное ядро. Например, в этот класс входит следующий неэксплицируемый трехэлементный сценарий с двумя несравнимыми элементами, размеченный двумя метками: $\bullet \leftarrow \bullet \rightarrow \circ$. Обозначим через *HAOSM* полную подкатегорию в категории всех АО-сценариев, объекты которой образуют этот класс. Вложение категории **Pos** в *HAOSM*, задающее дискретную разметку сценария, является АО-расширением: правым сопряженным к нему является функтор, «забывающий» разметку, а левым сопряженным – функтор, переводящий произвольный *HAOSM*-объект $\langle X, \leq, l : X \to L\rangle$ в частично упорядоченное множество $\langle L, \preccurlyeq\rangle$, где $\preccurlyeq$ – пересечение всех частичных порядков на L, превращающих l в гомоморфизм. Ниже мы увидим (см. следствие 4.3.3), что это АО-расширение является наибольшим, так что технология *SM* не обладает полным АО-расширением. В частности, функтор *mod**, задающий дискретную разметку сценария, не удо-

влетворяет второму пункту определения 4.10. Действительно, функтор *mod***, сопряженный к нему слева (это функтор дискретной структуры в $AO_{mod}(SM)$) переводит произвольный АО-сценарий $\langle X, \leq, l\rangle$ в частично упорядоченное множество $\langle X/{\equiv}, {\lesssim}/{\equiv}\rangle$, где $x \equiv y \Leftrightarrow (x \lesssim y \wedge y \lesssim x)$, $\lesssim$ – транзитивное замыкание объединения отношения порядка с отношением эквивалентности $\ker l$. Пусть *Il* – сценарий, содержащий более двух событий и состоящий из двух линейно упорядоченных аспектов, исполняющихся в перемежающемся режиме (interleaving): $\bullet \to \circ \to \bullet \to \circ \to \ldots$ (он изоморфен начальному отрезку множества натуральных чисел, содержащему число 2, с отношением сравнимости по mod 2). Имеем *mod***(*Il*) = **1**, несмотря на то, что $|str(Il)| = 2$.

Для формальных технологий, не обладающих свойством аспектной универсальности, можно предложить более слабые подходы к модуляризации аспектной структуры. В частности, аспектную разметку произвольной АО-модели можно представить на модульном уровне частичным морфизмом модульной основы. Напомним [136, определение 28.1(1)], что частичным морфизмом из *X* в *Y* в категории *C* называется пара *C*-морфизмов с общим началом, один из которых является мономорфизмом и направлен в *X* (он выделяет «часть» объекта *X*, выступающую областью определения частичного морфизма), а другой произволен и направлен в *Y* (он задает действие частичного морфизма). Частичный морфизм можно рассматривать как диаграмму со схемой $\boldsymbol{V}$ (напомним, что так обозначается частично упорядоченное множество $\{1' \leftarrow 0 \rightarrow 1\}$ с несравнимыми элементами 1 и 1'). Если в категории *C* имеется достаточно декартовых квадратов, то определена композиция частичных морфизмов: если для пары произвольных частичных морфизмов $m : X \leftarrow A \rightarrow Y : f$ и $m' : Y \leftarrow B \rightarrow Z : f'$ декартов квадрат пары *C*-морфизмов $f : A \rightarrow Y \leftarrow B : m'$ имеет ребра $m'' : A \leftarrow G \rightarrow B : g$ (в частности, выполняется соотношение $f \circ m'' = g \circ m'$, причем m'' является мономорфизмом ввиду стабильности мономорфизмов относительно декартовых квадратов, см. [136, предложение 11.18]), то их композиция представляет собой частичный морфизм $m \circ m'' : X \leftarrow G \rightarrow Z : f' \circ g$. Непосредственно проверяется, что совокупность всех *C*-объектов и всех их частичных морфизмов с таким законом композиции образует категорию (причем тождественный морфизм объекта *T* в ней имеет вид $1_T : T \leftarrow T \rightarrow T : 1_T$, и вообще любой *C*-морфизм $p : T \rightarrow S$ порождает частичный морфизм $1_T : T \leftarrow T \rightarrow S : p$, так что *C* входит в эту категорию в качестве подкатегории).

Зададим канонический вид частичного морфизма в категории моделей *c-DESC*, взаимно однозначно определяемый АО-моделью. Напомним, что коединица ε состоит из мономорфизмов.

Определение 4.16. *Частичной модуляризацией* произвольной АО-модели $\langle A, l : sig(A) \to L\rangle$ называется следующий частичный *c-DESC*-морфизм из ее модульной основы A в дискретную реализацию аспектной структуры L:

$$\varepsilon_A : A \leftarrow sig^*(sig(A)) \to sig^*(L) : sig^*(l). \square$$

Предложение 4.22. Отображение, сопоставляющее каждой АО-модели ее частичную модуляризацию, задает полное вложение *AO* в категорию функторов $c\text{-}DESC^{\mathbf{V}}$.

Доказательство. Выберем произвольно АО-модели $S = \langle A, l : sig(A) \to L\rangle$ и $S' = \langle A', l' : sig(A') \to L'\rangle$, положим $A^* = sig^*(sig(A))$, $A'^* = sig^*(sig(A'))$. Если они удовлетворяют условиям $\varepsilon_A = \varepsilon_{A'}$ и $sig^*(l) = sig^*(l')$, то $A = \mathrm{codom}\,\varepsilon_A = \mathrm{codom}\,\varepsilon_{A'} = A'$ и $l = sig(sig^*(l)) = sig(sig^*(l')) = l'$. Любой *AO*-морфизм $\langle f, b\rangle : S \to S'$ определяет семейство *c-DESC*-морфизмов $f : A \to A'$, $f^* : A^* \to A'^*$, $sig^*(b) : sig^*(L) \to sig^*(L')$, где $f^* = sig^*(sig(f))$, которое является естественным преобразованием диаграмм частичной модуляризации: по определению коединицы имеем $f \circ \varepsilon_A = \varepsilon_{A'} \circ f^*$, а по определению *AO*-морфизма – $sig^*(b) \circ sig^*(l) = sig^*(l') \circ f^*$. Ясно, что различные *AO*-морфизмы определяют различные естественные преобразования такого вида.

Таким образом, мы задали вложение $pmao : AO \hookrightarrow c\text{-}DESC^{\mathbf{V}}$. Чтобы убедиться в его полноте, рассмотрим произвольное естественное преобразование диаграммы $pmao(S)$ в $pmao(S')$ с компонентами $p : A \to A'$, $q : A^* \to A'^*$, $s : sig^*(L) \to sig^*(L')$, так что имеют место равенства $p \circ \varepsilon_A = \varepsilon_{A'} \circ q$ и $s \circ sig^*(l) = sig^*(l') \circ q$. Поскольку ε состоит из мономорфизмов, из первого равенства вытекает, что $q = sig^*(sig(p))$. С учетом этого, применяя функтор *sig* ко второму равенству, получаем, что $sig(s) \circ l = l' \circ sig(p)$. Следовательно, имеется *AO*-морфизм $\langle p, sig(s)\rangle : S \to S'$. $\square$

Конечно, частичная модуляризация АО-модели $\langle A, l\rangle$ не отражает трансформацию, порождающую ее модульную основу из аспектной структуры, поскольку ее действие не обязано быть трассой трансформации. Тем не менее, можно построить «аппроксимацию» модуляризации аспектной структуры, вычисляя копредел диаграммы частичной модуляризации (если он существует: ребро копредела частичного морфизма, параллельное его действию, можно рассматривать как его универсальное расширение на всю свою область). В литературе копределы диаграмм, представляющих частичные морфизмы некоторого специального вида, применяются при формализации технологий MDE для вычисления результатов процедур коллективного редактирования моделей [243]. В нашем случае речь идет о соединении (в смысле разд.2.1) модульной основы с дискретной реализацией аспектной структуры АО-модели. Согласно предложению 4.19 для любой АО-модели, содержащейся в аспектном ядре, именно так строится ее универсальная экспликация. Здесь мы покажем, что таким способом можно вычислить модуляризацию аспектной структуры любой АО-модели, допускающей ее (т.е. содержащейся в кооб-

ласти какого-либо АО-расширения). При некотором техническом ограничении, встречавшемся нам ранее, отсюда выводится существование наибольшего АО-расширения. Оно модуляризирует в точности все АО-модели, аспектную разметку которых можно поднять на модульный уровень в виде регулярного эпиморфизма.

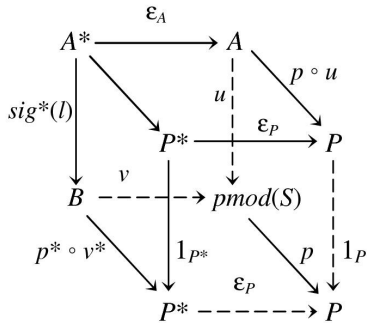
Отправным пунктом для получения этих результатов служит следующий факт: существование копредела у любой частичной модуляризации является критерием структурируемости технологии $AO_{mod}(AR)$ – необходимого условия аспектной универсальности (см. следствие 4.2.2). Этому критерию удовлетворяет ряд ранее рассмотренных случаев: когда формальная технология *AR* аспектно тривиальна (так что действие любой частичной модуляризации является изоморфизмом), когда функтор *sig* является эквивалентностью категорий (так что коединица ε состоит из изоморфизмов), а также когда в категории *c-DESC* существуют все кодекартовы квадраты (это имеет место в технологии моделирования сценариев *SM*, поскольку категория **Pos** кополна).

Теорема 4.3. АО-технология $AO_{mod}(AR)$ структурируема тогда и только тогда, когда диаграмма частичной модуляризации любой АО-модели имеет копредел.

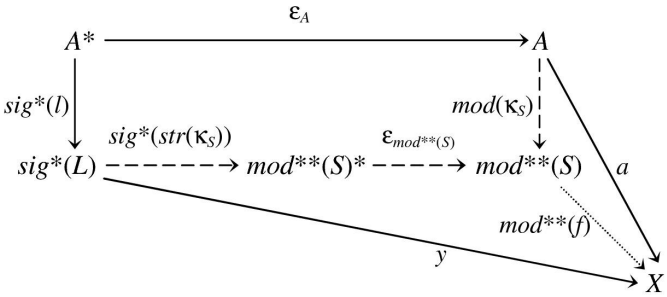
Доказательство. Если все диаграммы из класса *pmao*(Ob *AO*), рассматриваемого как подкатегория в **D***c-DESC*, имеют копредел, то можно определить функтор

$$pmod : AO \to c\text{-}DESC : S \mapsto colim(pmao(S)),\ \ f \mapsto colim(\langle pmao(f),\ 1_V\rangle).$$

Проверим, что он является левым сопряженным к *mod** с тождественной коединицей. Действительно, поскольку любая АО-модель из класса *mod**(Ob *c-DESC*) имеет тождественный морфизм в качестве разметки, функтор *colim* можно выбрать так, чтобы выполнялось соотношение $pmod \circ mod^* = 1_{c\text{-}DESC}$. Покажем, что для любых *c-DESC*-объекта *P*, АО-модели $S = \langle A, l : sig(A) \to L\rangle$ и *c-DESC*-морфизма $p : pmod(S) \to P$ существует единственный *AO*-морфизм $r : S \to mod^*(P)$ такой, что $pmod(r) = p$. Действительно, пусть $u : A \to pmod(S) \leftarrow sig^*(L) : v$ – ребра копредела диаграммы *pmao*(*S*), в частности $u \circ \varepsilon_A = v \circ sig^*(l)$. Отсюда $p \circ u \circ \varepsilon_A = p \circ v \circ sig^*(l)$, следовательно $sig(p \circ u) = sig(p \circ v) \circ l$, так что имеется *AO*-морфизм $\langle p \circ u, sig(p \circ v)\rangle : S \to mod^*(P)$. Ввиду предложения 4.22 под действием функтора *pmod* он переходит в *p*, поэтому является искомым *r* (его единственность вытекает из унивалентности функтора *mod*, установленной в теореме 4.1). Единица сопряжения $pmod \dashv mod^*$ переводит *S* в *AO*-морфизм $\langle u, sig(v)\rangle : S \to mod^*(pmod(S))$.

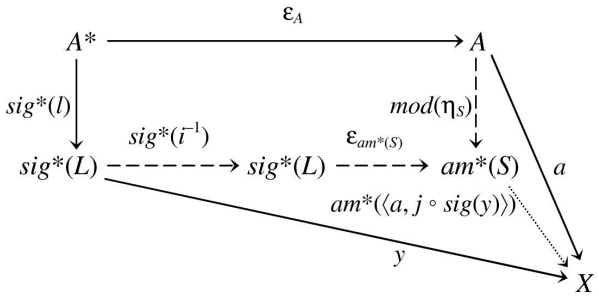


Обратно, предположим, что функтор *mod** обладает левым сопряженным *mod*** : *AO* → *c-DESC* с тождественной коединицей, пусть κ – единица этого сопряжения. Для произвольной АО-модели $S = \langle A, l : sig(A) \to L\rangle$ положим $T = mod^{**}(S)$, $s = mod(\kappa_S) : A \to T$, по определению АО-морфизма имеем $str(\kappa_S) \circ l = 1_{sig(T)} \circ sig(s) = sig(s)$. Поэтому тождество коединицы сопряжения $sig^* \dashv sig$ для s можно записать в виде $s \circ \varepsilon_A = \varepsilon_T \circ sig^*(sig(s)) = w \circ sig^*(l)$, где $w = \varepsilon_T \circ sig^*(str(\kappa_S))$. Проверим, что оно определяет кодекартов квадрат – копредел диаграммы *pmao*(*S*). Выберем произвольно пару *c-DESC*-морфизмов $a : A \to X \leftarrow sig^*(L) : y$ такую, что $a \circ \varepsilon_A = y \circ sig^*(l)$. Применяя функтор *sig* к этому равенству, получаем $sig(a) = sig(y) \circ l$, так что имеется *AO*-морфизм $f = \langle a, sig(y)\rangle : S \to mod^*(X)$. Тождество единицы для него имеет вид $f = mod^*(q) \circ \kappa_S$, где $q = mod^{**}(f) : T \to X$. Применяя к нему функтор *mod*, получаем, что $a = q \circ s$. Применяя к нему функтор *str*, получаем, что $sig(y) = str(mod^*(q)) \circ str(\kappa_S) = sig(q \circ w)$, откуда $y = q \circ w$ ввиду унивалентности функтора *sig*. Поэтому q – стрелка копредела (ее единственность вытекает из определения сопряжения: для любого *c-DESC*-морфизма q' соотношения $a = q' \circ s$ и $y = q' \circ w$ влекут $f = mod^*(q') \circ \kappa_S$, откуда $q' = q$). □



Следствие 4.3.1. Если некоторая АО-модель содержится в кообласти некоторого АО-расширения, то диаграмма ее частичной модуляризации имеет копредел, сохраняемый функтором *sig*. Его можно выбрать так, чтобы ребро, параллельное действию частичной модуляризации, было модуляризацией аспектной структуры модели.

Доказательство. Пусть $am : c\text{-}DESC \to m\text{-}AO$ – АО-расширение, η – единица сопряжения $am^* \dashv am$. Выберем произвольно АО-модель $S = \langle A, l : sig(A) \to L\rangle \in \mathrm{Ob}\ m\text{-}AO$, напомним, что $str(\eta_S) = 1_L$, так что $i \circ int(\eta_S) = l$, где $i = asp(am(am^*(S)))$ – изоморфизм (ввиду предложения 4.13). Аналогично второй половине доказательства теоремы 4.3, получаем, что равенство $mod(\eta_S) \circ \varepsilon_A = (\varepsilon_{am^*(S)} \circ sig^*(i^{-1})) \circ sig^*(l)$ (тождество коединицы для модуляризации аспектной структуры) задает кодекартов квадрат – копредел диаграммы частичной модуляризации АО-модели S (произвольная пара $c\text{-}DESC$-морфизмов $a : A \to X \leftarrow sig^*(L) : y$, удовлетворяющая условию $a \circ \varepsilon_A = y \circ sig^*(l)$, порождает AO-морфизм $\langle a, j \circ sig(y)\rangle : S \to am(X)$, где $j = asp(am(X)) \in \mathrm{Iso}\ SIG$. Функтор sig переводит этот кодекартов квадрат в коммутативный квадрат $int(\eta_S) \circ 1_{sig(A)} = i^{-1} \circ l$, две параллельных стороны которого являются изоморфизмами, так что он также кодекартов. □



Следствие 4.3.2. Модуляризация аспектной структуры АО-модели, если она существует, является регулярным эпиморфизмом.

Доказательство. Выводится из следствия 4.3.1, аналогично импликации (iv) ⇒ (ii) предложения 4.19. □

Следствие 4.3.3. Если категория *c-DESC* подвижна, то формальная технология проектирования *AR* обладает наибольшим АО-расширением, причем следующие утверждения эквивалентны для произвольной АО-модели $\langle A, l\rangle$:

(i) Она содержится в кообласти наибольшего АО-расширения.

(ii) Диаграмма ее частичной модуляризации имеет копредел, сохраняемый функтором *sig*.

(iii) Существует регулярный *c-DESC*-эпиморфизм с областью A, переходящий в l под действием функтора *sig*.

Доказательство. Мы воспользуемся способностью функтора *sig* поднимать копределы некоторых диаграмм. Пусть *max-AO* – полная подкатегория в *AO*, класс объектов которой состоит из всех АО-моделей, копределы диаграмм частичной модуляризации которых поднимает функтор *sig*. Он переводит диаграмму $pmao(S)$ частичной модуляризации произвольной АО-

модели $S = \langle A, l : sig(A) \to L \rangle$ в *SIG*-диаграмму $1_{sig(A)} : sig(A) \leftarrow sig(A) \to L : l$, которая имеет копредел, задаваемый коммутативным квадратом $l \circ 1_{sig(A)} = 1_L \circ l$. Ввиду предложения 2.10 S содержится в *max-AO*, если диаграмма $pmao(S)$ имеет копредел, сохраняемый функтором *sig*. В частности, АО-расширение с кообластью *max-AO*, если оно существует, ввиду следствия 4.3.1 является наибольшим.

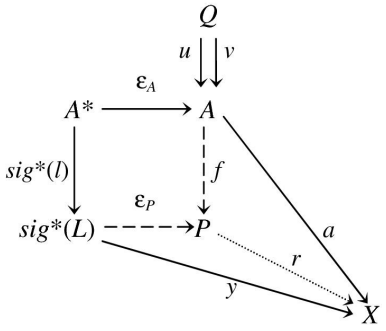
Убедимся в его существовании. Если $l = 1_{sig(A)}$, то диаграмма $pmao(S)$ имеет вид $\varepsilon_A : A \leftarrow A^* \to A^* : 1_{A^*}$, где $A^* = sig^*(sig(A))$, так что тождество $1_A \circ \varepsilon_A = \varepsilon_A \circ 1_{A^*}$ задает ее копредел, сохраняемый любым функтором, поэтому $S \in \mathrm{Ob}\ max\text{-}AO$. Следовательно, можно определить функтор, действующий из *c-DESC* в *max-AO* как *mod**. Он обладает правым сопряженным, действующим из *max-AO* в *c-DESC* как *mod*, и левым сопряженным, действующим как *pmod* (при построении которого условие $sig(pmod(-)) = str(-)$ обеспечивается путем выбора копредела диаграммы $pmao(S)$ так, чтобы функтор $\mathbf{D}sig$ переводил его в копредел диаграммы $sig \circ pmao(S)$, задаваемый коммутативным квадратом $l \circ 1_{sig(A)} = 1_L \circ l$), поэтому является АО-расширением.

Проверим эквивалентность условий (i)-(iii).

(i) $\Rightarrow$ (iii). Если $S \in \mathrm{Ob}\ max\text{-}AO$, то ввиду следствия 4.3.2 модуляризация t аспектной структуры АО-модели S является регулярным эпиморфизмом, а поскольку рассматриваемое АО-расширение действует как *mod**, имеем $sig(t) = l$.

(iii) $\Rightarrow$ (ii). Пусть $f : A \to P$ – коуравнитель некоторой пары *c-DESC*-морфизмов $u, v : Q \rightrightarrows A$, причем $sig(f) = l$. Проверим, что тождество коединицы $f \circ \varepsilon_A = \varepsilon_P \circ sig^*(l)$ задает кодекартов квадрат (тогда функтор *sig* сохраняет его – переводит в кодекартов квадрат $l \circ 1_{sig(A)} = 1_L \circ l$). Будем рассуждать по существу аналогично доказательству предложения 4.19. Выберем произвольную пару *c-DESC*-морфизмов $a : A \to X \leftarrow sig^*(L) : y$ такую, что $a \circ \varepsilon_A = y \circ sig^*(l)$. Имеем $sig(a \circ u) = sig(a \circ \varepsilon_A) \circ sig(u) = sig(y) \circ l \circ sig(u) = sig(y) \circ sig(f \circ u) = sig(y) \circ sig(f \circ v) = sig(a \circ v)$, откуда $a \circ u = a \circ v$ ввиду унивалентности функтора *sig*. По определению коуравнителя, существует единственный *c-DESC*-морфизм $r : P \to X$ такой, что $r \circ f = a$. Тогда $(r \circ \varepsilon_P) \circ sig^*(l) = r \circ f \circ \varepsilon_A = a \circ \varepsilon_A = y \circ sig^*(l)$, так что и $r \circ \varepsilon_P = y$ ввиду наличия у морфизма l правого обратного. Таким образом, r – искомая стрелка копредела.

(ii) $\Rightarrow$ (i). См. первый абзац доказательства. □



Отметим, что кообласть АО-расширения из следствия 4.3.3 замкнута относительно изоморфизмов, поскольку условие (ii) устойчиво относительно изоморфизмов АО-моделей: ввиду предложения 4.22 они порождают изоморфизмы диаграмм частичной модуляризации. Также отметим, что утверждение следствия 4.3.3 не допускает обращения: среди технологий проектирования, в которых категория моделей не является подвижной, встречаются как обладающие наибольшим АО-расширением, так и не обладающие им. Пример первого рода строится на базе категории $\boldsymbol{V}$ и ее морфизмов $q : 0 \to 1$, $q' : 0 \to 1'$. Обозначим через $\boldsymbol{V}^{\#}$ категорию, порожденную из $\boldsymbol{V}$ путем добавления морфизма $q^{-1} : 1 \to 0$, удовлетворяющего соотношениям $q \circ q^{-1} = 1_1$ и $q^{-1} \circ q = 1_0$ (а также производного морфизма $q' \circ q^{-1} : 1 \to 1'$). Пусть, как обычно, $\boldsymbol{2}^{\#}$ – полная подкатегория в $\boldsymbol{V}^{\#}$ с классом объектов $\{0, 1\}$, $vv^{\#} : \boldsymbol{V}^{\#} \to \boldsymbol{2}^{\#}$ – функтор, который действует на $\boldsymbol{2}^{\#}$ тождественно и переводит объект 1' в 0 (соответственно, морфизм q' в 1_0). Четверка $VV^{\#} = \langle \boldsymbol{V}^{\#}, \varnothing, vv^{\#}, (\boldsymbol{V}^{\#})^{\mathrm{op}} \rangle$ представляет собой аспектно тривиальную формальную технологию проектирования. Пусть av – ее произвольное АО-расширение. Рассуждая как в первом абзаце доказательства предложения 4.13, легко установить, что $av(P) = \langle P, 1_{vv\#(P)} \rangle$ для любого $\boldsymbol{V}^{\#}$-объекта P. Поэтому кообласть функтора av не содержит АО-модель $\langle 1', q \rangle$: в противном случае выполнялось бы соотношение $av^*(\langle 1_{1'}, q \rangle : \langle 1', 1_0 \rangle \to \langle 1', q \rangle) = 1_{1'}$, вступающее в противоречие со вторым пунктом определения 4.10, поскольку $vv^{\#}(1_{1'}) = 1_0 \neq q$. В свою очередь, аспектное ядро технологии $VV^{\#}$ содержит все АО-модели, кроме $\langle 1', q \rangle$, поэтому в силу теоремы 4.2 оно задает наибольшее АО-расширение технологии $VV^{\#}$ (несмотря на то, что ввиду предложения 4.21 условие подвижности для нее не выполняется). Ситуация кардинально изменится, если «склеить» объекты 0 и 1. Рассмотрим категорию $X_1 = (\{A, B\}, \{1_A, r : A \to A, 1_B : B \to B, p_1, q_1 : A \to B\})$, где $r \circ r = 1_A$, $p_1 \circ r = q_1$, полную подкатегорию в ней $Y = (\{A\}, \{1_A, r\})$ и функтор $xy_1 : X_1 \to Y : r \mapsto r, p_1 \mapsto r$ (мы используем обозначения, которые встретятся в разд.4.5). Четверка $XY = \langle X_1, \varnothing, xy_1, X_1^{\mathrm{op}} \rangle$ представляет собой аспектно тривиальную формальную технологию проектирования, обладающую по крайней мере двумя различными тривиальными АО-расширениями:

$$X_1 \cong (\{\langle A, 1_A \rangle, \langle B, 1_A \rangle\}, \{\langle 1_A, 1_A \rangle, \langle r, r \rangle, \langle 1_B, 1_A \rangle, \langle p_1, r \rangle, \langle q_1, 1_A \rangle\}),$$

$$X_1 \cong (\{\langle A, r \rangle,\ \langle B, r \rangle\ \}, \{\langle 1_A, 1_A \rangle, \langle r, r \rangle, \langle 1_B, 1_A \rangle, \langle p_1, r \rangle, \langle q_1, 1_A \rangle\}).$$

Объединение классов объектов кообластей этих АО-расширений представляет собой класс всех АО-моделей над XY, однако технология XY не обладает АО-расширением, кообласть которого содержит все АО-модели, по той же причине, по какой им не обладает $VV^{\#}$: левый сопряженный к такому АО-расширению переводил бы морфизм АО-моделей $\langle 1_B, r \rangle : \langle B, 1_A \rangle \to \langle B, r \rangle$ в 1_B. Таким образом, технология XY не обладает наибольшим АО-расширением.

4.5 Аспектно-ориентированный синтез технологий специфицирования

Как указывалось в разд.4.1, семантическая модель расширения технологий разработки систем аспектно-ориентированными приемами позволяет распознавать эти приемы (а значит, обеспечивать снижение затрат путем их целенаправленного привлечения) в условиях, когда использовать терминологию АОП не принято. В качестве примера такого «неявного» применения аспектно-ориентированного подхода мы рассмотрим синтез специализированных технологий специфицирования систем – краеугольный камень разработки, управляемой моделями (MDE). Покажем, что функтор выделения интерфейсов, дополняющий формальную технологию конфигурирования до технологии специфицирования, представляет собой в точности ее эксплицируемую разметку в формальной технологии синтеза технологий конфигурирования ***SCONF*** (см. предложение 2.22). Следовательно, *технологией проектирования технологий специфицирования* правомерно называть ядерную АО-технологию над ***SCONF*** (см. следствие 4.4.1 ниже). Получается, что категория интерфейсов отражает аспектную структуру своих реализаций (например ассортимент ролей, которые компоненты играют в составе систем [191]). Таким образом, можно снижать затраты на синтез сложных технологий специфицирования, пользуясь аспектным связыванием. Более того, мы докажем, что конструкция АО-технологии совместима с аспектной структурой: если модульная технология проектирования аспектно полна (в том смысле, что сборка систем в ней не разрушает аспектную структуру), то переход к АО-технологии над ней представляет собой трансформацию. В связи с этим введем следующее понятие.

Определение 4.17. Формальная технология проектирования *AR* называется *аспектно полной*, если класс *Conf* аспектно замкнут относительно функтора *mod*. □

Предложение 4.23. Следующие утверждения эквивалентны.

(i) Формальная технология *AR* аспектно полна.

(ii) Класс всех конфигураций любой АО-технологии над *AR* совпадает с $\mathbf{D}mod^{-1}(Conf)$.

(iii) Класс $\mathbf{D}mod^{-1}(Conf)$ состоит из аспектно детерминированных диаграмм и класс *Conf* замкнут относительно накачек *sig*-трассами.

(iv) Функтор *mod* задает трассу трансформации технологии конфигурирования $\boldsymbol{conf}(\boldsymbol{spec}(AR))$ в $\boldsymbol{conf}(\boldsymbol{spec}(AO_{ai}(AR)))$ для любого функтора *ai*, порождающего АО-технологию над *AR*.

(v) Любое АО-расширение технологии *AR* задает **CONF**-морфизм технологии $\boldsymbol{conf}(\boldsymbol{spec}(AR))$ в $\boldsymbol{conf}(\boldsymbol{spec}(AO_{ai}(AR)))$ для любого функтора *ai*, порождающего АО-технологию над *AR*.

Доказательство.

(i) ⇒ (ii). Пусть *ai* – произвольный функтор, порождающий АО-технологию над *AR*. Согласно следствию 4.1.3 и определению 4.4, имеем $\mathbf{D}int^{-1}(AOInt_{int}) \subseteq \mathbf{D}ai^{-1}(AOInt_{ai}) \subseteq \mathbf{D}mod^{-1}(Conf)$. Поскольку $\mathbf{D}int^{-1}(sig \circ Conf) = \mathbf{D}mod^{-1}(Conf)$, по предположению $sig \circ Conf \subseteq AOInt_{int}$, так что $\mathbf{D}mod^{-1}(Conf) \subseteq \mathbf{D}ai^{-1}(AOInt_{ai}) \subseteq \mathbf{D}mod^{-1}(Conf)$.

(ii) ⇒ (iii). Пусть $\tau \rightrightarrows \Sigma$ – накачка произвольной диаграммы $\Sigma \in Conf$ произвольным семейством трасс τ. Рассмотрим диаграмму $\Omega = mod^* \circ (\tau \rightrightarrows \Sigma) = mod^*(\tau) \rightrightarrows mod^* \circ \Sigma$. Это – накачка диаграммы $mod^* \circ \Sigma \in \mathbf{D}mod^{-1}(Conf)$ семейством *tr-AO*-морфизмов $mod^*(\tau)$. По предположению, $AOInt_{1_{AO}} = \mathbf{D}mod^{-1}(Conf)$, поэтому $\Omega \in \mathbf{D}mod^{-1}(Conf)$, так что $\tau \rightrightarrows \Sigma = mod \circ \Omega \in Conf$.

(iii) ⇒ (i). Ввиду определения 4.4 требуется показать, что $\omega \rightrightarrows \Delta \in \mathbf{D}mod^{-1}(Conf)$, где ω – произвольное семейство *tr-AO*-морфизмов, Δ – произвольная *AO*-диаграмма такая, что $mod \circ \Delta \in Conf$. Действительно, поскольку семейство $mod(\omega)$ состоит из *sig*-трасс, по предположению $mod \circ (\omega \rightrightarrows \Delta) = mod^*(\omega) \rightrightarrows mod \circ \Delta \in \mathbf{D}mod^{-1}(Conf)$.

(ii) ⇒ (v). Пусть $am : c\text{-}DESC \to m\text{-}AO$ – произвольное АО-расширение технологии *AR*, $mao : m\text{-}AO \hookrightarrow AO$ – полное вложение. Поскольку ввиду предложения 4.12 для любой диаграммы $\Theta \in Conf$ имеем $mod \circ (mao \circ am \circ \Theta) = am^* \circ am \circ \Theta = \Theta$, а ввиду предложения 4.13 функтор $mao \circ am$ сохраняет копределы всех *c-DESC*-диаграмм, имеется **CONF**-морфизм $mao \circ am : \langle c\text{-}DESC, Conf\rangle \to \langle AO, \mathbf{D}mod^{-1}(Conf)\rangle$.

(v) ⇒ (i). По предположению, тривиальное АО-расширение $c\text{-}DESC \cong mod^*(c\text{-}DESC)$ порождает **CONF**-морфизм $mod^* : \langle c\text{-}DESC, Conf\rangle \to \langle AO, \mathbf{D}mod^{-1}(AOInt_{mod})\rangle$, т.е. $Conf = mod \circ (mod^* \circ Conf) \subseteq AOInt_{mod}$.

(iv) ⇔ (ii). Вытекает из теоремы 4.1 и определения класса *r-CONF* (см. разд.2.6). □

Например, технология моделирования сценариев *SM* аспектно полна (см. следствие 5.1.6), а технология ***SCONF*** – нет (см. ниже). Для аспектно тривиальных технологий имеется следующий критерий аспектной полноты.

Предложение 4.24. Аспектно тривиальная технология *AR* аспектно полна тогда и только тогда, когда класс *Conf* замкнут относительно накачек *sig*-трассами.

Доказательство. Предположим, что *AR* аспектно тривиальна. Ввиду предложения 4.23 нужно показать, что произвольная диаграмма $\Delta : X \to AO$ такая, что $mod \circ \Delta \in Conf$, аспектно детерминирована. Пусть $\theta : mod \circ \Delta \to \ulcorner A \urcorner$ – копредел. Ввиду предложения 2.6 $sig \circ \theta$ – копредел диаграммы $int \circ \Delta$. А поскольку ввиду предложения 4.2 естественное преобразование β^{Ξ}, где $\Xi = il \circ asp \circ \Delta$, состоит из изоморфизмов, коконус $\gamma = (sig \circ \theta) \circ \langle \beta^{\Xi}, 1_X\rangle^{-1} : str \circ \Delta \to \ulcorner sig(A) \urcorner$ представляет собой копредел диаграммы $str \circ \Delta$. Если υ – ее произвольный копредел, то суще-

ствует *SIG*-изоморфизм i такой, что $\upsilon = \ulcorner i \urcorner \circ \gamma$, следовательно коконус $\langle sig \circ \theta, \upsilon \rangle : \Xi \to \ulcorner i \urcorner$ представляет собой копредел SIG^2-диаграммы Ξ. А поскольку любой изоморфизм является *sig*-разметкой (предложение 4.1), $\langle \theta, \upsilon \rangle : \Delta \to \ulcorner \langle A, i \rangle \urcorner$ – копредел. Таким образом, функтор $\langle mod, str \rangle : AO \to c\text{-}DESC \times SIG$ поднимает копределы диаграммы Δ, так что ввиду предложения 2.2 она аспектно детерминирована. □

Обозначим через *AOSC* произвольную АО-технологию над ***SCONF***. Как мы сейчас увидим, в ней достаточно мало аспектов – нетривиальные интерфейсы неразделимо связаны со своими реализациями. Из практики программирования хорошо известны трудности, возникающие при попытке избежать изменения интерфейсов при изменении реализаций (несмотря на то, что разработчики многих технологий программирования заявляют об обеспечении независимости интерфейса от реализации). Аспекты в *AOSC* имеют довольно бедную структуру предпорядка с наименьшим элементом. Технологии проектирования, получающиеся из них, являются «самыми» аспектно тривиальными: класс разметок в них состоит из единственного тождественного морфизма, поэтому категория АО-моделей изоморфна исходной категории моделей. Примером служит технология TS_σ, введенная в разд.2.3. Разделение ответственности всегда тривиально: оно возможно только для формальных технологий специфицирования, имеющих не более одного подаспекта.

Теорема 4.4.

(i) Формальная технология конфигурирования *CS* может служить модульной основой аспекта в *AOSC* тогда и только тогда, когда категория ***desc***(*CS*) представляет собой предпорядок, имеющий наименьший элемент.

(ii) АО-модель над ***SCONF*** обладает экспликацией тогда и только тогда, когда она является формальной технологией специфицирования; любая ее экспликация универсальна.

(iii) Трансформация в *AOSC* обладает экспликацией тогда и только тогда, когда ее область и кообласть являются формальными технологиями специфицирования.

Доказательство. Зафиксируем произвольную формальную технологию специфицирования $SC = \langle c\text{-}DESC, Conf, sig : c\text{-}DESC \to SIG \rangle$, положим $SConf = sig \circ Conf$.

(i). Как отмечалось в доказательстве предложения 2.22, технология ***SCONF*** элементарна. Класс всех ***desc***-разметок состоит из всех унивалентных функторов, имеющих левый сопряженный с тождественной единицей. Теперь применяем предложение 4.6 и доказательство условия (i) предложения 2.13.

(ii). Морфизм формальных технологий конфигурирования $sig : \langle c\text{-}DESC, Conf \rangle \to \langle SIG, SConf \rangle \in r\text{-}CONF$ является трассой единственной экспликации технологии *SC*. Эта трансформация обратима, поскольку функтор *sig**, правый обратный к *sig*, сохраняет все копределы

и потому задает **CONF**-морфизм $sig^* : \langle SIG, SConf\rangle \to \langle c\text{-}DESC, Conf\rangle$ (таким образом, технология ***SCONF*** поддерживает трассирование). Ввиду предложения 4.20 экспликация технологии *SC* универсальна.

(iii). Пусть $\langle cm, sm\rangle : SC \to \langle c\text{-}DESC', Conf', sig' : c\text{-}DESC' \to SIG'\rangle$ – **SPEC**-морфизм такой, что $cm \in r\text{-}CONF$, в частности $sm \circ sig = sig' \circ cm$ и $Conf' = cm \circ Conf$. Ввиду уже доказанного утверждения (ii), имеется **CONF**-морфизм $sm : \langle SIG, SConf\rangle \to \langle SIG', sig' \circ Conf'\rangle$. Требуется доказать, что $sm \in r\text{-}CONF$. Рассмотрим функтор $scm = sig' \circ cm \in r\text{-}CONF$. Функтор $scm^* = cm^* \circ sig'^*$ является левым сопряженным к нему (мы будем обозначать через ν коединицу этого сопряжения). Из равенства $sm \circ sig = scm$ вытекает, что $sm \circ SConf = scm \circ Conf = sig' \circ Conf'$.

Из него же следует, что $sm = scm \circ sig^*$, так что функтор sm унивалентен (будучи композицией унивалентных функторов), и функтор $sig \circ scm^*$ является левым сопряженным к нему с тождественной единицей (действительно, $sm \circ sig \circ scm^* = scm \circ scm^* = 1_{SIG'}$ и для произвольных $A \in \mathrm{Ob}\, SIG'$, $X \in \mathrm{Ob}\, SIG$, $f : A \to sm(X)$, существует, причем единственный в силу унивалентности функтора sm, SIG-морфизм $g : sig(scm^*(A)) \to X$, удовлетворяющий условию $sm(g) = f$: положим $g = sig(\nu_{sig^*(X)} \circ scm^*(f))$, тогда $sm(g) = scm(\nu_{sig^*(X)} \circ scm^*(f)) = 1_{scm(sig^*(X))} \circ scm(scm^*(f)) = f$).

Далее, в силу предложения 2.3 функтор sm поднимает копределы всех диаграмм из класса $SConf$. Наконец, пусть произвольные диаграммы $\Delta \in Conf$ и $\Xi \in \mathrm{Ob}\, \mathbf{D}SIG$ таковы, что $sm \circ (sig \circ \Delta) = sm \circ \Xi$. Тогда $scm \circ \Delta = scm \circ sig^* \circ \Xi$, откуда $sig^* \circ \Xi \in Conf$, следовательно $\Xi = sig \circ sig^* \circ \Xi \in SConf$. □

Следствие 4.4.1. Ядерная АО-технология над ***SCONF*** имеет категорию моделей **SPEC** и функтор выделения интерфейсов ***conf***. □

Следствие 4.4.2. Пусть выбраны произвольно:

- формальная технология специфицирования $SC_1 = \langle c\text{-}DESC_1, Conf_1, sig_1 : c\text{-}DESC_1 \to SIG_1\rangle$;
- пара функторов $cs = \langle cm : c\text{-}DESC_1 \to c\text{-}DESC_2, sm : SIG_1 \to SIG_2\rangle$;
- тройка $SC_2 = \langle c\text{-}DESC_2, Conf_2, sig_2 : c\text{-}DESC_2 \to SIG_2\rangle$.

Тройка SC_2 является формальной технология специфицирования, а пара cs – трассой ее трансформации в SC_1, тогда и только тогда, когда выполняются следующие условия:

(i) Тройка $\langle c\text{-}DESC_1, Conf_1, cm\rangle$ является формальной технологией специфицирования.

(ii) Тройка $\langle SIG_1, sig_1 \circ Conf_1, sm\rangle$ является формальной технологией специфицирования.

(iii) $cm \circ Conf_1 = Conf_2$.

(iv) Условие (iii) определения 2.13.

Доказательство. Необходимость условия (ii) вытекает из утверждения (iii) теоремы 4.4, а остальных условий – из определений. Для доказательства достаточности выведем из условий (i)-(iv) условия (i)-(v) определения 2.3 для тройки SC_2. Наличие копредела у любой диаграммы из класса $Conf_2$ вытекает из условий (iii) и (i) ввиду предложения 2.6. Остальные из искомых условий определения 2.3 вытекают из утверждения (iii) теоремы 4.4, поскольку ввиду условия (iv) пара $\langle sig_1, sig_2\rangle$ является трассой трансформации технологии специфицирования из условия (ii) в технологию из условия (i). □

Следствие 4.4.3. Технология ***SCONF*** обладает наибольшим АО-расширением, кообласть которого состоит из всех троек вида $\langle c\text{-}DESC, Conf, sig\rangle$, удовлетворяющих условиям (i)-(iii) определения 2.3 и следующему аналогу условия (iv): функтор *sig* сохраняет копределы всех диаграмм из класса *Conf*.

Доказательство. Ввиду подвижности категории **CONF** и следствия 4.3.3, ***SCONF*** обладает наибольшим АО-расширением, кообласть которого состоит из всех троек $\langle c\text{-}DESC, Conf, sig\rangle$, удовлетворяющих условиям (i)-(iii) определения 2.3 и следующему условию: имеется регулярный **CONF**-эпиморфизм $sig : \langle c\text{-}DESC, Conf\rangle \to \langle SIG, CConf\rangle$ для некоторого класса *SIG*-диаграмм *CConf*. Ввиду определения 2.13, чтобы функтор *sig* задавал **CONF**-морфизм с областью $\langle c\text{-}DESC, Conf\rangle$, необходимо, чтобы он сохранял копределы всех диаграмм из класса *Conf*. Проверим, что это условие является и достаточным: **CONF**-морфизм $sig : \langle c\text{-}DESC, Conf\rangle \to \langle SIG, sig \circ Conf\rangle$, если он существует, является регулярным эпиморфизмом, а именно коуравнителем пары **CONF**-морфизмов $1_{c\text{-}DESC}, (sig^* \circ sig) : \langle c\text{-}DESC, \varnothing\rangle \rightrightarrows \langle c\text{-}DESC, Conf\rangle$. Действительно, имеем $sig \circ 1_{c\text{-}DESC} = sig \circ (sig^* \circ sig)$, и для любых технологии конфигурирования *DX* и **CONF**-морфизма $dx : \langle c\text{-}DESC, Conf\rangle \to DX$ такого, что $dx \circ 1_{c\text{-}DESC} = dx \circ (sig^* \circ sig)$, функтор $ds = dx \circ sig^*$ задает **CONF**-морфизм технологии $\langle SIG, sig \circ Conf\rangle$ в *DX*, поскольку, как мы сейчас увидим, он сохраняет копределы всех диаграмм из класса $sig \circ Conf$. Выберем произвольно диаграмму $\Theta \in Conf$, положим $\Delta = sig \circ \Theta$, $\delta = \operatorname{colim} \Delta$. Пусть $\theta = \operatorname{colim} \Theta$, так что $sig \circ \theta$ – копредел диаграммы Δ, поэтому существует *SIG*-изоморфизм *i* такой, что $\delta = \ulcorner i \urcorner \circ (sig \circ \theta)$. Следовательно, $ds \circ \delta = \ulcorner ds(i) \urcorner \circ (dx \circ \theta)$, а поскольку функтор *dx* выбран так, что он сохраняет копределы всех диаграмм из класса *Conf*, диаграмма $ds \circ \Delta$ имеет копредел $ds \circ \delta$. □

Следствие 4.4.4. Непустая формальная технология специфицирования обладает разделением ответственности тогда и только тогда, когда она является аспектом в *AOSC*.

Доказательство. Допустим, что технология специфицирования $\langle c\text{-}DESC, Conf, sig\rangle$ обладает разделением ответственности и категория *c-DESC* непуста. В силу следствия 4.2.3 в

AOSC существует аспект вида $\langle c\text{-}DESC, Conf, ss\rangle$ (с подходящим функтором *ss*), следовательно, ввиду утверждения (i) теоремы 4.4, категория *c-DESC* связна. Поэтому рассматриваемая технология содержит в точности один подаспект, который в силу предложения 4.18 является изоморфизмом. □

Следствие 4.4.5. Пусть $AR = \langle c\text{-}DESC, Conf, sig, r\text{-}DESC\rangle$ – произвольная формальная технология проектирования. Для любого функтора *ai*, порождающего АО-технологию над ней, функтор $mod : AO \to c\text{-}DESC$, построенный из ее компонентов, индуцирует **ARCH**-морфизм, действующий из $AO_{ai}(AR)$ в *AR*. Технология *AR* аспектно полна тогда и только тогда, когда для любого порождающего функтора *ai* ***spec***-образ этого морфизма является трассой эксплицируемой трансформации технологий специфицирования.

Доказательство. Положим $ao_{ai} = \langle mod, int \circ ai^*, mod(-^{op})^{op}\rangle$. Ввиду условия (ii) следствия 4.1.1 и определения 4.4 эта тройка представляет собой **ARCH**-морфизм, действующий из $AO_{ai}(AR)$ в *AR*. Далее применяем предложение 4.23 (эквивалентность условий (i) и (iv)) и теорему 4.4 (утверждение (iii)). □

Следствие 4.4.6. Для любой формальной технологии проектирования $AR = \langle c\text{-}DESC, Conf, sig, r\text{-}DESC\rangle$ и любого функтора *ai*, порождающего АО-технологию над *AR*, тройка $\langle \mathrm{codom}\ ai, ai \circ \mathbf{D}int^{-1}(AOInt_{int}), int \circ ai^*\rangle$ представляет собой формальную технологию специфицирования с категорией интерфейсов *SIG*, трансформируемую в $\boldsymbol{spec}(AO_{int}(AR))$ при помощи функтора *ai*.

Доказательство. Построим из компонентов АО-технологии над *AR* следующую четверку:

$$AOI = \langle AO, \mathbf{D}int^{-1}(AOInt_{int}), ai, tr\text{-}AO^{op}\rangle.$$

Она является формальной технологией проектирования (условие (iv) определения 2.3 для нее вытекает из следствия 4.1.3, а условие (v) проверяется так: для любых *AO*-диаграмм Δ, Ξ, если $ai \circ \Delta = ai \circ \Xi$ и $\Delta \in \mathbf{D}int^{-1}(AOInt_{int})$, то $int \circ \Xi = int \circ ai^* \circ ai \circ \Xi = int \circ ai^* \circ ai \circ \Delta = int \circ \Delta \in AOInt_{int}$). Имеется морфизм формальных технологий

$$aoi = \langle 1_{AO}, int \circ ai^*, 1_{tr\text{-}AO^{op}}\rangle : AOI \to AO_{int}(AR).$$

Обозначим через ψ единицу сопряжения $\boldsymbol{conf}^{**} \dashv \boldsymbol{conf}^*$ (напомним, что ввиду следствия 4.4.1 и теоремы 4.2 функтор $\boldsymbol{conf}^*$ имеет левый сопряженный), положим $\boldsymbol{dspec} = \boldsymbol{conf}^* \circ \boldsymbol{conf}^{**} \circ \boldsymbol{spec}$. Рассмотрим **SPEC**-морфизм

$$\boldsymbol{dspec}(aoi) \circ \psi_{\boldsymbol{spec}(AOI)} = \langle int, int \circ ai^*\rangle : \boldsymbol{spec}(AOI) \to \langle SIG, AOInt_{int}, 1_{SIG}\rangle.$$

По теореме 4.1 он является трассой трансформации технологий специфицирования. Из следствия 4.4.2 (см. условие (ii) в его формулировке) вытекает, что тройка, указанная в условии до-

казываемого утверждения, действительно является технологией специфицирования. Пара $\langle ai, 1_{SIG}\rangle$ представляет собой трассу ее трансформации в $\boldsymbol{spec}(AO_{int}(AR))$. □

Следствие 4.4.6 раскрывает «финальный» характер функтора *int* по отношению к другим функторам, порождающим АО-технологии: интерфейсы АО-моделей образуют технологии специфицирования, которые можно трансформировать в АО-технологию, порожденную им. Рассмотрим, какие технологии специфицирования интерфейсов дают фундаментальные функторы:

- при $ai = 1_{AO}$ получаем технологию $\boldsymbol{spec}(AO_{int}(AR))$ с тождественной трансформацией в себя;
- при $ai = mod$, если AR аспектно полна, то ввиду предложения 4.23 получается технология $\boldsymbol{spec}(AR)$ и трансформация $\boldsymbol{spec}(ao_{int})^{\mathrm{op}}$ из следствия 4.4.5;
- при $ai = int$ получаем технологию $\boldsymbol{dspec}(AO_{int}(AR))$, трансформируемую в $\boldsymbol{spec}(AO_{int}(AR))$ посредством универсальной экспликации последней;
- при $ai = asp$ получается структурируемая технология $\langle LAB, asp \circ \mathbf{D}int^{-1}(AOInt_{int}), dom \circ il\rangle$, которую естественно назвать *технологией специфицирования разметок* над AR: в ней в общем случае меньше конфигураций, чем в $\boldsymbol{dspec}(AO_{asp}(AR))$, но зато интерфейсом любой разметки выступает ее область (а не она сама, как в $\boldsymbol{dspec}(AO_{asp}(AR))$).

Любопытно, что формальная технология ***SCONF*** сама аспектно неполна. Чтобы убедиться в этом, построим **SPEC**-диаграмму Ξ, обладающую копределом в категории $\boldsymbol{desc} \downarrow \mathbf{CAT}$, вершина которого не принадлежит категории моделей технологии *AOSC*, несмотря на то, что **CONF**-диаграмма $\boldsymbol{conf} \circ \Xi$ является конфигурацией в ***SCONF***. Таким образом, этот пример также покажет, что технология проектирования технологий специфицирования не скоординирована. Рассмотрим следующие категории и функторы:

$X = (\{A, B\}, \{1_A, r : A \to A, 1_B : B \to B, p_1, p_2, q_1, q_2 : A \to B\})$, где $r \circ r = 1_A$, $p_i \circ r = q_i$ $(i = 1, 2)$,

$X_i = (\{A, B\}, \{1_A, r, 1_B, p_i, q_i\}) \subseteq X$ $(i = 1, 2)$,

$X_0 = (\{A, B\}, \{1_A, r, 1_B\}) \subseteq X$,

$Y = (\{A\}, \{1_A, r\}) \subseteq X_0$,

$ex_i : X_i \hookrightarrow X$ $(i = 0, 1, 2)$,

$jx_i : X_0 \hookrightarrow X_i$ $(i = 1, 2)$,

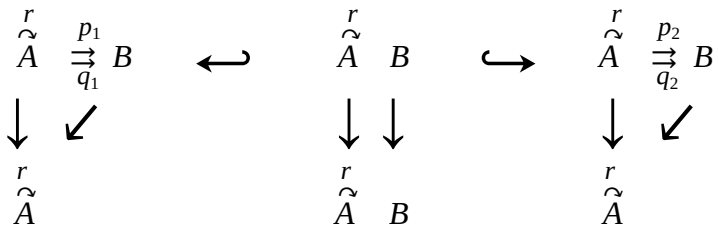
$yx : Y \hookrightarrow X_0$,

$xy : X \to Y : A \mapsto A, B \mapsto A, r \mapsto r, p_i \mapsto r, q_i \mapsto 1_A$ $(i = 1, 2)$,

$xy_i = xy \circ ex_i$ $(i = 0, 1, 2)$.

Любой из функторов xy_i ($i = 1, 2$) унивалентен, и функтор $jx_i \circ yx$ сопряжен слева к нему с тождественной единицей. Обозначим через Ξ следующую **SPEC**-диаграмму, состоящую из вложений формальных технологий:

$\langle jx_1, xy_0\rangle : \langle X_1, \varnothing, xy_1\rangle \hookleftarrow \boldsymbol{conf}^*(\boldsymbol{desc}^*(X_0)) \hookrightarrow \langle X_2, \varnothing, xy_2\rangle : \langle jx_2, xy_0\rangle$.



Пусть $\boldsymbol{dc} : \mathbf{SPEC} \hookrightarrow \boldsymbol{desc} \downarrow \mathbf{CAT}$ – вложение категорий. Диаграмма $\boldsymbol{dc} \circ \Xi$ обладает копределом в $\boldsymbol{desc} \downarrow \mathbf{CAT}$ с ребрами

$$\langle ex_1, 1_Y\rangle : \langle X_1, \varnothing, xy_1\rangle \hookrightarrow \langle X, \varnothing, xy\rangle \hookleftarrow \langle X_2, \varnothing, xy_2\rangle : \langle ex_2, 1_Y\rangle.$$

Функтор xy не унивалентен, поэтому $colim(\boldsymbol{dc} \circ \Xi)$ не принадлежит категории моделей технологии *AOSC*, так что диаграмма Ξ не является аспектно детерминированной. В то же время **CONF**-диаграмма $\boldsymbol{conf} \circ \Xi$ является конфигурацией в $\boldsymbol{SCONF}$. Действительно, рассмотрим произвольную диаграмму $\Delta : Z \to X_1$. Если ее схема пуста, то она не имеет копредела, поскольку ни A, ни B не являются инициальными объектами в X_1. Если категория Z непуста и связна, то Δ имеет копредел тогда и только тогда, когда для всякого $I \in \Delta^{-1}(A)$ каждое из следующих множеств состоит не более чем из одного элемента: (i) для всякого $J \in \Delta^{-1}(A)$ множество $\Delta(\mathrm{Mor}(I, J))$, (ii) множество $\bigcup_{J \in \Delta^{-1}(B)} \Delta(\mathrm{Mor}(I, J))$ (при этом если $\Delta(\mathrm{Ob}\ Z) = \{A\}$, то $colim(\Delta) = A$, а в противном случае $colim(\Delta) = B$). Если же категория Z дискретна и состоит более чем из одного объекта, то Δ имеет копредел тогда и только тогда, когда $\Delta(\mathrm{Ob}\ Z) = \{B\}$ (и тогда $colim(\Delta) = B$). В любом из описанных случаев, если диаграмма Δ обладает копределом, то функтор ex_1 сохраняет его. Аналогично проверяется, что функтор ex_2 сохраняет копределы всех X_2-диаграмм. Из предложения 2.21 получаем, что диаграмма $\boldsymbol{desc} \circ \boldsymbol{conf} \circ \Xi$ является $\boldsymbol{desc}$-предконфигурацией.

Построим формальную технологию синтеза технологий проектирования. Для них технологии специфицирования не могут служить интерфейсами, поскольку функтор $\boldsymbol{spec} : \mathbf{ARCH} \to \mathbf{SPEC}$, «забывающий» категорию трансформаций, не унивалентен. (Тем не менее, он обладает левым сопряженным

$$\boldsymbol{spec}^* : \mathbf{SPEC} \to \mathbf{ARCH} : \langle c\text{-}DESC, Conf, sig\rangle \mapsto \langle c\text{-}DESC, Conf, sig, \mathrm{Iso}\ c\text{-}DESC\rangle,$$

причем единица этого сопряжения тождественна, и $\boldsymbol{spec}^* \circ \boldsymbol{conf}^* \circ \boldsymbol{desc}^* = \boldsymbol{triv}$). Целесообразно извлекать интерфейсы из технологий проектирования таким же способом, каким функтор $\boldsymbol{conf}$

извлекает интерфейсы из технологий специфицирования – путем «забывания» функтора выделения интерфейсов (sic!). Поэтому интерфейсом технологии проектирования является любая тройка $\langle c\text{-}DESC, Conf, r\text{-}DESC\rangle$, удовлетворяющая условиям (i), (vi)-(viii) определения 2.3. В [221] такие тройки названы *формализмами проектирования* (design formalisms). Морфизмом формализма $\langle c\text{-}DESC_1, Conf_1, r\text{-}DESC_1\rangle$ в формализм $\langle c\text{-}DESC_2, Conf_2, r\text{-}DESC_2\rangle$ служит любая пара функторов $\langle cm : c\text{-}DESC_1 \to c\text{-}DESC_2, rm : r\text{-}DESC_1 \to r\text{-}DESC_2\rangle$, удовлетворяющая условиям (i), (ii), (iv) определения 2.13. Обозначим категорию всех формализмов и всех их морфизмов через **FORM**. Имеются забывающие функторы

$$\boldsymbol{form} : \mathbf{ARCH} \to \mathbf{FORM} : \langle c\text{-}DESC, Conf, sig, r\text{-}DESC\rangle \mapsto \langle c\text{-}DESC, Conf, r\text{-}DESC\rangle,$$

$$\boldsymbol{fconf} : \mathbf{FORM} \to \mathbf{CONF} : \langle c\text{-}DESC, Conf, r\text{-}DESC\rangle \mapsto \langle c\text{-}DESC, Conf\rangle,$$

удовлетворяющие соотношению $\boldsymbol{fconf} \circ \boldsymbol{form} = \boldsymbol{conf} \circ \boldsymbol{spec}$. Конфигурации технологий проектирования отбираются из потенциала комплексируемости функтора ***form*** по критерию совместимости с конфигурациями технологий специфицирования: введем класс **CONF**-диаграмм

$$\begin{aligned} PForm = \{\Delta \mid\ & \mathbf{D}(\boldsymbol{fconf})^{-1}(\{\Delta\}) \subseteq FPConf, \\ & \mathbf{D}(\boldsymbol{conf})^{-1}(\{\Delta\}) \subseteq CSConf, \\ & \mathbf{D}(\boldsymbol{conf} \circ \boldsymbol{spec})^{-1}(\{\Delta\}) \subseteq PCspec\}, \end{aligned}$$

где *FPConf* – класс всех ***form***-предконфигураций, *CSConf* – класс всех конфигураций в технологии проектирования технологий специфицирования, *PCspec* – класс всех **ARCH**-диаграмм, копределы которых сохраняет функтор ***spec***. Заметим, что класс *PForm* содержит все дискретные **CONF**-диаграммы (см. явную конструкцию суммы в **ARCH**, приведенную в разд.2.6). Трансформации технологий проектирования расширяют трансформации технологий специфицирования так, чтобы обеспечить трассируемость – для этого они должны быть согласованы с дискретной реализацией интерфейсов:

Определение 4.18. **ARCH**$^{\mathrm{op}}$-морфизм $\langle cm, sm, rm\rangle^{\mathrm{op}} : AR \to AR'$ называется *трансформацией технологий проектирования*, если он удовлетворяет следующим условиям.

(i) Пара $\langle cm, sm\rangle$ представляет собой трассу трансформации технологии специфицирования $\boldsymbol{spec}(AR)$ в $\boldsymbol{spec}(AR')$.

(ii) Существует функтор *drm*, правый обратный к *rm* и удовлетворяющий условию $drm(i) = cm^*(i)$ для любого $i \in \mathrm{Iso}\ \mathrm{codom}\ cm$. □

Например, двойственным к трансформации является **ARCH**-морфизм $ao_{ai} : AO_{ai}(AR) \to AR$, построенный в доказательстве следствия 4.4.5, при условии, что технология *AR* аспектно полна и все трансформации в ней трассируемы. При этом трансформация $ao_{int}{}^{\mathrm{op}}$ обратима: ввиду предложения 4.23 (импликация (i) ⇒ (v)) **ARCH**-морфизм, правый обратный к ее трассе, индуцируется AO-расширением – он имеет вид $\langle am(-), 1_{SIG}, am(-^{\mathrm{op}})^{\mathrm{op}}\rangle : AR \to AO_{int}(AR)$, где *am* –

произвольное АО-расширение технологии *AR*. Трансформация ao_{int}^{op} является первичной в том смысле, что любую трансформацию вида ao_{ai}^{op} можно выполнить в два шага, порождая из *AR* технологию $AO_{int}(AR)$ на первом шаге:

Предложение 4.25. Для любой аспектно полной формальной технологии проектирования $AR = \langle c\text{-}DESC, Conf, sig, r\text{-}DESC\rangle$ и любого функтора *ai*, порождающего АО-технологию над *AR*, существует трасса трансформации технологий проектирования $aoint_{ai} : AO_{ai}(AR) \to AO_{int}(AR)$, удовлетворяющая условию $ao_{int} \circ aoint_{ai} = ao_{ai}$.

Доказательство. Положим $aoint_{ai} = \langle 1_{AO}, int \circ ai^*, 1_{tr\text{-}AO^{\mathrm{op}}}\rangle$. В силу предложения 4.23 имеем $\boldsymbol{conf}(\boldsymbol{spec}(aoint_{ai})) \in r\text{-}CONF$. □

Для трансформаций технологий проектирования справедлив следующий аналог следствия 4.4.2.

Предложение 4.26. Пусть выбраны произвольно:

- формальная технология проектирования $AR_1 = \langle c\text{-}DESC_1, Conf_1, sig_1 : c\text{-}DESC_1 \to SIG_1, r\text{-}DESC_1\rangle$;
- тройка функторов $csr = \langle cm : c\text{-}DESC_1 \to c\text{-}DESC_2, sm : SIG_1 \to SIG_2, rm : r\text{-}DESC_1 \to r\text{-}DESC_2\rangle$;
- четверка $AR_2 = \langle c\text{-}DESC_2, Conf_2, sig_2 : c\text{-}DESC_2 \to SIG_2, r\text{-}DESC_2\rangle$.

Четверка AR_2 является формальной технологией проектирования, а тройка *csr* – двойственной к ее трансформации в AR_1, тогда и только тогда, когда выполняются следующие условия:

(i) Тройка $\langle c\text{-}DESC_2, Conf_2, sig_2\rangle$ является формальной технологией специфицирования, а пара $\langle cm, sm\rangle$ – трассой ее трансформации в $\boldsymbol{spec}(AR_1)$.

(ii) Условие (ii) определения 4.18 (для пары $\langle cm, rm\rangle$).

(iii) Условие (iv) определения 2.13.

Доказательство. Необходимость выполнения указанных условий непосредственно вытекает из определения 4.18. Для доказательства достаточности выведем из них условия (vi)-(viii) определения 2.3 для четверки AR_2.

(vi). Ввиду условий (ii) и (iii) любой $r\text{-}DESC_2$-объект *A* можно представить в виде $A = rm(drm(A)) = cm(drm(A)) \in \mathrm{Ob}\ c\text{-}DESC_2$.

(vii). Ввиду условий (i) и (iii) любой $c\text{-}DESC_2$-изоморфизм *i* можно представить в виде $i = cm(cm^*(i)) = rm(cm^*(i)) \in \mathrm{Mor}\ r\text{-}DESC_2$.

(viii). Выберем произвольно диаграмму $\Delta \in Conf_2$ и естественное преобразование $\varphi : |\Delta| \to \Sigma \in \mathrm{Mor}\ r\text{-}DESC_2^{|sch(\Delta)|}$. Ввиду условия (i) имеем $cm^* \circ \Delta \in Conf_1$, а ввиду условия (ii) – $drm(\varphi) \in \mathrm{Mor}\ r\text{-}DESC_1^{|sch(\Delta)|}$. Поскольку AR_1 является формальной технологией проектирования, имеется диаграмма $(cm^* \circ \Delta) \oplus drm(\varphi) \in Conf_1$, которую мы обозначим через Ξ, и $r\text{-}DESC_1$-

морфизм $r : cm^*(colim(\Delta)) \to colim(\Xi)$. В качестве $\Delta \oplus \varphi$ можно взять диаграмму $cm \circ \Xi$, поскольку она ввиду условия (i) принадлежит классу $Conf_2$, содержит Σ в качестве поддиаграммы и обладает копределом $cm \circ \operatorname{colim} \Xi$, так что имеется $r\text{-}DESC_2$-морфизм $cm(r) : colim(\Delta) \to colim(cm \circ \Xi)$. □

Обозначим через *r-ARCH* класс всех **ARCH**-морфизмов, двойственных к трансформациям технологий проектирования. Положим

$$\boldsymbol{SARCH} = \langle \mathbf{ARCH}, \mathbf{D}(\boldsymbol{conf} \circ \boldsymbol{spec})^{-1}(PForm), \boldsymbol{form}, (\mathrm{Ob}\ \mathbf{ARCH}, r\text{-}ARCH)^{\mathrm{op}} \rangle.$$

Предложение 4.27. ***SARCH*** является кооднородной технологией проектирования, в которой все трансформации трассируемы.

Доказательство. Очевидно, что функтор ***form*** унивалентен. Левый сопряженный к нему имеет вид $\boldsymbol{form}^* : \langle c\text{-}DESC, Conf, r\text{-}DESC \rangle \mapsto \langle c\text{-}DESC, Conf, 1_{c\text{-}DESC}, r\text{-}DESC \rangle$, единица этого сопряжения тождественна (ср. конструирование функтора *mod** в доказательстве теоремы 4.1). Далее применяем предложения 2.7 и 2.19. Правый обратный к ***form***-образу произвольного *r-ARCH*-морфизма $\langle cm, sm, rm \rangle : AR \to AR'$ имеет вид $\langle cm^*, drm \rangle : \boldsymbol{form}(AR') \to \boldsymbol{form}(AR)$. □

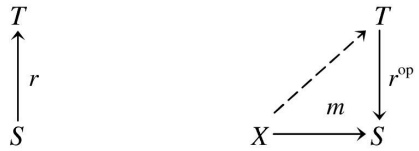
Отметим, что ввиду следствия 4.4.1 тройка $\langle \boldsymbol{spec}, \boldsymbol{fconf}, \boldsymbol{spec}(-^{\mathrm{op}})^{\mathrm{op}} \rangle$ представляет собой **ARCH**-морфизм из технологии проектирования технологий проектирования ***SARCH*** в технологию проектирования технологий специфицирования.

Глава 5. ТЕОРИЯ И ПРИЛОЖЕНИЯ ФОРМАЛЬНОГО МОДЕЛИРОВАНИЯ

5.1 Синтез технологий проектирования

В гл.4 предполагалось, что исходная «модульная» технология проектирования систем задана, и требовалось расширить ее приемами аспектно-ориентированного синтеза. Однако, как мы видели в разд.1.4, для разработки, управляемой моделями (MDE), характерна другая постановка задачи: выбор технологии допускает определенный произвол, и требуется произвести его так, чтобы получить наибольший эффект (снижение затрат) от ее применения в жизненном цикле больших систем. Согласно результатам гл.2, проще всего варьировать структуру трансформаций, обеспечивающих движение вдоль процесса разработки системы. Действительно, технология специфицирования, как правило, достаточно жестко определяется выбранным формальным методом моделирования: категория интерфейсов «естественным» образом выбирается среди полных корефлективных подкатегорий в категории моделей (строго говоря, среди их изоморфных образов – об этом говорит условие (iii) определения 2.3), а конфигурации отбираются из потенциала комплексируемости корефлектора согласно предложению 2.7. Если требуются сложные технологии специфицирования, в том числе поддерживающие модели нескольких видов одновременно, то они синтезируются из более простых при помощи формализованных модульных и аспектно-ориентированных приемов в рамках технологии проектирования технологий специфицирования (следствие 4.4.1).

Как было показано в предыдущих главах, главным критерием рационального выбора трансформаций является обеспечение как можно более широких возможностей для трассирования. Трансформации должны быть трассируемыми (двойственными по отношению к действиям по интеграции), нетривиальными (выходить за рамки изоморфизмов), и в то же время реализация трассирования не должна быть сопряжена с чрезмерными затратами. Приемлемым компромиссом выглядит требование, чтобы трансформации позволяли трассировать включения компонентов в системы – действия по интеграции, не изменяющие внутреннюю структуру компонентов и благодаря этому трассируемые «бесплатно» [77]. Воспользуемся соображениями, изложенными в разд.4.2. Зафиксируем произвольную формальную технологию специфицирования $SC = \langle c\text{-}DESC, Conf, sig : c\text{-}DESC \to SIG \rangle$. Если компонент X включается в систему S (имеется включение $m : X \to S$), а S трансформируется в систему T (имеется трансформация $r : S \to T$), то X должен включаться в T, причем так, чтобы композиция (в категории $c\text{-}DESC$) трассы трансформации $r^{op} : T \to S$ с этим включением давала m.



Это условие легко выполнить на уровне интерфейсов (причем для произвольного морфизма *m*), поскольку *sig*-образ любой трассы обратим справа: искомое включение *sig*(*X*) в *sig*(*T*) имеет вид $s \circ sig(m)$, где *s* – любой морфизм, удовлетворяющий условию $sig(r^{op}) \circ s = 1_{sig(S)}$. Поэтому трансформация *r*, обеспечивающая трассирование включений, должна позволять «поднять» любой *sig*-морфизм из *sig*(*X*) в *sig*(*T*), композиция $sig(r^{op})$ с которым равна *sig*-образу заданного включения *m*, на уровень моделей – до морфизма, действующего из *X* в *T*. Как мы увидим далее (определение 5.1), в теории категорий это условие является обобщением определения понятия инициального морфизма.

В связи с этим целесообразно конструировать хорошо трассируемые трансформации начиная с уровня интерфейсов – т.е. прежде всего зафиксировать класс L всех допустимых разметок формальных моделей. Согласно утверждению (ii) предложения 4.1, он должен состоять из ретракций и содержать все изоморфизмы. Трассы всех трансформаций находятся в классе $sig^{-1}(L)$, поэтому его можно считать состоящим из действий по интеграции, наиболее сильно изменяющих структуру компонентов. Следовательно, включения, не разрушающие ее, можно определить как действия по интеграции, «ортогональные» им. В теории категорий имеется формальное понятие ортогональности, которое дает следующую характеризацию класса всех включений: любое действие по интеграции должно однозначно (с точностью до изоморфизма) разлагаться в композицию включения с морфизмом, переходящим в разметку под действием функтора *sig*. Классы морфизмов, порождающие разложения такого рода, называются компонентами факторизационной системы (factorization system [159], factorization structure [136]). Мы воспроизведем точное определение ниже, а здесь приведем классический пример: факторизационной системой является пара (Epi, Mono) в категории **Set**, где мономорфизмы задают включения подмножеств, а любой эпиморфизм размечает свою область элементами кообласти. Ортогональность включений трассам означает, что на уровне интерфейсов любое действие по интеграции компонента в систему сводится к включению в нее результата некоторой разметки компонента. Иначе говоря, воздействие системного окружения на интерфейс компонента заключается в его «огрублении» – отмене некоторой трансформации, определяемой однозначно с точностью до изоморфизма. (Такая организация интеграции отражена в структурном шаблоне объектно-ориентированного проектирования Адаптер [27], изображенном в разд.2.1: в нем трассирование интерфейса реализуется путем множественного наследования.)

Чтобы неразрушающий характер включения распространялся на дискретную структуру, дополнительно накладывается условие сохранения дискретности, согласно которому в *sig*-дискретную систему должны включаться только дискретные компоненты. В подтверждение правомерности этого условия можно привести тот факт, что в любой формальной технологии любой *c-DESC*-морфизм с дискретной кообластью, переходящий в сечение под действием функтора *sig*, имеет дискретную область. Введем обозначения: ε – коединица сопряжения $sig^* \dashv sig$, $f^* = sig^*(sig(f))$ для произвольного *c-DESC*-морфизма *f*. Напомним, что *c-DESC*-объект *S* называется дискретным, если ε_S является изоморфизмом. Для любых $n : A \to B$ и $u : sig(B) \to sig(A)$, удовлетворяющих условиям $\varepsilon_B \in$ Iso *c-DESC* и $u \circ sig(n) = 1_{sig(A)}$, из тождества коединицы $n \circ \varepsilon_A = \varepsilon_B \circ n^*$ вытекает, что $(sig^*(u) \circ \varepsilon_B^{-1} \circ n) \circ \varepsilon_A = 1_{A^*}$, т.е. ε_A – сечение. А поскольку ε состоит из эпиморфизмов, получаем, что $\varepsilon_A \in$ Iso *c-DESC*. Например, в технологиях над **Set** обычно все мономорфизмы сохраняют дискретность в указанном смысле.

Кроме того, в целях обеспечения трассируемости на уровне конфигураций, накладывается условие кооднородности (см. определение 2.12). Трансформации отбираются из класса $sig^{-1}(\mathrm{L})^{\mathrm{op}}$ по критерию способности трассировать включения компонентов. В результате получается формальная технология проектирования, которую мы будем называть L-трансформационной. Следуя работе [77], мы покажем, что трансформационные технологии очень хорошо поддаются аспектной ориентации. Все трансформации в них обратимы в смысле определения 4.1, а мономорфные включения являются неразрушающими в смысле определения 2.9. Процедура построения трансформационных технологий трассируется вдоль аспектных трансформаций технологий специфицирования (поэтому является естественной относительно подходящего класса **SPEC**-морфизмов), в частности опускается на уровень интерфейсов, а также поднимается на уровень аспектно-ориентированных технологий. Поэтому категория *tr-AO* может быть построена как категория запятой определенного вида, и (при некоторых дополнительных ограничениях) обеспечиваются такие свойства, как аспектная полнота, структурируемость и скоординированность АО-технологий. В качестве примеров технологий проектирования, обеспечивающих трассирование включений и обладающих свойствами такого рода, мы рассмотрим технологии моделирования ключевых составляющих информационно-управляющих систем – данных и сценариев исполнения процессов.

Определение 5.1. Пусть $ff : C \to D$ – произвольный унивалентный функтор, M – произвольный класс *C*-морфизмов. *C*-морфизм $f : T \to S$ называется M-*инициальным*, если для любых M-морфизма $m : X \to S$ и *D*-морфизма $k : ff(X) \to ff(T)$ таких, что $ff(f) \circ k = ff(m)$, существует *C*-морфизм $k^+ : X \to T$ такой, что $ff(k^+) = k$. □

Будем обозначать через M-Init класс всех M-инициальных морфизмов. Это понятие построено как обобщение известного в теории категорий понятия инициального морфизма, который является в точности Mor-инициальным морфизмом, т.е. M-инициальным для любого класса M ([136, определение 8.6], см. также разд.2.4). Его можно обобщить на источники (ср. пояснения перед определением 2.7): для произвольного класса **M** источников в категории *C*, назовем **M**-*инициальным* поднятие $(A, \overline{f_i} : A \to A_i)_{i \in I}$ *ff*-структурированного источника $(B, f_i : B \to ff(A_i))_{i \in I}$ такое, что для любых *D*-объекта *B*', *D*-морфизма $h : B' \to B$ и поднятия $(A', \overline{f_i}' : A' \to A_i)_{i \in I}$ *ff*-структурированного источника $(B', f_i \circ h : B' \to ff(A_i))_{i \in I}$, принадлежащего классу **M**, существует единственный *C*-морфизм $\overline{h} : A' \to A$ такой, что $ff(\overline{h}) = h$ и $\overline{f_i}' = \overline{f_i} \circ \overline{h}$, $i \in I$.

Существуют категории, имеющие факторизационную систему вида (sig^{-1}(Iso), Mor-Init), например все топологические категории (см. [136, упражнение 21M(3)]). Этот факт можно обобщить для M-инициальных морфизмов формальных моделей систем. Пусть M – класс морфизмов некоторой категории. Будем говорить, что выполняется отношение M-Difip(*e*, *f*) между морфизмами *e* и *f*, если для всякого коммутативного квадрата $f \circ u = m \circ e$ такого, что $m \in$ M, существует единственный «диагональный» морфизм *d* такой, что $d \circ e = u$ и $f \circ d = m$. Заметим, что если *e* – эпиморфизм, то для установления M-Difip(*e*, *f*) достаточно найти *d* такой, что $d \circ e = u$. Напомним, что отношение Difip (diagonal fill-in property), совпадающее с нашим Mor-Difip, применяется в теории категорий для определения факторизационной системы. А именно, факторизационной системой называется пара классов морфизмов (E, F) такая, что:

(i) Iso ∘ E ⊆ E. (ii) F ∘ Iso ⊆ F. (iii) F ∘ E = Mor. (iv) Difip(E, F).

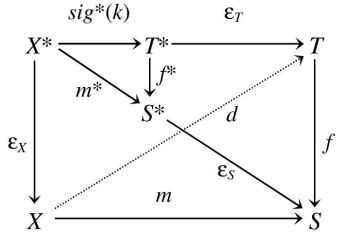
Перечислим несколько свойств произвольной факторизационной системы, установленных в [136, гл.14]. (E, F)-разложение любого морфизма (т.е. его представление в виде композиции $f \circ e$, где $e \in$ E и $f \in$ F) определяется однозначно с точностью до изоморфизма. Класс E замкнут относительно композиции и содержит все изоморфизмы. Он слабо сократим справа в том смысле, что условия $e \in$ E и $e' \circ e \in$ E влекут $e' \in$ E. Также он замкнут относительно формирования копроизведений и кодекартовых квадратов (если они существуют) в том смысле, что копроизведение E-морфизмов и ребро кодекартова квадрата, параллельное E-морфизму, содержатся в E. Двойственные утверждения справедливы для F.

Предложение 5.1. Пусть M ⊆ Mor *c-DESC*. *c-DESC*-морфизм *f* является M-инициальным тогда и только тогда, когда для всякого *c-DESC*-морфизма *e* условие $sig(e) \in$ Iso *SIG* влечет M-Difip(*e*, *f*).

Доказательство. Пусть $f \in$ M-Init, $sig(e) \in$ Iso *SIG* (так что $e \in$ Epi *SIG*), $f \circ u = m \circ e$ для некоторого $m \in$ M. Имеем $sig(f) \circ (sig(u) \circ sig(e)^{-1}) = sig(m)$, поэтому существует *c-DESC*-

морфизм $d : \mathrm{codom}\, e \to \mathrm{dom}\, f$ такой, что $sig(d) = sig(u) \circ sig(e)^{-1}$. Поскольку sig унивалентен, $d \circ e = u$, так что d – искомая диагональ.

Предположим теперь, что имеет место M-Difip(sig^{-1}(Iso SIG), f) для некоторого $f : T \to S$. Выберем $m : X \to S \in$ M и $k : sig(X) \to sig(T)$ так, что $sig(f) \circ k = sig(m)$. Имеем $f^* \circ sig^*(k) = m^*$, поэтому $f \circ (\varepsilon_T \circ sig^*(k)) = \varepsilon_S \circ f^* \circ sig^*(k) = \varepsilon_S \circ m^* = m \circ \varepsilon_X$. Поскольку $sig(\varepsilon_X) = 1_{sig(X)} \in$ Iso SIG, существует диагональ $d : X \to T$ такая, что $d \circ \varepsilon_X = \varepsilon_T \circ sig^*(k)$, откуда $sig(d) = k$, так что f M-инициален. □



Предложение 5.2. Пусть M ⊆ Mor *c-DESC*. Любой *c-DESC*-морфизм с дискретной кообластью является M-инициальным тогда и только тогда, когда любой M-морфизм с дискретной кообластью имеет дискретную область.

Доказательство. Допустим, что любой *c-DESC*-морфизм с дискретной кообластью является M-инициальным. Если $m : X \to S \in$ M и $\varepsilon_S \in$ Iso *c-DESC*, то $\varepsilon_S \circ m^* : X^* \to S \in$ M-Init, так что в силу предложения 5.1 коммутативный квадрат $(\varepsilon_S \circ m^*) \circ 1_{X^*} = m \circ \varepsilon_X$ имеет диагональ d такую, что $d \circ \varepsilon_X = 1_{X^*}$. Поэтому ε_X, будучи эпиморфизмом, обратимым слева, является изоморфизмом.

Обратно, допустим, что любой M-морфизм с дискретной кообластью имеет дискретную область. Выберем $f : T \to S$, $m : X \to S \in$ M, $k : sig(X) \to sig(T)$ так, что $\varepsilon_S \in$ Iso *c-DESC* (а значит, и $\varepsilon_X \in$ Iso *c-DESC*), $sig(f) \circ k = sig(m)$. Имеем $sig(\varepsilon_T \circ sig^*(k) \circ \varepsilon_X^{-1}) = k$. □

Предложение 5.3. Пусть (E, M) – факторизационная система в категории *c-DESC* такая, что sig^{-1}(Iso SIG) ⊆ E ⊆ Epi *c-DESC*. Тогда все *c-DESC*-объекты и все M-инициальные морфизмы образуют подкатегорию в *c-DESC*, обладающую факторизационной системой (E ⋂ M-Init, M).

Доказательство. Сначала установим, что для произвольных *c-DESC*-морфизмов n, p условие $n \circ p \in$ M влечет $p \in$ M. Действительно, если $p = p' \circ e$ и $(n \circ p') = p'' \circ e'$ – (E, M)-разложения, то $n \circ p = p'' \circ (e' \circ e)$ – (E, M)-разложение M-морфизма, поэтому $e' \circ e \in$ Iso *c-DESC*, так что e обратим слева, а любой обратимый слева эпиморфизм является изоморфизмом, следовательно $p \in$ M ∘ Iso *c-DESC* = M. Следовательно, в обозначениях определения 5.1 морфизм k^+

принадлежит классу M (имеем $f \circ k^+ = m$ ввиду унивалентности). С учетом этого непосредственно проверяется, что класс M-Init замкнут относительно композиции.

Далее, любой M-морфизм является инициальным (следовательно, M-инициальным) в силу предложения 5.1, поскольку имеет место Difip(sig^{-1}(Iso *SIG*), M). В частности, Iso *c-DESC* $\subseteq$ M $\subseteq$ M-Init. Теперь достаточно показать, что условия $n \in$ M и $n \circ f \in$ M-Init влекут $f \in$ M-Init: тогда (E, M)-разложение любого M-инициального морфизма будет его искомым (E $\bigcap$ M-Init, M)-разложением (см. [136, предложение 14.7]). Пусть $f : T \to S$, выберем произвольно $m : X \to S \in$ M и $k : sig(X) \to sig(T)$ такие, что $sig(f) \circ k = sig(m)$. Тогда $sig(n \circ f) \circ k = sig(n \circ m)$, а поскольку $n \circ f \in$ M-Init и $n \circ m \in$ M, существует $k^+ : X \to T$ такой, что $sig(k^+) = k$. □

Определение 5.2. Формальная технология $SC = \langle c\text{-}DESC, Conf, sig \rangle$ *поддерживает* L-*разметки*, где L – некоторый класс *SIG*-морфизмов, если выполняются следующие условия:

(i) любой L-морфизм является ретракцией;

(ii) существует класс M $\subseteq$ Mor *c-DESC* такой, что пара (sig^{-1}(L), M) является факторизационной системой в *c-DESC*, и любой M-морфизм с дискретной кообластью имеет дискретную область;

(iii) класс *Conf* замкнут относительно накачек sig^{-1}(L)-морфизмами.

При этом четверка $SC_L = \langle c\text{-}DESC, Conf, sig, (\mathrm{Tr\ L})^{op} \rangle$, где Tr L = (Ob *c-DESC*, sig^{-1}(L) $\bigcap$ M-Init), называется L-*трансформационной технологией проектирования* над *SC*, M-морфизмы называются SC_L-*включениями*. □

Предложение 5.4. Любая L-трансформационная технология SC_L является кооднородной формальной технологией проектирования, поддерживающей трассирование, с классом всех разметок L, причем

RegMono *c-DESC* $\subseteq$ Mono *c-DESC* $\bigcap$ Mor-Init =

Mono *c-DESC* $\bigcap$ M-Init $\subseteq$ M $\subseteq$ Mor-Init $\subseteq$ M-Init.

Доказательство. Условие (ii) определения 5.2 гарантирует, что Iso *c-DESC* $\subseteq sig^{-1}$(L), откуда Iso *SIG* = *sig*(*sig**(Iso *SIG*)) $\subseteq$ *sig*(Iso *c-DESC*) $\subseteq$ L, а также что Iso *c-DESC* $\subseteq$ M. Поэтому ввиду предложения 5.3 Tr L действительно является подкатегорией в *c-DESC*, содержащей все изоморфизмы. Отсюда с учетом условия (iii) определения 5.2 и предложения 2.19 вытекает, что SC_L является кооднородной формальной технологией проектирования. Далее, согласно условию (i) определения 5.2 для любого (Tr L)-морфизма $t : T \to S$ существует *SIG*-морфизм *s* такой, что $sig(t) \circ s = 1_{sig(S)}$. Поскольку $1_S \in$ M, в силу определения 5.1 существует *c-DESC*-морфизм t^+ такой, что $t \circ t^+ = 1_S$, так что SC_L поддерживает трассирование. Из заключительной части условия (ii) определения 5.2 и предложения 5.2 вытекает, что *sig**(L) $\subseteq sig^{-1}$(L) $\bigcap$ M-Init = Mor Tr L, поэтому *sig*(Mor Tr L) $\supseteq$ *sig*(*sig**(L)) = L.

Соотношение $\mathrm{M} \subseteq$ Mor-Init установлено в доказательстве предложения 5.3. Пусть $m : X \to S$ – M-инициальный *c-DESC*-мономорфизм, $m = m' \circ e$ – его (sig^{-1}(L), M)-разложение. Тогда e – мономорфизм, а поскольку функтор *sig* ввиду наличия у него левого сопряженного сохраняет мономорфизмы, $sig(e) \in$ Iso *SIG*. В силу предложения 5.1 коммутативный квадрат $m \circ 1_X = m' \circ e$ имеет диагональ d такую, что $d \circ e = 1_X$, следовательно, $e \in$ Iso *c-DESC*, так что $m \in$ M. Таким образом, Mono *c-DESC* $\bigcap$ M-Init $\subseteq$ M. Наконец, как указывалось в разд.2.4, любой регулярный *c-DESC*-мономорфизм является инициальным. □

Предложение 5.5. В произвольной L-трансформационной технологии SC_{L} любой sig^{-1}(L)-морфизм является трассой трансформации тогда и только тогда, когда функтор *sig* является эквивалентностью категорий *c-DESC* и *SIG*.

Доказательство. Имеем следующую цепочку равносильных утверждений:

sig – эквивалентность ⇔ (по определению эквивалентности)

$\varepsilon \subseteq$ Iso *c-DESC* ⇔ (для любого $f : A \to B$ имеем $f \circ \varepsilon_A = \varepsilon_B \circ f^*$)

sig^{-1}(Iso *SIG*) = Iso *c-DESC* ⇔ (учитываем, что $\mathrm{L} \supseteq$ Iso *SIG* и $\mathrm{M} \supseteq$ Iso *c-DESC*)

M-Difip(sig^{-1}(Iso *SIG*), sig^{-1}(L)) ⇔ (в силу предложения 5.1)

$sig^{-1}(\mathrm{L}) \subseteq$ M-Init ⇔ (по определению 5.2)

Mor Tr L = sig^{-1}(L). □

Предложение 5.6. В произвольной L-трансформационной технологии SC_{L} любая трансформация является *c-DESC*-изоморфизмом тогда и только тогда, когда L = Iso *SIG*.

Доказательство. Необходимость следует из предложений 5.4 и 4.1. Для доказательства достаточности заметим, что согласно условию (ii) определения 5.2 и предложению 5.1 условие L = Iso *SIG* влечет M = Mor-Init. Но любой инициальный *c-DESC*-морфизм, переходящий в изоморфизм под действием функтора *sig*, сам является изоморфизмом [136, предложение 8.14]. □

Предложение 5.7. Структурируемая технология *SC* поддерживает (Iso *SIG*)-разметки тогда и только тогда, когда функтор *sig* является эквивалентностью категорий и класс *Conf* замкнут относительно накачек изоморфизмами.

Доказательство. Если *sig* – эквивалентность, то sig^{-1}(Iso *SIG*) = Iso *c-DESC* (в частности, все *c-DESC*-объекты дискретны, так что факторизационная система (Iso *c-DESC*, Mor *c-DESC*) удовлетворяет условию (ii) определения 5.2 при L = Iso *SIG*). Обратно, предположим, что *SC* поддерживает (Iso *SIG*)-разметки. Пусть M – соответствующий класс включений, σ – единица сопряжения $sig^{**} \dashv sig^*$, A – произвольный *c-DESC*-объект, $\sigma_A = m \circ e$ – (sig^{-1}(Iso *SIG*), M)-разложение. Поскольку *c-DESC*-объект codom m = codom $\sigma_A = sig^*(sig^{**}(A))$ дискретен, *c-DESC*-объект dom m = codom e также дискретен ввиду условия (ii) определения 5.2. Тогда тождество коединицы для e преобразуется в соотношение $(sig^*(sig(e)^{-1}) \circ \varepsilon_{\mathrm{codom}\, e}^{-1} \circ e) \circ \varepsilon_A = 1_{A^*}$,

показывающее, что ε_A – сечение, а поскольку ε состоит из эпиморфизмов, получаем, что $\varepsilon_A \in$ Iso *c-DESC* (ср. пояснения к условию сохранения дискретности в начальной части настоящего раздела). Таким образом, ε состоит из изоморфизмов, так что *sig* – эквивалентность. □

Теорема 5.1. Пусть $SC' = \langle c\text{-}DESC', Conf', sig' : c\text{-}DESC' \to SIG' \rangle$ – произвольная формальная технология специфицирования, $\langle cm, sm \rangle : SC \to SC'$ – аспектная ***conf***-трасса. Если *SC* поддерживает L-разметки, то *SC*' поддерживает *sm*(L)-разметки, причем класс всех $SC'_{sm(\mathrm{L})}$-включений имеет вид *cm*(M) ∘ Iso *c-DESC*', где M – класс всех SC_{L}-включений.

Доказательство. Введем обозначения: Ł = $sig'^{-1}(sm(\mathrm{L}))$, Ṁ = *cm*(M) ∘ Iso *c-DESC*', ε' – коединица сопряжения *sig*'* ⊣ *sig*', ζ – коединица сопряжения *cm** ⊣ *cm*. Мы будем пользоваться следующими свойствами **SPEC**-морфизма $\langle cm, sm \rangle$:

(a) $sm \circ sig = sig' \circ cm$ (ввиду следствия 4.4.1);

(b) $cm(sig^{-1}(\mathrm{L})) \subseteq$ Ł (из (a) $sig'(cm(q)) = sm(sig(q)) \in sm(\mathrm{L})$ для любого $q \in sig^{-1}(\mathrm{L})$);

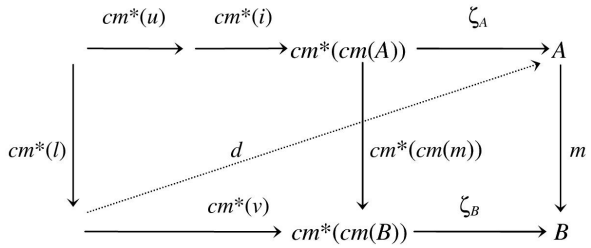
(c) $sig = sm^{-1} \circ sig' \circ cm$ (из (a) с учетом того, что ввиду предложения 4.4 функтор *sm* является изоморфизмом категорий);

(d) $cm^*(Ł) \subseteq sig^{-1}(\mathrm{L})$ (из (c) $sig(cm^*(h)) = sm^{-1}(sig'(cm(cm^*(h)))) = sm^{-1}(sig'(h)) \in$ L для любого $h \in$ Ł с учетом соотношения $cm \circ cm^* = 1_{c\text{-}DESC'}$).

Проверим выполнение всех условий определения 5.2 для технологии *SC*' и класса *SIG*'-морфизмов *sm*(L).

(i). Любой функтор сохраняет ретракции, поэтому *sm*(L) состоит из ретракций.

(ii). Положим Ṁ' = $\{cm(m) \mid m \in \mathrm{M}, \zeta_{\mathrm{codom}\, m} = 1_{\mathrm{codom}\, m}\}$ ∘ Iso *c-DESC*' и установим наличие в *c-DESC*' факторизационной системы (Ł, Ṁ'). Сначала проверим, что $j \circ g \in$ Ł для любых $j \in$ Iso *c-DESC*' и $g \in$ Ł. Ввиду утверждения (d) и условия (ii) определения 5.2 $cm^*(j \circ g) \in$ Iso *c-DESC* ∘ $cm^*(Ł) \subseteq$ Iso *c-DESC* ∘ $sig^{-1}(\mathrm{L}) \subseteq sig^{-1}(\mathrm{L})$, откуда с учетом утверждения (b) $j \circ g = cm(cm^*(j \circ g)) \in cm(sig^{-1}(\mathrm{L})) \subseteq$ Ł. Далее, ввиду условия (ii) определения 5.2 для любого *c-DESC*'-морфизма *f* существует (sig^{-1}(L), M)-разложение $cm^*(f) = n \circ e$, откуда получается равенство $f = cm(n) \circ cm(e)$, которое ввиду утверждения (b) представляет собой (Ł, Ṁ')-разложение. Проверим Difip(Ł, Ṁ'). Если $(cm(m) \circ i) \circ u = v \circ l$ для некоторых $m : A \to B \in$ M, $i \in$ Iso *c-DESC*' и $l \in$ Ł, то, полагая $w = cm^*(i \circ u)$, имеем $m \circ (\zeta_A \circ w) = \zeta_B \circ cm^*(cm(m)) \circ w = (\zeta_B \circ cm^*(v)) \circ cm^*(l)$, поэтому ввиду Difip($sig^{-1}$(L), M) (условие (ii) определения 5.2) и утверждения (d) существует *c-DESC*-морфизм *d* такой, что $d \circ cm^*(l) = \zeta_A \circ w$. Действуя на это равенство функтором *cm*, получаем $cm(d) \circ l = i \circ u$, так что $i^{-1} \circ cm(d)$ – искомая диагональ.



Проверим, что для любого $\dot{M}'$-морфизма $p : X \to Y$ такого, что $\varepsilon'_Y \in \mathrm{Iso}\ c\text{-}DESC'$, имеет место $\varepsilon'_X \in \mathrm{Iso}\ c\text{-}DESC'$. Пусть $p = cm(r) \circ t$ для некоторых $r : P \to Q \in \mathrm{M}$ и $t : X \to cm(P) \in \mathrm{Iso}\ c\text{-}DESC'$ таких, что $\zeta_Q = 1_Q$ и $cm(Q) = Y$. Из утверждения (c) и правил вычисления композиции сопряжений функторов [93, разд.4.8] вытекает, что $\varepsilon_Z = \zeta_Z \circ cm^*(\varepsilon'_{cm(Z)})$ для любого $c\text{-}DESC$-объекта Z, в частности по предположению $\varepsilon_Q = cm^*(\varepsilon'_Y) \in \mathrm{Iso}\ c\text{-}DESC$. Отсюда ввиду условия (ii) определения 5.2 $\varepsilon_P \in \mathrm{Iso}\ c\text{-}DESC$, следовательно, подставляя P вместо Z, убеждаемся, что $c\text{-}DESC$-морфизм $cm^*(\varepsilon'_{cm(P)})$ обратим слева. А поскольку он является эпиморфизмом (коединица ε' состоит из эпиморфизмов, а поскольку функтор cm^* обладает правым сопряженным, он сохраняет все копределы, в частности все эпиморфизмы), он – изоморфизм. Отсюда $\varepsilon'_{cm(P)} = cm(cm^*(\varepsilon'_{cm(P)})) \in \mathrm{Iso}\ c\text{-}DESC'$, следовательно $\varepsilon'_X = t^{-1} \circ \varepsilon'_{cm(P)} \circ cm^*(cm(t)) \in \mathrm{Iso}\ c\text{-}DESC'$.

Осталось убедиться, что $\dot{M} = \dot{M}'$. С одной стороны, по построению $\dot{M}' \subseteq \dot{M}$. С другой стороны, имеет место $\mathrm{Difip}(Ł, \dot{M})$ (это соотношение проверяется дословно так же, как $\mathrm{Difip}(Ł, \dot{M}')$ выше), поэтому $\dot{M} \subseteq \dot{M}'$.

(iii). Пусть $\tau \rightrightarrows \Sigma$ – накачка произвольной диаграммы $\Sigma \in Conf'$ произвольным семейством Ł-морфизмов τ. Рассмотрим диаграмму $\Delta = cm^* \circ (\tau \rightrightarrows \Sigma) = cm^*(\tau) \rightrightarrows cm^* \circ \Sigma$. Ввиду утверждения (d) семейство $cm^*(\tau)$ состоит из sig^{-1}(L)-морфизмов, а с учетом определения ***conf***-трассы соотношение $cm \circ cm^* \circ \Sigma = \Sigma$ влечет $cm^* \circ \Sigma \in Conf$. Отсюда $\Delta \in Conf$ ввиду условия (iii) определения 5.2, так что $\tau \rightrightarrows \Sigma = cm \circ \Delta \in Conf'$. □

Следствие 5.1.1. Если SC поддерживает L-разметки, то тройка $SSIG = \langle SIG, sig \circ Conf, 1_{SIG} \rangle$ является формальной технологией специфицирования, поддерживающей L-разметки, для которой класс всех L-трансформаций совпадает с L^{op}, а класс всех включений – с sig(M), где M – класс всех SC_{L}-включений. Функтор sig индуцирует трансформацию L-трансформационной технологии $SSIG_{\mathrm{L}}$ в SC_{L}.

Доказательство. Ввиду следствия 4.4.1 и предложения 4.15 имеем $SSIG = \boldsymbol{conf}^*(\boldsymbol{conf}^{**}(SC)) \in \mathrm{Ob}\ \mathbf{SPEC}$. Из утверждения (ii) теоремы 4.4 и теоремы 4.2 вытекает, что компонента единицы сопряжения $\boldsymbol{conf}^{**} \dashv \boldsymbol{conf}^*$ переводит SC в изоаспектную трассу $\langle sig, 1_{SIG} \rangle : SC \to SSIG$. Теперь применяем теорему 5.1. В частности, как установлено в ее дока-

зательстве, любое $SSIG_{\mathrm{L}}$-включение имеет вид $sig(m) \circ i$ для некоторых $m : P \to Q \in \mathrm{M}$ и $i \in \mathrm{Iso}\ SIG$, причем $\varepsilon_Q = 1_Q$. Но тогда ввиду условия (ii) определения 5.2 $\varepsilon_P \in \mathrm{Iso}\ c\text{-}DESC$, а поскольку класс M замкнут относительно композиции и содержит все *c-DESC*-изоморфизмы, имеем $sig(m) \circ i = sig(m \circ \varepsilon_P \circ sig^*(i)) \in sig(\mathrm{M})$

Наконец, поскольку $SSIG_{\mathrm{L}} = \langle SIG, sig \circ Conf, 1_{SIG}, (\mathrm{Ob}\ SIG, \mathrm{L})^{\mathrm{op}} \rangle$, имеется **ARCH**-морфизм $\langle sig, 1_{SIG}, sig(-^{\mathrm{op}})^{\mathrm{op}} \rangle : SC_{\mathrm{L}} \to SSIG_{\mathrm{L}}$, который ввиду предложения 5.2 является трассой трансформации технологий проектирования. □

Следствие 5.1.2. Если *SC* поддерживает L-разметки, то все *SIG*-объекты и все L-морфизмы образуют подкатегорию в *SIG*, содержащую все *SIG*-изоморфизмы.

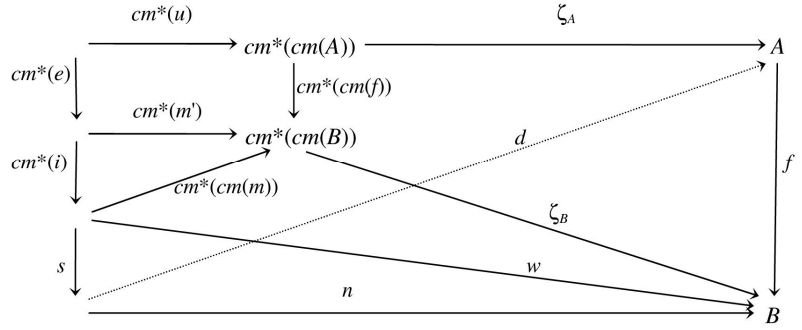
Доказательство непосредственно вытекает из следствия 5.1.1. □

Следствие 5.1.3. Пусть $SC' = \langle c\text{-}DESC', Conf', sig' : c\text{-}DESC' \to SIG' \rangle$ – произвольная формальная технология специфицирования, $\langle cm, sm \rangle : SC \to SC'$ – **SPEC**-морфизм такой, что *cm* – ***desc***-разметка, *sm* – изоморфизм категорий. Если *SC* поддерживает L-разметки и *SC'* поддерживает L'-разметки, то соотношение $sm(\mathrm{L}) \subseteq \mathrm{L}'$ выполняется тогда и только тогда, когда тройка $\langle cm, sm, cm(-^{\mathrm{op}})^{\mathrm{op}} \rangle$ представляет собой **ARCH**-морфизм из SC_{L} в $SC'_{\mathrm{L}'}$.

Доказательство. Обозначим через M' класс всех $SC'_{\mathrm{L}'}$-включений. Мы будем пользоваться коединицей ζ, факторизационной системой $(Ł, \dot{M})$ в *c-DESC'* и утверждениями (a)-(d), которые задаются точно так же, как в доказательстве теоремы 5.1: для этого достаточно свойств **SPEC**-морфизма $\langle cm, sm \rangle$, указанных в условии доказываемого утверждения.

Сначала предположим, что $sm(\mathrm{L}) \subseteq \mathrm{L}'$, так что $Ł \subseteq sig'^{-1}(\mathrm{L}')$, и проверим, что $cm(f) \in \mathrm{Mor\ Tr\ L}'$ для произвольного (Tr L)-морфизма $f : A \to B$. Ввиду утверждения (b) по предположению имеем $cm(f) \in Ł \subseteq sig'^{-1}(\mathrm{L}')$. Далее воспользуемся предложением 5.1: установим M'-Difip(e, $cm(f)$) для произвольного $e \in sig'^{-1}(\mathrm{Iso}\ SIG')$. Пусть $cm(f) \circ u = m' \circ e$ – произвольный коммутативный квадрат такой, что $m' \in \mathrm{M}'$. По предположению, имеет место Difip(Ł, m'), поэтому $m' \in \dot{M}$, т.е. $m' = cm(m) \circ i$ для некоторых $m \in \mathrm{M}$ и $i \in \mathrm{Iso}\ c\text{-}DESC'$. Действуя функтором cm^* на наш квадрат, получаем соотношение $cm^*(cm(f)) \circ cm^*(u) = cm^*(cm(m)) \circ cm^*(i) \circ cm^*(e)$. Умножая это равенство слева на ζ_B и применяя тождество коединицы $f \circ \zeta_A = \zeta_B \circ cm^*(cm(f))$, получаем соотношение $f \circ \zeta_A \circ cm^*(u) = w \circ cm^*(i) \circ cm^*(e)$, где $w = \zeta_B \circ cm^*(cm(m))$. Пусть $w = n \circ s$ – $(sig^{-1}(\mathrm{L}), \mathrm{M})$-разложение. Ввиду следствия 5.1.1 соотношение $sig(w) = sig(n) \circ sig(s)$ представляет собой (L, sig(M))-разложение, а поскольку с учетом утверждения (c) имеем $sig(w) = sm^{-1}(sig'(cm(\zeta_B \circ m))) = sig(m) \in sig(\mathrm{M})$, получаем $sig(s) \in \mathrm{Iso}\ SIG$. Окончательно, имеем в *c-DESC* коммутативный квадрат $f \circ h = n \circ g$, где $h = \zeta_A \circ cm^*(u)$ и $g = s \circ cm^*(i) \circ cm^*(e)$, так что ввиду утверждения (c) $sig(g) = sig(s) \circ sig(cm^*(i)) \circ sm^{-1}(sig'(e)) \in \mathrm{Iso}\ SIG$. А поскольку $f \in \mathrm{M\text{-}Init}$ и $n \in \mathrm{M}$, ввиду предложения 5.1 существует диагональ d такая, что $d \circ g = h$. Действуя на это

соотношение функтором *cm*, получаем $d' \circ e = u$, где $d' = cm(d) \circ cm(s) \circ i$, так что d' – искомая диагональ исходного квадрата. Окончательно, имеем $cm(f) \in sig'^{-1}(\mathrm{L}') \bigcap \mathrm{M}'\text{-Init} = \mathrm{Mor\ Tr\ L}'$.



Теперь предположим, что $cm(\mathrm{Mor\ Tr\ L}) \subseteq \mathrm{Mor\ Tr\ L}'$, и проверим, что $sm(l) \in \mathrm{L}'$ для произвольного L-морфизма *l*. Из заключительной части условия (ii) определения 5.2 и предложения 5.2 вытекает, что $sig^*(l) \in \mathrm{Mor\ Tr\ L}$, поэтому с учетом утверждения (a) $sm(l) = sm(sig(sig^*(l))) = sig'(cm(sig^*(l))) \in \mathrm{L}'$. □

Следствие 5.1.4. Технология специфицирования $\boldsymbol{spec}(AO_{int}(SC_{\mathrm{L}}))$ поддерживает L-разметки, причем L-трансформационная технология $\boldsymbol{spec}(AO_{int}(SC_{\mathrm{L}}))_{\mathrm{L}}$ является подтехнологией в $AO_{int}(SC_{\mathrm{L}})$. Морфизм АО-моделей над SC_{L} является включением тогда и только тогда, когда функторы *mod* и *str* переводят его во включения (т.е. в М-морфизм и в *sig*(М)-морфизм, соответственно).

Доказательство. Построим категории *AO, LAB, tr-AO* и функторы *il, mod, int, asp, str* из компонентов технологии SC_{L}, как описано в разд.4.2. Положим $\mathrm{M}_{AO} = \{\langle a, x\rangle \mid a \in \mathrm{M}, x \in sig(\mathrm{M})\} \subseteq \mathrm{Mor}\ AO$, проверим, что $(int^{-1}(\mathrm{L}), \mathrm{M}_{AO})$ – факторизационная система в *AO*. Пусть $\langle f, b\rangle : \langle A, l\rangle \to \langle B, h\rangle$ – произвольный *AO*-морфизм, $f = m \circ e$ – $(sig^{-1}(\mathrm{L}), \mathrm{M})$-разложение, $b = p \circ q$ и $(h \circ sig(m)) = n \circ s$ – $(\mathrm{L}, sig(\mathrm{M}))$-разложения. Поскольку $n \circ (s \circ sig(e)) = h \circ sig(m \circ e) = h \circ sig(f) = b \circ l = p \circ (q \circ l)$ – $(\mathrm{L}, sig(\mathrm{M}))$-разложения, существует *SIG*-изоморфизм *i* такой, что $p = n \circ i$ и $s \circ sig(e) = i \circ q \circ l$. Тогда $\langle f, b\rangle = \langle m, n\rangle \circ \langle e, i \circ q\rangle$ – $(int^{-1}(\mathrm{L}), \mathrm{M}_{AO})$-разложение с промежуточным объектом $\langle \mathrm{dom}\ m, s\rangle$. Диагональ коммутативного *AO*-квадрата $r \circ u = v \circ w$, где $r \in \mathrm{M}_{AO}$ и $int(w) \in \mathrm{L}$, представляет собой пару $\langle d, d'\rangle$, где d – диагональ *mod*-образа этого квадрата, d' – диагональ его *str*-образа.

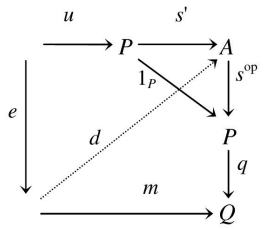
Предположим, что $\langle f, b\rangle \in \mathrm{Mor}\ \mathrm{M}_{AO}$ и $\langle B, h\rangle$ – дискретный объект, т.е. $\varepsilon_B \in \mathrm{Iso}\ c\text{-}DESC$ и $h \in \mathrm{Iso}\ SIG$ (ввиду доказательства теоремы 4.1). Поскольку $f \in \mathrm{M}$, $\varepsilon_A \in \mathrm{Iso}\ c\text{-}DESC$. Кроме того, равенство $(h \circ sig(f)) = b \circ l$ задает $(\mathrm{L}, sig(\mathrm{M}))$-разложение $sig(\mathrm{M})$-морфизма $h \circ sig(f)$, поэтому $l \in \mathrm{Iso}\ SIG$. Таким образом, $\langle A, l\rangle$ – также дискретный объект.

Осталось показать, что если $\langle f, b\rangle \in int^{-1}(\mathrm{L}) \cap \mathrm{M}_{AO}\text{-Init}$, то $f \in$ M-Init, так что категория Tr L, построенная из компонентов технологии $\boldsymbol{spec}(AO_{int}(SC_{\mathrm{L}}))$, содержится в *tr-AO* в качестве подкатегории, поэтому функтор 1_{AO} индуцирует вложение технологии $\boldsymbol{spec}(AO_{int}(SC_{\mathrm{L}}))_{\mathrm{L}}$ в $AO_{int}(SC_{\mathrm{L}})$. Пусть $sig(f) \circ k = sig(m)$ для некоторых *SIG*-морфизма k и M-морфизма $m : X \to B$. Если $(h \circ sig(m)) = n \circ s$ – (L, *sig*(M))-разложение, то имеется M_{AO}-морфизм $\langle m, n\rangle : \langle X, s\rangle \to \langle B, h\rangle$, поэтому существует *AO*-морфизм $k' : \langle X, s\rangle \to \langle A, l\rangle$ такой, что $int(k') = k$, и можно выбрать $k^{+} = mod(k')$, чтобы удовлетворить определению 5.1 для *f*. □

Следствие 5.1.5. Категория *tr-AO*, порожденная технологией SC_{L}, совпадает с категорией запятой $sig_{\mathrm{L}} \downarrow$ (Ob *SIG*, L), где функтор $sig_{\mathrm{L}} :$ Tr L $\to$ (Ob *SIG*, L) действует так же, как *sig*. Поэтому трансформация в АО-технологии над SC_{L} обладает экспликацией тогда и только тогда, когда ее область и кообласть находятся в аспектном ядре.

Доказательство. Для всякого *tr-AO*-морфизма $\langle t, b\rangle : \langle A, l\rangle \to \langle B, h\rangle$ имеем $b \in$ L, поскольку $b \circ l = h \circ sig(t) \in$ L и $l \in$ L, а ввиду следствия 5.1.1 класс L слабо сократим справа.

Далее, ввиду предложений 5.4 и 4.20 любая экспликация аспектной структуры АО-модели в АО-технологии над SC_{L} универсальна, поэтому эксплицируемая трансформация может иметь место только между элементами аспектного ядра. Установим, что любая их трансформация эксплицируема. Предположим, что АО-модели $\langle A, l\rangle$ и $\langle B, h\rangle$ обладают экспликациями $s : P \to A$ и $r : Q \to B$, соответственно. Пусть $q : P \to Q$ – экспликация действия *tr-AO*-морфизма $\langle t, b\rangle$ вдоль s и r, т.е. $q \circ s^{\mathrm{op}} = r^{\mathrm{op}} \circ t$. Ввиду трассируемости трансформации s имеем $sig(q) = b \in$ L. Проверим, что $q \in$ M-Init, при помощи предложения 5.1: найдем диагональ у произвольного коммутативного квадрата $q \circ u = m \circ e$, где $sig(e) \in$ Iso *SIG* и $m \in$ M. Пусть s' – *c-DESC*-морфизм, правый обратный к s^{op}. В силу того же предложения 5.1 коммутативный квадрат $(r^{\mathrm{op}} \circ t) \circ (s' \circ u) = (q \circ s^{\mathrm{op}}) \circ (s' \circ u) = m \circ e$ имеет диагональ d такую, что $d \circ e = s' \circ u$. Тогда $(s^{\mathrm{op}} \circ d) \circ e = s^{\mathrm{op}} \circ s' \circ u = u$, так что $s^{\mathrm{op}} \circ d$ – искомая диагональ исходного квадрата. Окончательно, имеем $q \in$ Mor Tr L. □



Следствие 5.1.6. Технология SC_{L} аспектно полна тогда и только тогда, когда любая *SIG*-диаграмма из класса $str \circ \mathbf{D}mod^{-1}(Conf)$ обладает копределом.

Доказательство. Необходимость вытекает из определений 4.17 и 4.4. Для доказательства достаточности ввиду предложения 4.23 (импликация (iii) $\Rightarrow$ (i)) и условия (iii) определения

5.2 нужно показать, что произвольная диаграмма $\Delta : X \to AO$ такая, что $mod \circ \Delta \in Conf$, аспектно детерминирована. Диаграмма $mod \circ \Delta$ обладает копределом ввиду условия (i) определения 2.3, а $int \circ \Delta$ – ввиду предложения 2.6. По условию, *SIG*-диаграмма $str \circ \Delta$ также обладает копределом, поэтому стрелка копредела $u = colim(\langle \beta^{\Xi}, 1_X \rangle)$, где $\Xi = il \circ asp \circ \Delta$, представляет собой объект копредела диаграммы Ξ в категории стрелок SIG^2. Непосредственно проверяется, что поскольку согласно следствию 5.1.1 выполняется $\mathrm{Difip}(\{\beta^{\Xi}_I \mid I \in \mathrm{Ob}\, X\}, sig(\mathrm{M}))$, имеет место $\mathrm{Difip}(u, sig(\mathrm{M}))$ (общее двойственное утверждение приводится в [159]). Следовательно, $u \in \mathrm{L}$, так что диаграмма Δ обладает копределом $\langle \mathrm{colim}\, mod \circ \Delta, \mathrm{colim}\, str \circ \Delta \rangle : \Delta \to \ulcorner \langle colim(mod \circ \Delta), u \rangle \urcorner$, поэтому функтор $\langle mod, str \rangle : AO \to c\text{-}DESC \times SIG$ поднимает ее копределы. Ввиду предложения 2.2 Δ аспектно детерминирована. □

Следствие 5.1.7. Если формальная технология *SC* скоординирована и для любой *AO*-диаграммы Δ *c-DESC*-диаграмма $mod \circ \Delta$ обладает копределом тогда и только тогда, когда им обладает *SIG*-диаграмма $str \circ \Delta$, то AO-технология над SC_{L}, порожденная любым фундаментальным функтором, скоординирована.

Доказательство. Проверим, что при указанных условиях любой фундаментальный функтор поднимает копределы произвольной диаграммы $\Delta : X \to AO$. Как в доказательстве теоремы 4.1, достаточно проверить функторы *mod* и *asp*.

Пусть $\delta : mod \circ \Delta \to \ulcorner A \urcorner$ – копредел. Ввиду скоординированности технологии *SC* и предложения 2.11, функтор *sig* сохраняет все копределы, поэтому $sig \circ \delta$ – копредел диаграммы $int \circ \Delta = sig \circ mod \circ \Delta$. По условию, *SIG*-диаграмма $str \circ \Delta$ обладает копределом. Как было указано в доказательстве следствия 5.1.6, стрелка копредела u такая, что $(\mathrm{colim}\, str \circ \Delta) \circ \langle \beta^{il \circ asp \circ \Delta}, 1_X \rangle = \ulcorner u \urcorner \circ (sig \circ \delta)$, принадлежит классу L. Поэтому $\langle \delta, \mathrm{colim}\, str \circ \Delta \rangle : \Delta \to \ulcorner \langle A, u \rangle \urcorner$ – копредел.

Пусть теперь $\sigma : asp \circ \Delta \to \ulcorner k \urcorner$ – копредел. Поскольку функтор $codom \circ il$ имеет правый сопряженный (таковым является $1_{\_} : A \mapsto 1_A$,), он сохраняет все копределы, следовательно коконус $\upsilon = codom \circ il \circ \sigma$ является копределом *SIG*-диаграммы $str \circ \Delta$. По условию, *c-DESC*-диаграмма $mod \circ \Delta$ обладает копределом, так что ввиду скоординированности технологии *SC* диаграмма $int \circ \Delta$ также обладает им. Как было указано в доказательстве следствия 5.1.6, стрелка копредела v такая, что $\upsilon \circ \langle \beta^{il \circ asp \circ \Delta}, 1_X \rangle = \ulcorner v \urcorner \circ \rho$, где $\rho = \mathrm{colim}\, int \circ \Delta$, принадлежит классу L, так что имеется копредел $\langle \rho, \upsilon \rangle : asp \circ \Delta \to \ulcorner v \urcorner$. Поскольку копредел определен однозначно с точностью до изоморфизма, существует *LAB*-изоморфизм i такой, что $\sigma = \ulcorner i \urcorner \circ \langle \rho, \upsilon \rangle$, поэтому $dom \circ il \circ \sigma = \ulcorner dom(il(i)) \urcorner \circ \rho$ – копредел диаграммы $int \circ \Delta$. По условию, функтор *sig* поднимает

все копределы, поэтому в *c-DESC* существует копредел $\theta : mod \circ \Delta \to \ulcorner B \urcorner$ такой, что $sig \circ \theta = dom \circ il \circ \sigma$. Тогда $\langle \theta, \upsilon \rangle : \Delta \to \ulcorner \langle B, k \rangle \urcorner$ – копредел. □

Следствие 5.1.8. Технология $AO_{mod}(SC_{\mathrm{L}})$ скоординирована, если обладают копределами следующие *SIG*-диаграммы:

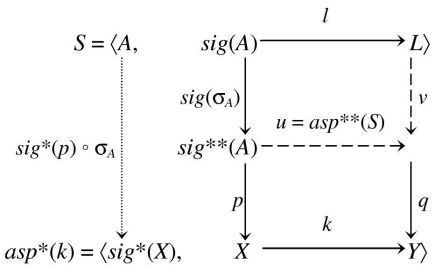
- $int \circ \Delta$ и $str \circ \Delta$ для любой *AO*-диаграммы Δ такой, что *c-DESC*-диаграмма $mod \circ \Delta$ обладает копределом;
- любая пара стрелок с общим началом, одна из которых принадлежит классу L.

Доказательство. Проверим, что при наличии указанных копределов функтор *mod* поднимает копределы произвольной диаграммы $\Delta : X \to AO$. Пусть $\delta : mod \circ \Delta \to \ulcorner A \urcorner$ – копредел. По условию, существуют копределы $\rho = \mathrm{colim}\ int \circ \Delta$ и $\upsilon = \mathrm{colim}\ str \circ \Delta$. Как было указано в доказательстве следствия 5.1.6, стрелка копредела $u = colim(\langle \beta^{\Xi}, 1_X \rangle)$, где $\Xi = il \circ asp \circ \Delta$, принадлежит классу L. Однако *SIG*-коконус $sig \circ \delta$ в общем случае не является копределом, поэтому процедура вычисления копредела *AO*-диаграммы, описанная в первой части доказательства следствия 5.1.7, может не сработать, и мы модифицируем ее следующим образом. Пусть r – стрелка копредела такая, что $sig \circ \delta = \ulcorner r \urcorner \circ \rho$. По условию, *SIG*-диаграмма $r : sig(A) \leftarrow colim(int \circ \Delta) \to colim(str \circ \Delta) : u$ обладает копределом – кодекартовым квадратом. Пусть l и k – его ребра, удовлетворяющие соотношению $l \circ r = k \circ u$. Ребро l, параллельное L-морфизму u, само принадлежит классу L, поскольку этот класс ввиду следствия 5.1.1 является первой компонентой факторизационной системы в *SIG* и тем самым замкнут относительно формирования кодекартовых квадратов. Следовательно, имеется *AO*-коконус $\langle \delta, \ulcorner k \urcorner \circ \upsilon \rangle : \Delta \to \ulcorner \langle A, l \rangle \urcorner$. Проверим, что он является копределом. Пусть $\gamma : \Delta \to \ulcorner \langle B, h \rangle \urcorner$ – произвольный *AO*-коконус с основанием Δ, b и q – стрелки копределов такие, что $mod \circ \gamma = \ulcorner b \urcorner \circ \delta$ и $str \circ \gamma = \ulcorner q \urcorner \circ \upsilon$. Имеем $\ulcorner h \circ sig(b) \circ r \urcorner \circ \rho = \ulcorner h \urcorner \circ (sig \circ (\ulcorner b \urcorner \circ \delta)) = \ulcorner h \urcorner \circ (int \circ \gamma) = (str \circ \gamma) \circ \langle \beta^{\Xi}, 1_X \rangle = \ulcorner q \urcorner \circ \upsilon \circ \langle \beta^{\Xi}, 1_X \rangle = \ulcorner q \circ u \urcorner \circ \rho$, а поскольку ρ – копредел, то $(h \circ sig(b)) \circ r = q \circ u$. Из определения кодекартова квадрата вытекает, что существует *SIG*-морфизм $w : \mathrm{codom}\ l \to \mathrm{codom}\ h$ такой, что $w \circ l = h \circ sig(b)$ и $w \circ k = q$. Следовательно, существует *AO*-морфизм $\langle b, w \rangle : \langle A, l \rangle \to \langle B, h \rangle$, удовлетворяющий соотношению $\gamma = \ulcorner \langle b, w \rangle \urcorner \circ \langle \delta, \ulcorner k \urcorner \circ \upsilon \rangle$. Из унивалентности функтора *mod* (см. теорему 4.1) вытекает единственность *AO*-морфизма, удовлетворяющего этому соотношению, поэтому он является искомой стрелкой копредела. □

Следствие 5.1.9. Если формальная технология *SC* структурируема и любая *c-DESC*-диаграмма, состоящая из пары стрелок с общим началом, одна из которых принадлежит классу *sig**(L), обладает копределом, то AO-технология над SC_{L}, порожденная любым фундаментальным функтором, структурируема.

Доказательство. Структурируемость технологии $AO_{mod}(SC_{\mathrm{L}})$ вытекает из теоремы 4.3. Отсюда и из структурируемости технологии SC вытекает, что технология $AO_{int}(SC_{\mathrm{L}})$ структурируема. Осталось проверить, что $AO_{asp}(SC_{\mathrm{L}})$ структурируема. Обозначим через σ единицу сопряжения $sig^{**} \dashv sig^{*}$. Для произвольной АО-модели над SC_{L} $S = \langle A, l : sig(A) \to L\rangle$ обозначим через Σ_S SIG-диаграмму $sig(\sigma_A) : sig^{**}(A) \leftarrow sig(A) \to L : l$. Поскольку $l \in \mathrm{L}$, по условию c-$DESC$-диаграмма $sig^{*} \circ \Sigma_S$ обладает копределом, поэтому ввиду предложения 2.5 диаграмма $\Sigma_S = sig \circ sig^{*} \circ \Sigma_S$ сама обладает копределом. Пусть u и v – ребра копредела, удовлетворяющие соотношению $sig(\sigma_A) \circ u = v \circ l$. Поскольку $l \in \mathrm{L}$, то и $u \in \mathrm{L}$ (напомним, что в силу следствия 5.1.1 класс L замкнут относительно кодекартовых квадратов), так что положим $asp^{**}(S) = u$. Действие функтора asp^{**} на морфизмах зададим следующим образом. Выберем произвольно АО-модель $T = \langle B, m : sig(B) \to M\rangle$ и морфизм $\langle f, b\rangle : S \to T$. Тройка морфизмов $(sig^{**}(f) : sig^{**}(A) \to sig^{**}(B), sig(f) : sig(A) \to sig(B), b : L \to M)$ представляет собой естественное преобразование Σ_S в Σ_T, которое мы обозначим через φ. Положим $asp^{**}(\langle f, b\rangle) = \langle sig^{**}(f), colim(\langle \varphi, 1_V\rangle)\rangle : asp^{**}(S) \to asp^{**}(T)$. Непосредственно проверяется, что таким способом действительно задается функтор, действующий из AO в LAB.

Проверим, что этот функтор сопряжен слева к asp^{*}, и что коединица этого сопряжения тождественна. Если $S \in asp^{*}(\mathrm{L})$, то $A \in sig^{*}(\mathrm{Ob}\ SIG)$ (см. конструкцию функтора asp^{*} в доказательстве теоремы 4.1), поэтому $\sigma_A = 1_A$, следовательно копредел диаграммы Σ_S можно выбрать так, чтобы выполнялось соотношение $asp^{**}(S) = l$, так что функтор asp^{**} является левым обратным к asp^{*}. Для произвольных L-морфизма $k : X \to Y$ и LAB-морфизма $\langle p, q\rangle : u \to k$ требуется найти единственный морфизм из S в $asp^{*}(k) = \langle sig^{*}(X), k\rangle$, переходящий в $\langle p, q\rangle$ под действием функтора asp^{**}. Как показано на диаграмме ниже, таким морфизмом служит $\langle sig^{*}(p) \circ \sigma_A, q \circ v\rangle : S \to asp^{*}(k)$. Его единственность вытекает из унивалентности функтора mod и того, что в силу определения сопряжения функторов $sig^{*}(p) \circ \sigma_A$ – это единственный морфизм из A в $sig^{*}(X)$, переходящий в p под действием функтора sig^{**}.



Из этой же диаграммы видно, что единица сопряжения $asp^{**} \dashv asp^{*}$ переводит S в морфизм $\langle \sigma_A, v \rangle : S \to asp^{*}(asp^{**}(S))$. □

Следствие 5.1.3 позволяет выявить функториальную природу процедуры синтеза трансформационных технологий. Построим категорию, класс объектов которой состоит из всех пар $\langle SC, \mathrm{L} \rangle$, где SC – технология специфицирования, поддерживающая L-разметки. Морфизмом пары $\langle SC, \mathrm{L} \rangle$ в $\langle SC', \mathrm{L}' \rangle$ служит любой **SPEC**-морфизм $\langle cm, sm \rangle : SC \to SC'$ такой, что cm – ***desc***-разметка, sm – изоморфизм категорий и $sm(\mathrm{L}) \subseteq \mathrm{L}'$. Получается категория, которую мы обозначим через **SPL**, снабженная забывающим функтором ***specl*** : **SPL** → **SPEC** : $\langle SC, \mathrm{L} \rangle \mapsto SC$. Согласно следствию 5.1.3, существует вложение ***ispl*** : **SPL** ↪ **ARCH** : $\langle SC, \mathrm{L} \rangle \mapsto SC_{\mathrm{L}}$, причем ***spec*** ∘ ***ispl*** = ***specl***.

Привлекая следствие 5.1.4, покажем, что процедура АО-расширения естественна относительно этого вложения. Будем для краткости обозначать через $s\text{-}AO(AR)$ технологию специфицирования $\boldsymbol{spec}(AO_{int}(AR))$. Пусть **SPLL** – подкатегория в **SPL**, класс объектов которой совпадает с Ob **SPL**, а морфизмом объекта $\langle SC, \mathrm{L} \rangle$ в $\langle SC', \mathrm{L}' \rangle$ служит любой **SPL**-морфизм $\langle cm, sm \rangle : \langle SC, \mathrm{L} \rangle \to \langle SC', \mathrm{L}' \rangle$, удовлетворяющий условию $sm(\mathrm{L}) = \mathrm{L}'$ (о нетривиальности класса таких морфизмов свидетельствует теорема 5.1). Построим для произвольного **SPLL**-морфизма функтор $aom : AO \to AO'$, индуцирующий **SPLL**-морфизм $\langle aom, sm \rangle : \langle s\text{-}AO(SC_{\mathrm{L}}), \mathrm{L} \rangle \to \langle s\text{-}AO(SC'_{\mathrm{L}'}), \mathrm{L}' \rangle$. Начнем с установленного в разд.4.2 факта, что категория АО-моделей AO, построенная из компонентов L-трансформационной технологии SC_{L}, является объектом предела диаграммы со схемой $\boldsymbol{V}^{\mathrm{op}}$ $sig : c\text{-}DESC \to SIG \leftarrow LAB : dom \circ il$, где LAB – полная подкатегория в категории стрелок $SIG^{\boldsymbol{2}}$, класс объектов которой состоит из всех L-морфизмов. Легко проверить, что **SPLL**-морфизм $\langle cm, sm \rangle$ индуцирует естественное преобразование таких диаграмм, которое мы обозначим через ξ, с компонентами $cm : c\text{-}DESC \to c\text{-}DESC'$, $sm : SIG \to SIG'$, $sml : LAB \to LAB'$, где sml – функтор, действующий так же, как $sm^{\boldsymbol{2}}$. Отметим, что композиция **SPLL**-морфизмов индуцирует композицию таких естественных преобразований. Положим $aom = lim(\langle \xi, 1_{\boldsymbol{V}^{\mathrm{op}}} \rangle) : AO \to AO'$. По определению предела имеем $mod' \circ aom = cm \circ mod$ (откуда $int' \circ aom = sig' \circ mod' \circ aom = sig' \circ cm \circ mod = sm \circ sig \circ mod = sm \circ int$) и $asp' \circ aom = sml \circ asp$ (откуда $str' \circ aom = codom' \circ il' \circ asp' \circ aom = codom' \circ il' \circ sml \circ asp = sm \circ codom \circ il \circ asp = sm \circ str$). Следовательно, имеем

$$aom : \langle A, l \rangle \mapsto \langle cm(A), sm(l) \rangle, \langle f, b \rangle \mapsto \langle cm(f), sm(b) \rangle.$$

Функтор aom унивалентен, поскольку правая часть соотношения $mod' \circ aom = cm \circ mod$ состоит из унивалентных функторов. Кроме того, функтор aom обладает левым сопряженным, который имеет вид

$$aom^* : AO' \to AO : \langle A', l' \rangle \mapsto \langle cm^*(A'), sm^{-1}(l') \rangle, \langle f', b' \rangle \mapsto \langle cm^*(f'), sm^{-1}(b') \rangle,$$

единица этого сопряжения тождественна. Далее, функтор *aom* сохраняет копределы всех конфигураций технологии *s-AO*(SC_{L}), поскольку функторы $cm \circ mod$, *int* и *str* сохраняют копределы любой конфигурации, а функтор *sm*, будучи изоморфизмом категорий, сохраняет копределы любой диаграммы. Проверим, что функтор *aom* переводит конфигурации технологии *s-AO*(SC_{L}) в конфигурации технологии *s-AO*($SC'_{\mathrm{L}'}$), т.е. что $sm \circ AOInt_{int} \subseteq AOInt_{int'}$. Выберем произвольно диаграммы $\Theta \in AOInt_{int}$ и $\Delta \in \mathbf{D}int'^{-1}(\{sm \circ \Theta\})$, положим $\Sigma = aom^* \circ \Delta$. Имеем $aom \circ \Sigma = \Delta$ и $int \circ \Sigma = sm^{-1} \circ sm \circ int \circ \Sigma = sm^{-1} \circ int' \circ \Delta = \Theta$, так что $mod' \circ \Delta = mod' \circ aom \circ \Sigma = cm \circ mod \circ \Sigma \in Conf'$ и кроме того, диаграмма $str' \circ \Delta = str' \circ aom \circ \Sigma = sm \circ str \circ \Sigma$ обладает копределом. Отсюда, как в доказательстве следствия 5.1.6, выводится, что диаграмма Δ аспектно детерминирована. Поэтому аспектно детерминированной является и любая ее накачка *tr-AO*'-морфизмами, так что $sm \circ \Theta \in AOInt_{int'}$.

Таким образом, пара $\langle aom, sm \rangle$ действительно представляет собой **SPLL**-морфизм АО-технологий, так что ввиду следствия 5.1.3 существует функтор

$$\boldsymbol{aom} : \mathbf{SPLL} \to \mathbf{ARCH} : \langle SC, \mathrm{L} \rangle \mapsto AO_{int}(SC_{\mathrm{L}}), \langle cm, sm \rangle \mapsto \langle aom, sm, aom(-^{\mathrm{op}})^{\mathrm{op}} \rangle.$$

Существует естественное преобразование этого функтора в стандартное вложение $\boldsymbol{ispl} \circ \boldsymbol{ispll} : \mathbf{SPLL} \hookrightarrow \mathbf{ARCH}$, где $\boldsymbol{ispll} : \mathbf{SPLL} \hookrightarrow \mathbf{SPL}$ – вложение подкатегории. Это естественное преобразование сопоставляет каждому **SPLL**-объекту $\langle SC, \mathrm{L} \rangle$ **ARCH**-морфизм $ao_{int} : AO_{int}(SC_{\mathrm{L}}) \to SC_{\mathrm{L}}$, построенный в доказательстве следствия 4.4.5. Это иллюстрирует естественность процедуры АО-расширения трансформационных технологий.

В заключение раздела покажем, что ограничения, накладываемые на **SPEC**-морфизм $\langle cm, sm \rangle : SC \to SC'$ в условии теоремы 5.1 в целях переноса способности своей области поддерживать некоторый класс разметок на кообласть, не могут быть существенно ослаблены. Мы будем использовать тот факт, что технология специфицирования сценариев $SS = \boldsymbol{spec}(SM)$ не поддерживает (Iso **Set**)-разметки (см. предложение 5.9 ниже). **SPEC**-морфизм $\langle 1_{\mathbf{Pos}}, |-| \rangle : TS \to SS$, где $TS = \boldsymbol{conf}^*(\boldsymbol{conf}(SS))$, является ***conf***-трассой, но не обладает свойством аспектности, и *TS* поддерживает (Iso **Pos**)-разметки. В свою очередь, **SPEC**-морфизм $\langle id, 1_{\mathbf{Set}} \rangle : US \to SS$, где $US = \boldsymbol{conf}^*(\boldsymbol{conf}^{**}(SS))$, изоаспектен, но не является ***conf***-трассой, и *US* поддерживает (Iso **Set**)-разметки.

5.2 Формальный подход к моделированию данных

Хранилище данных находится в основе архитектуры любой информационно-управляющей системы, поэтому ограничения качества непосредственно влияют на проектные решения по его организации, а снижение затрат на выработку этих решений способно значительно повысить эффективность всего процесса создания системы. В связи с этим интерес представляет формальная технология моделирования данных достаточно общего вида, обеспечивающая широкие возможности трассирования и использования аспектно-ориентированных приемов. Она требуется в качестве семантической базы при применении информационно-ориентированного стиля проектирования больших информационно-управляющих систем.

Наиболее общая формальная модель массива данных представляет собой множество, состоящее из хранящихся в нем информационных элементов. Действие по интеграции массивов – это в точности любое отображение множеств, поскольку каждому элементу массива-компонента сопоставляется единственный элемент массива-системы. Массивы данных могут комплексироваться друг с другом без всяких ограничений, так что любая диаграмма множеств и отображений является допустимой конфигурацией. Получается формальная технология конфигурирования $DC = \langle \mathbf{Set}, \mathrm{Ob}\ \mathbf{D}(\mathbf{Set}) \rangle$. Как легко проверить, категория **Set** не имеет полных корефлективных подкатегорий с унивалентным корефлектором, не эквивалентных ей (действительно, коединица такой корефлексии состоит из эпимономорфизмов, а $\mathrm{Epi}\ \mathbf{Set} \cap \mathrm{Mono}\ \mathbf{Set} = \mathrm{Iso}\ \mathbf{Set}$). Поэтому переход к технологии специфицирования осуществляется формально, при помощи функтора ***conf****: получается формальная технология над **Set** $DS = \boldsymbol{conf}^*(DC) = \langle \mathbf{Set}, \mathrm{Ob}\ \mathbf{D}(\mathbf{Set}), 1_{\mathbf{Set}} \rangle$. Таким образом, специфицирование моделей данных фактически сводится к их конфигурированию (этот факт хорошо известен из практики информационного моделирования посредством диаграмм типа «сущность–связь»). Технология *DS* элементарна (терминальным объектом в **Set** служит одноэлементное множество), структурируема и скоординирована (предложение 2.12), поддерживает параллелизм (любое множество раскладывается в прямую сумму одноэлементных множеств). Она описывает специфицирование массивов данных в самом общем виде, без уточнения способа их реализации.

Частные информационные технологии оперируют с разнообразными специальными видами множеств, поэтому формально описываются подтехнологиями в *DS*. Например таблица в реляционной базе данных представляет собой подмножество прямого произведения основных множеств типов атрибутов (domains, см. например [18]). Интеграция двух таблиц выполняется путем формирования внешнего ключа – отображения множества всех записей одной таблицы во множество записей второй. Таким образом, морфизмы таких множеств и составленные из них диаграммы описывают конструкции реляционной алгебры. При моделировании объектно-

ориентированных баз данных множества сопоставляются классам, а отображения – декларации атрибутов и наследованию [196]. Подтехнологиями в *DS* можно описать и модели теории родов структур Н.Бурбаки, о которых говорилось в разд.1.4. Также подтехнологии в *DS* возникают при действии на формальные технологии проектирования над **Set** функтора «специфицирования интерфейсов» ***dspec*** : **ARCH** → **SPEC** из разд.4.5 (ср. следствие 5.1.1). Этот факт иллюстрирует известное из практики программирования правило, что для специфицирования интеграционных интерфейсов программных компонентов нужно в первую очередь определить структуры данных, для обмена которыми организуется интеграция. Отметим, что имеется много инструментов для решения задач типа перехода от платформенно-независимых моделей данных (PIM) к платформенно-зависимым (PSM), путем автоматической трансляции схем данных в машинно-читаемые тексты на языках SQL, XML и т.п. (например ERwin, см. [23] и др.)

Чтобы предоставить формальную базу для этих и других приемов проектирования моделей данных, выявим классы разметок, которые поддерживает технология *DS*. Согласно перечню всех факторизационных систем в **Set**, приведенному в [136, пример 14.2(4)], условия определения 5.2 выполняются для нее только в случае, если в качестве класса разметок L выбрать один из двух классов отображений: класс Bij всех биекций либо класс Surj всех сюръекций. В Bij-трансформационной технологии над *DS* класс всех трансформаций совпадает с Bij, поэтому она фактически воспроизводит специфицирование интерфейсов (и не имеет нетривиального аспектно-ориентированного расширения, ввиду предложения 4.2). С классом Surj дело обстоит иначе: в DS_{Surj} в силу предложения 5.5 класс всех трансформаций состоит из всех тотальных антифункциональных бинарных отношений (Tr Surj = (Ob **Set**, Surj)), а класс всех включений – из всех инъекций. Любое действие по интеграции массивов данных однозначно (с точностью до изоморфизма) раскладывается в композицию трассы и включения.

Обозначим через *DM* технологию DS_{Surj}. Она хорошо поддается аспектной ориентации: категория АО-моделей над *DM* представляет собой полную подкатегорию в $\mathbf{Set}^{\mathbf{2}}$, класс объектов которой состоит из всех сюръекций, мы будем обозначать ее через **Surj**. Поэтому *DM* аспектно универсальна: разметка любого **Surj**-объекта представляет собой трассу его единственной экспликации. Кроме того, в силу следствия 5.1.6 *DM* аспектно полна. Более того, ввиду предложения 4.3 среди АО-технологий над *DM* можно выбрать каноническую – это $AO_{mod}(DM)$, совпадающая с ядерной АО-технологией над *DM* (и с $AO_{int}(DM)$), которую мы будем обозначать через ***AODM***. Пусть *isurj* : **Surj** ↪ $\mathbf{Set}^{\mathbf{2}}$ – вложение. С учетом следствия 5.1.5, имеем

$$\boldsymbol{AODM} = \langle \mathbf{Surj}, \mathrm{Ob}\,\mathbf{D}(\mathbf{Surj}), \mathit{dom} \circ \mathit{isurj} : \mathbf{Surj} \to \mathbf{Set}, ((\mathrm{Ob}\,\mathbf{Set}, \mathrm{Surj})^{\mathbf{2}})^{\mathrm{op}} \rangle.$$

Это технология над **Set**, элементарная в силу следствия 4.1.2. Она скоординирована (следствие 5.1.7), причем правым сопряженным к функтору *dom* ∘ *isurj* служит функтор сингулярной раз-

метки, который переводит пустое множество $\varnothing$ в $1_{\varnothing}$, а любому непустому множеству A сопоставляет аспект $!_A : A \to \mathbf{1}$. Кроме того, технология ***AODM*** структурируема (следствие 5.1.9, либо условие (iii) предложения 4.3), причем дискретная структура совпадает с аспектной и выделяется функтором $codom \circ isurj : \mathbf{Surj} \to \mathbf{Set}$. Ввиду следствия 5.1.4, технология ***spec***(***AODM***) поддерживает Surj-разметки, что открывает путь к повторному (и далее, многократному) проведению процедуры аспектно-ориентированного расширения, которая порождает также скоординированные и структурируемые технологии над **Set**. Таким путем строятся модели массивов данных, снабженных несколькими взаимно независимыми разметками.

Surj-объекты получаются из исходных массивов данных путем разметки элементов именами классов задач, оперирующих с ними, так что **Surj** эквивалентна категории **Equ** всех множеств, снабженных отношением эквивалентности, и всех их гомоморфизмов. Отсюда можно вывести, что ***AODM*** действительно является канонической технологией аспектно-ориентированного моделирования данных в том смысле, что любой функтор, порождающий АО-технологию над *DM*, по существу (с точностью до эквивалентности категорий) совпадает либо с $dom \circ isurj$, либо с $1_{\mathbf{Surj}}$. Действительно, проверим, что любая полная корефлективная подкатегория в **Equ** с унивалентным корефлектором эквивалентна либо **Set**, либо **Equ**. Пусть *XEqu* – полная подкатегория в **Equ**, $xequ : \mathbf{Equ} \to XEqu$ – унивалентный корефлектор, ε – коединица корефлексии. Поскольку ε состоит из биекций, сохраняющих отношение эквивалентности, в *XEqu* найдется множество произвольной мощности с равенством в качестве отношения эквивалентности. Предположим, что имеется еще некоторый *XEqu*-объект X с отношением эквивалентности, не совпадающим с равенством, он содержит элементы a и b, удовлетворяющие соотношениям $a \neq b$ и $a \sim b$. Покажем, что в этом случае ε состоит из изоморфизмов. Выберем произвольно **Equ**-объект Y и элементы $p, q \in xequ(Y)$, удовлетворяющие соотношению $\varepsilon_Y(p) \sim \varepsilon_Y(q)$. Определим отображение $f : X \to Y$ следующим образом:

$$f(x) = \begin{cases} \varepsilon_Y(p), & x \sim a \text{ и } x \neq b, \\ \varepsilon_Y(q) & \text{иначе.} \end{cases}$$

Ясно, что оно является **Equ**-морфизмом, поэтому по определению корефлексии существует **Equ**-морфизм $xequ(f) : X \to xequ(Y)$, причем $\varepsilon_Y \circ xequ(f) = f$, так что $p = xequ(f)(a) \sim xequ(f)(b) = q$. Ввиду произвольности выбора p и q отсюда следует, что ε_Y – изоморфизм. Следовательно, *XEqu* либо состоит из множеств с равенством и эквивалентно **Set**, либо эквивалентно **Equ**.

Реализовать любую разметку массива данных нетрудно ввиду аспектной универсальности технологии *DM*: ее можно задать действием по его интеграции в массив-справочник классов задач [2]. Например, в информационно-управляющих системах ведется общий реестр обо-

рудования, включающий технологические узлы объекта, измерительные приборы и исполнительные механизмы, вычислительные устройства и др. В целях разделения единиц оборудования по их назначению, в структуру реестра добавляется специальный атрибут-метка, имеющий перечислимый тип (классификатор оборудования).

Связывание в технологии моделирования данных существует всегда, причем всегда является частным случаем модульной компоновки. Дело в том, что любая **Surj**-диаграмма является конфигурацией в ***AODM***, а функтор $codom \circ isurj$ сохраняет (а значит, в силу предложения 4.10 детерминирует) произведения (проверяется непосредственно) и копределы (поскольку ввиду утверждения (iii) предложения 4.3 он обладает правым сопряженным). Выберем произвольно сюръекции $b : B \to L$, $w : W \to M$, $c : C \to K$, и пару **Surj**-морфизмов $\langle j, p\rangle : b \leftarrow c \to w : \langle e, q\rangle$. Пусть $s : X \to Y$ – результат связывания этой пары, т.е. копредел схемы связывания $\langle j, p\rangle : b \leftarrow c \to c \times w : \langle\langle 1_C, e\rangle, \langle 1_K, q\rangle\rangle$, которую мы будем обозначать через Ξ. Он строится согласно общему правилу вычисления копределов в категории стрелок. А именно, множество X – это вершина копредела **Set**-диаграммы $dom \circ isurj \circ \Xi$, имеющей вид $j : B \leftarrow C \to C \times W : \langle 1_C, e\rangle$. Она получается из раздельного объединения $B \coprod (C \times W)$ путем амальгамирования – факторизации по отношению эквивалентности, порожденному множеством пар $\{(j(x), (x, e(x))) \mid x \in C\}$ [30, разд.3.14]. Аналогично, множество Y представляет собой вершину копредела **Set**-диаграммы $codom \circ isurj \circ \Xi$, имеющей вид $p : L \leftarrow K \to K \times M : \langle 1_K, q\rangle$ – это результат факторизации множества $L \coprod (K \times M)$ по отношению эквивалентности, порожденному множеством пар $\{(p(x), (x, q(x))) \mid x \in K\}$. Имеем $s = colim(\langle \beta^{isurj \circ \Xi}, 1_V\rangle)$, где $\beta^{isurj \circ \Xi} = (b, c, c \times w)$ – естественное преобразование диаграммы $dom \circ isurj \circ \Xi$ в $codom \circ isurj \circ \Xi$.

Связывание имеет прозрачный смысл для реляционных таблиц: здесь спецификация связывания описывает добавление к базе атрибута типа «список ссылок на совет» (отношение вида «многие-ко-многим»), с использованием связки в качестве источника ссылочных ключей. Примером такой спецификации может служить схема данных из разд.1.3, в которой связкой выступает таблица компонентов измерительных каналов (`Measuring channel components`). Добавление атрибута в результате связывания является неразрушающим в том смысле, что структура базовой таблицы не меняется, поэтому оно не требует значительных затрат. Такой прием характерен для аспектно-ориентированного синтеза информационных моделей сложных систем управления, объектам которых свойственно вступать в многообразные и изменчивые отношения [2]. Здесь метки связки обозначают функции (роли) отношений между сущностями, описываемыми в таблицах базы и совета. Отношения, помеченные ролями, возникают между видами объектов и средств управления (онтологические отношения), их типами и марками (нормативные), и экземплярами (фактические). Например, на онтологическом уровне описыва-

ются возможности установки различных видов приборов, реализующих те или иные функции управления, на различных видах зданий и сооружений, на нормативном – правила выбора марок приборов для оснащения зданий различных серий, на фактическом – результаты работ по монтажу конкретных приборов в конкретных зданиях.

Любая АО-модель данных допускает разделение ответственности. Поэтому для моделирования данных достаточно применять традиционные модульные подходы, специализированных технологий АОП не требуется. Аспектно-ориентированный подход здесь применяется фактически только на концептуальном уровне (об этом свидетельствует пример реализации связывания в реляционной базе данных). Примером явной реализации разделения ответственности служит секционирование (partitioning) больших реляционных таблиц [126] – физическое разделение на идентичные по структуре секции, в каждую из которых попадают все записи, обладающие одним и тем же значением некоторого атрибута (в общем случае – некоторой функции от значений атрибутов). Если в качестве этого атрибута выбрать метку класса задач записи, то секционирование позволяет повысить производительность без дополнительных затрат труда, исключая конкуренцию компонентов, автоматизирующих разные классы задач, за доступ к общей таблице. Таким образом, конструкции в технологии ***AODM*** позволяет построить формальную семантическую модель секционирования.

Этот пример позволяет предположить, что разделение ответственности в моделях данных фактически представляет собой распараллеливание. Точная формулировка и доказательство этого предположения основывается на понятии слоя (fibre [136, определение 5.4]) в конкретной категории. Напомним, что для произвольной конкретной категории, т.е. пары $(C, ff : C \to D)$, состоящей из категории C и унивалентного функтора ff с областью C, слоем D-объекта X называется подкатегория в C вида $(ff^{-1}(X), ff^{-1}(1_X))$. Ввиду унивалентности слой является предпорядком.

Предложение 5.8. Разделение ответственности индуцирует эквивалентность между слоем произвольного непустого множества A в конкретной категории **Surj** и категорией сумм $Sums \downarrow A$.

Доказательство. По определению, слой множества A в категории **Surj** представляет собой подкатегорию в ней, класс объектов которой состоит из всех сюръекций с областью A, а класс морфизмов – из всех $\mathbf{Set}^2$-морфизмов вида $\langle 1_A, b\rangle : l \to h$, где l, h – сюръекции с областью A. Разделение ответственности произвольной сюръекции $l : A \to L$ состоит из включений в l аспектов $l_x : l^{-1}(x) \to \{x\}$, по одному для каждого $x \in L$ (т.е. L служит схемой дискретной **Surj**-диаграммы – основания коконуса, задающего разделение ответственности). Функтор $\mathbf{D}(dom \circ isurj)$ переводит разделение ответственности в **Set**-коконус, который мы будем обозначать через $soc(l)$ – это сумма с вершиной A. Произвольный морфизм объектов слоя $\langle 1_A, b\rangle : l \to h$

удовлетворяет условию $b \circ l = h$, поэтому задает $(\mathbf{D}(\mathbf{Set}) \downarrow \ulcorner A \urcorner)$-морфизм $\langle \zeta, b \rangle : soc(l) \to soc(h)$, где ζ состоит из включений множеств $l^{-1}(x)$ в $h^{-1}(b(x))$, $x \in L$. Непосредственно проверяется, что таким способом действительно задается функтор *soc*, действующий из слоя в категорию сумм. Он является эквивалентностью категорий: сопряженный к нему переводит произвольную сумму $\langle \pi, !_X \rangle : \Delta \to \ulcorner A \urcorner$, где X – схема диаграммы Δ, в сюръекцию $p : A \to \mathrm{Ob}\, X$, сопоставляющую каждому $a \in A$ единственный X-объект I такой, что $a \in \pi_I(\Delta(I))$. □

Отсюда, в частности, вытекает, что технология моделирования данных *DM* аспектно отражает любой оптимизируемый класс $1_{\mathbf{Set}}$-разбиений, поскольку каждое такое разбиение по определению является суммой.

5.3 Формальный подход к моделированию сценариев исполнения процессов

Моделирование сценариев относится к числу основных работ начальных этапов жизненного цикла любой интерактивной системы, когда вырабатываются требования к ней [57]. Сценарий – это последовательность действий и взаимодействий, происходящих при определенных условиях, изложенная без предложений с «если» и ветвления [57, с.259]. (Здесь слово «последовательность» следует понимать в широком смысле, т.к. сценарий может содержать взаимно независимые параллельные события, ни одно из которых не следует за другим). Участниками взаимодействия, описываемого сценариями, могут быть программные системы и компоненты, аппаратные единицы и оборудование, пользователи и организации. Поэтому моделирование сценариев позволяет выявить состав и структуру процессов, в выполнении которых может участвовать система, информационных единиц, с которыми они оперируют, и возможных областей их автоматизации. В результате формируются требования к системе, составляющие область применения двух стилей проектирования: взаимодействия и потокового.

Моделирование сценариев имеет хорошо разработанную формальную основу [236, 241]. Базовым математическим представлением сценария является частично упорядоченное множество (poset). Его элементы отвечают атомарным событиям, составляющим ход его выполнения. Частичный порядок естественным образом возникает из причинно-следственных связей между событиями. Заметим, что мы не налагаем условие конечности сценариев, поскольку на этапе разработки требований могут возникать наборы действий, не приходящие к завершению. Действиями по интеграции сценариев компонентов в сценарии системы являются в точности все монотонные отображения, поскольку ни события, ни взаимодействия не могут быть «забыты». Таким образом, модели сценариев образуют категорию **Pos**.

Диаграмма сценариев рассматривается как допустимая конфигурация, только если результат ее сборки задан явно, без привлечения структурных операций (за исключением тривиального раздельного объединения). Такое положение дел является типичным при разработке требований, когда отсутствуют знания, позволяющие задать мощные структурные правила комбинирования моделей. Таким образом, все конфигурации сценариев образуют класс *CPos* всех копроизведений **Pos**-коконусов. В качестве примера комплексирования сценариев приведем проектирование единого журнала событий. Журнал формально задается множеством вещественных чисел $\mathbb{R}$ с естественным линейным порядком, описывающим стрелу физического времени. Регистрация событий, образующих сценарий X, в журнале формально описывается монотонным отображением X в $\mathbb{R}$, а оснащение системы, описываемой конфигурацией Δ, единым журналом – диаграммой в форме коконуса с основанием Δ и вершиной $\mathbb{R}$.

В то же время все результаты настоящего раздела останутся в силе, если считать конфигурацией даже любой элемент наибольшего класса **Pos**-диаграмм, формально способных служить конфигурациями – таким классом является потенциал комплексируемости забывающего функтора $|{-}| : \mathbf{Pos} \to \mathbf{Set}$. Действительно, интерфейсом сценария естественным образом выступает множество его событий, получаемое путем «забывания» их упорядочения. Это по существу единственный возможный способ выделения нетривиальных интерфейсов: любая полная корефлективная подкатегория в **Pos** с унивалентным корефлектором эквивалентна либо **Set**, либо **Pos**. Этот факт доказывается дословно так же, как аналогичное свойство категории **Equ**, доказанное в разд.5.2 (с заменой символа отношения эквивалентности ~ символом отношения порядка ≤). Таким образом, получается тройка $SS = \langle \mathbf{Pos}, CPos, |{-}| : \mathbf{Pos} \to \mathbf{Set} \rangle$, которая является формальной технологией специфицирования в силу предложения 2.7. (Напомним, что левым сопряженным к забывающему функтору $|{-}|$ выступает функтор дискретного упорядочения $id : \mathbf{Set} \to \mathbf{Pos} : S \mapsto \langle S, = \rangle$, причем он имеет левый сопряженный с тождественной коединицей, сопоставляющий каждому частично упорядоченному множеству множество компонент связности его порядка, так что технология *SS* структурируема).

Формальная технология моделирования сценариев была кратко описана в разд.2.3, ряд ее свойств (в частности, структурируемость) был установлен на протяжении гл.2, 4. Здесь мы покажем, что она является единственной трансформационной технологией, которую можно получить из *SS*, и проанализируем порождаемые ею аспектно-ориентированные технологии.

Предложение 5.9. Технология *SS* поддерживает L-разметки тогда и только тогда, когда L = Surj, причем класс всех SS_{Surj}-включений совпадает с классом всех регулярных **Pos**-мономорфизмов, и $SS_{\mathrm{Surj}} = SM$.

Доказательство. Ввиду следствия 5.1.1 в качестве L, как и для технологии специфицирования данных *DS*, может выступать либо класс Bij всех биекций, либо класс Surj всех сюръекций. Однако ввиду предложения 5.7 технология *SS*, будучи структурируемой, не поддерживает Bij-разметки. С сюръекциями дело обстоит иначе: проверим условия определения 5.2 для технологии *SS* и класса Surj.

(i). В силу аксиомы выбора любое сюръективное отображение множеств является ретракцией.

(ii). В качестве M можно выбрать класс всех регулярных **Pos**-мономорфизмов, поскольку забывающий функтор $|\text{–}| : \mathbf{Pos} \to \mathbf{Set}$ сохраняет и отражает эпиморфизмы, категория **Pos** имеет факторизационную систему (Epi, RegMono) [136, пример 14.23(1)], и для произвольного **Pos**-мономорфизма вида $m : X \to id(S)$ выполняется соотношение $X = id(|X|)$.

(iii). Класс *CPos* замкнут относительно любых накачек.

Осталось проверить, что Tr Surj = $r\text{-}Pos^{\mathrm{op}}$. Сначала выберем произвольное сюръективное отображение частично упорядоченных множеств $f : T \to S$ такое, что условие $f(y) < f(y')$ эквивалентно $y < y'$ для любых $y, y' \in T$ – все такие отображения образуют класс Mor $r\text{-}Pos^{\mathrm{op}}$ (см. разд.2.3). Ясно, что f монотонно. Выберем произвольно отображения множеств $m : X \to S \in$ RegMono **Pos** и $k : X \to T$ такие, что $f(k(x)) = m(x)$ для любого $x \in X$, и проверим, что k монотонно, откуда будет следовать, что $f \in$ (RegMono **Pos**)-Init. Пусть произвольные $x, x' \in X$ удовлетворяют условию $x < x'$. Поскольку m инъективно, $m(x) < m(x')$, следовательно $k(x) < k(x')$ ввиду выбора f.

Обратно, пусть $t : P \to Q$ – произвольное сюръективное RegMono-инициальное отображение частично упорядоченных множеств. Выберем произвольно $p, p' \in P$ такие, что $t(p) < t(p')$, и проверим, что $p < p'$, откуда будет следовать, что $t^{\mathrm{op}} \in$ Mor $r\text{-}Pos$. Рассмотрим отображения $s : \mathbf{2} \to P : 0 \mapsto p, 1 \mapsto p'$ и $n : \mathbf{2} \to Q : x \mapsto t(s(x))$. Ясно, что $n \in$ RegMono **Pos**, поэтому s монотонно ввиду выбора t, следовательно $p < p'$. □

Напомним, что трасса трансформации сценариев разбивает свою область на совокупность прообразов точек, хорошо упорядоченную (well-ordered) в том смысле, что для любых x, y, z и u таких, что $t(x) = t(y) \neq t(z) = t(u)$, условие $x \leq z$ влечет $y \leq u$. Такая трактовка трансформаций предлагалась в [179], однако их взаимосвязь с регулярными (эквивалентно, инициальными, ввиду предложения 5.4) **Pos**-мономорфизмами, состоящая в том, что все трассы образуют класс (Epi **Pos**) ⋂ (RegMono **Pos**)-Init, не была выявлена.

Обозначим через *AOSM* категорию *AO*, порожденную технологией *SM*. Как и в любой формальной технологии над **Set**, аспекты в составе AO-модели сценария представляют собой не что иное, как прообразы меток, приписанных событиям и превращающих его в частично упоря-

доченное мультимножество (pomset) [236]. Поэтому *AOSM* эквивалентна категории **PosEqu**, состоящей из всех частично упорядоченных множеств с отношением эквивалентности и всех их гомоморфизмов (причем функтор, задающий эту эквивалентность, порождает структурируемую скоординированную АО-технологию над *SM*). Метки, множество которых извлекается из помеченного сценария посредством функтора $str : AOSM \to \mathbf{Set}$, могут рассматриваться как «имена» событий, обозначающие обрабатывающие их классы задач. Применение функтора $mod : AOSM \to \mathbf{Pos}$, «забывающего» метки, можно истолковать как выявление структуры времени помеченного сценария [62]. Этот функтор имеет не только левый сопряженный $mod^* : \mathbf{Pos} \to AOSM$, порождающий дискретную разметку (теорема 4.1), но и правый сопряженный mod_*, порождающий сингулярную разметку, как в технологии моделирования данных: он действует на пустое множество как mod^*, а любому непустому частично упорядоченному множеству S сопоставляет аспект $\langle S, !_S : S \to \mathbf{1}\rangle$. Более того, ввиду кополноты категории **Set** и следствия 5.1.8 технология $AO_{mod}(SM)$ скоординирована (несмотря на то, что *SM* таковой не является, так что нельзя воспользоваться следствием 5.1.7: забывающий функтор $|\text{–}| : \mathbf{Pos} \to \mathbf{Set}$ не поднимает копределы диаграммы Λ_2, которая состоит из двух тождественных отображений множеств $\mathbf{2} \hookleftarrow |\mathbf{2}| \hookrightarrow \mathbf{2}^{\mathrm{op}}$ и имеет копредел с одноточечным объектом). В свою очередь, технологии $AO_{asp}(SM)$ и $AO_{int}(SM)$ не скоординированы, поскольку ни функтор *asp*, ни *int* не поднимают копределы *AOSM*-диаграммы $mod^* \circ \Lambda_2$. Кроме того, поскольку категория **Pos** кополна, в силу следствия 5.1.9 все АО-технологии над *SM*, порожденные фундаментальными функторами, структурируемы (функтор mod^{**} был построен нами в пояснении после доказательства предложения 4.21, а функтор asp^{**} переводит произвольный помеченный сценарий $\langle S, l\rangle$ в каноническое отображение множества компонент связности порядка на фактор множества S по отношению эквивалентности, порожденному объединением отношения порядка с $\ker l$). Наконец, в силу следствия 5.1.6 технология *SM* аспектно полна. Отметим, что, в отличие от *DM*, существуют функторы, порождающие АО-технологии над *SM*, но не совпадающие ни с одним из фундаментальных (даже с точностью до эквивалентности категорий). Например, пусть **PosEquC** – полная подкатегория в **PosEqu**, состоящая из всех **PosEqu**-объектов, у которых любой класс эквивалентности является связным частично упорядоченным множеством. Рассмотрим функтор $mc : AOSM \to \mathbf{PosEquC}$, сопоставляющий произвольному помеченному сценарию $\langle S, l\rangle$ **PosEquC**-объект, получающийся из S путем оснащения пересечением отношения эквивалентности, порожденного отношением порядка, с $\ker l$. Непосредственно проверяется, что *mc* порождает АО-технологию над *SM*.

Частные технологии инженерии процессов оперируют с разнообразными подклассами в Ob *AOSM*, позволяя записывать их в специализированных нотациях: графических [31], алгебра-

ических [245], гипертекстовых [99], сетей Петри [135] и др. Обратим внимание, что разметка шагов классами задач в них присутствует изначально, и в этом они отличаются от традиционных технологий моделирования данных. Более того, фактически моделирование поведения систем помеченными сценариями выступает в роли операционной семантики АОП, поскольку, как мы видели в разд.4.3, оно непосредственно отражает концепцию связывания аспектов (weaving). Его частный случай, охватывающий упрощенный вариант АОП, известен как трассовая семантика аспектов [170]. Здесь совет представляет собой обработчик событий базы, образующих срез. Инструмент связывания можно рассматривать как монитор исполнения сценария базы, который при обнаружении точек соединения вызывает исполнение сценариев связываемых с ними советов. Сценарий функционирования монитора описывается связкой – центральным объектом в спецификации связывания. Таким образом, событийное программирование (event-based programming) является одним из способов реализации аспектно-ориентированного подхода [6, 176]. Оно придает системе способность реагировать на изменения в окружении, регистрируемые в виде событий, путем динамической подстройки хода исполнения процессов [193]. Как указывалось в гл.1, такая способность входит в число ключевых показателей качества больших информационно-управляющих систем.

Покажем, что связывание сценариев существует не всегда, хотя *AOSM* и (ко)полна, причем процедура вычисления копредела является частным случаем описанной в доказательстве следствия 5.1.8. Ввиду скоординированности технологии $AO_{mod}(SM)$, основные множества пределов и копределов в *AOSM* вычисляются так же, как в **Pos**. В частности, для вычисления результата связывания произвольной пары *AOSM*-морфизмов $j : B \leftarrow C \rightarrow W : e$ сначала, как и в технологии моделирования данных, выполняется амальгамирование **D***int*-образа схемы. Однако оно в общем случае превращает естественный частичный порядок раздельного объединения множеств $int(B) \amalg (int(C) \times int(W))$ в предпорядок, поэтому выполняется дополнительная факторизация по отношению, обозначаемому через $\lesseqgtr^{\infty}$, которое отождествляет изоморфные элементы этого предпорядка (рассматриваемого как категория). Функтор *str* сохраняет (а значит, в силу предложения 4.10 детерминирует) произведения, однако чтобы он сохранял копредел схемы связывания, связка должна быть совместима с конкуррентностью (concurrency) в том смысле, что она не должна фиксировать порядок вызова аспектов совета, привязываемых к одной точке соединения [6]. Это условие выполняется в технологиях АОП, позволяющих совету иметь только одну точку вызова, например в AspectJ. Формально оно задается следующим образом.

Предложение 5.10. Связывание пары *AOSM*-морфизмов $j : B \leftarrow C \rightarrow W : e$ существует тогда и только тогда, когда для любых $x, y \in C$ условия $j(x) = j(y)$ и $x \le y$ влекут $l(v) = l(e(x))$, где l – разметка сценария W, v – его произвольный элемент, удовлетворяющий условию $e(x) \le v \le e(y)$.

Доказательство. Чтобы множество меток копредела схемы связывания было амальгамой ее **D***str*-образа, необходимо и достаточно, чтобы факторизация по отношению $\leqslant^{\infty}$ не привела к отождествлению различных меток. Это равносильно тому, что для любых $x, y \in C$ таких, что $j(x) = j(y)$ и $x \le y$, отображение $str(\langle 1_C, e\rangle) : str(C) \to str(C) \times str(W)$ устанавливает биекцию между множествами $str(\{z \mid x \le z \le y\})$ и $str(\{(z, v) \mid x \le z \le y \wedge e(x) \le v \le e(y)\})$, т.е. тому, что $|str(\{v \mid e(x) \le v \le e(y)\})| = 1$. □

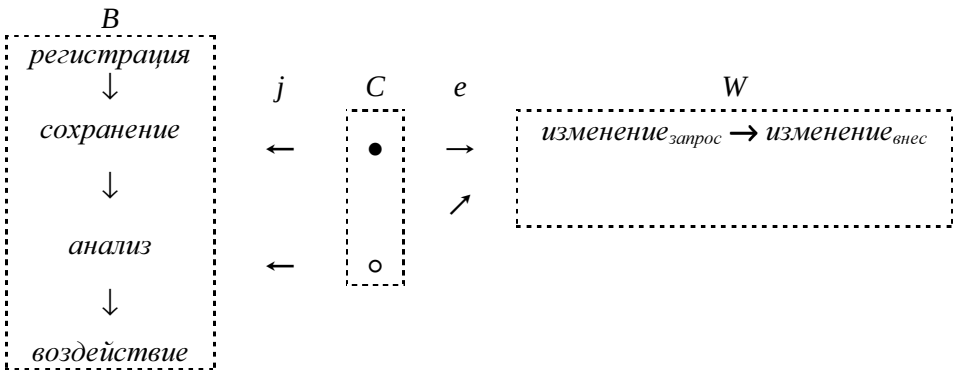
Рассмотрим подробнее связывание сценариев при проектировании информационно-управляющих систем с применением событийной модели [70]. Крупноблочный базовый сценарий B основного процесса функционирования системы состоит в повторяющемся исполнении следующей цепи классов задач по обработке событий, возникающих на объекте управления:

$$\textit{регистрация} \to \textit{сохранение} \to \textit{анализ} \to \textit{воздействие}.$$

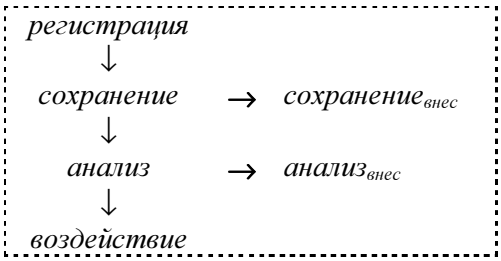
Инфраструктурные аспекты, обеспечивающие его корректную работу, например информационная модель объекта и средства защиты информации, комплексируются с ним путем связывания, препятствуя разделению сценария на модули. Управление большими объектами вынуждает распределять исполнение основного цикла обработки событий между несколькими аппаратными узлами, но для этого необходимо реплицировать инфраструктуру между ними. Такая репликация вызывает основные трудности при проектировании и эксплуатации распределенных информационно-управляющих систем по сравнению с локальными (АСУ ТП), поэтому привлечение технологий АОП способствует снижению затрат на создание больших систем. Рассмотрим ведение информационной модели (паспорта) объекта – процесс, приведенный в разд.1.3 как характерный пример области применения аспектно-ориентированного подхода. Необходимость изменить паспорт объекта может возникнуть при сохранении события в базу данных системы, если оно сигнализирует о фактических изменениях его структуры, например, о замене прибора. Другие фрагменты паспорта могут измениться по результатам анализа, если выявлено его несоответствие объекту: например, зарегистрировано потребление энергии единицей оборудования, не связанной в паспорте ни с каким питающим центром. Поведение аспекта изменения паспорта W можно упрощенно описать помеченным сценарием из двух последовательных событий, относящихся к одному классу задач:

$$\textit{изменение}_{\textit{запрос}} \to \textit{изменение}_{\textit{внес}}.$$

Его связка с базовым сценарием имеет вид дискретного двухэлементного сценария $\mathbf{1} \coprod \mathbf{1}$, где $\mathbf{1}$ – терминальный *AOSM*-объект: морфизм j представляет собой его биекцию на подмножество $\{\textit{сохранение}, \textit{анализ}\} \subseteq B$, а e – постоянное отображение на элемент $\textit{изменение}_{\textit{запрос}} \in W$. Получается спецификация связывания следующего вида:



Как легко проверить, связывание существует и порождает следующий помеченный сценарий:



где индексом *внес* снабжены события соответствующих классов задач, фиксирующие внесение изменений в паспорт объекта. Здесь наглядно видно, как процедуры паспортизации ассимилируются основным процессом функционирования информационно-управляющей системы: в соответствие с утверждением (ii) предложения 4.9, метка аспекта изменения паспорта в этом сценарии отсутствует – она «сливается» с метками точек соединения.

В то же время каждый акт изменения паспорта приобретает пометку инициировавшей его задачей, которую можно сохранить вместе с изменяемыми данными (при условии, что хранилище данных паспорта спроектировано в соответствие с подходом, изложенным в разд.5.2). В реальных системах вышеописанная процедура связывания позволяет рассеивать задачу ведения паспорта по сотням процессов-источников данных, с обеспечением трассирования каждого элемента к задачам, вызвавшим его изменение. Это позволяет достичь практически полной актуальности и достоверности паспорта. Обобщая этот пример, покажем, что между технологиями аспектно-ориентированного моделирования данных и процессов имеется взаимосвязь, обуславливающая построение совместных моделей типа изображенной в разд.1.3. Формально она имеет вид **ARCH**-морфизма из АО-технологии над *SM* в ***AODM***, который задает способ переноса разметки сценариев на массивы данных, с которыми они оперируют, естественный относительно сборки систем, выделения интерфейсов, и трансформаций.

Предложение 5.11. Функтор $asp : AOSM \to \mathbf{Surj}$ индуцирует **ARCH**-морфизм, действующий из любой АО-технологии над *SM* в АО-технологию моделирования данных ***AODM***.

Доказательство. Имеется **ARCH**-морфизм $\langle asp, 1_{\mathbf{Set}}, asp(-^{\mathrm{op}})^{\mathrm{op}}\rangle : AO_{int}(SM) \to \boldsymbol{AODM}$ (его вид определяется следствиями 5.1.4 и 5.1.3: это ***aom***-образ **SPLL**-морфизма $\langle |-|, 1_{\mathbf{Set}}\rangle : \langle SS, \mathrm{Surj}\rangle \to \langle DS, \mathrm{Surj}\rangle$). Существует композиция этого **ARCH**-морфизма с морфизмом $aoint_{ai} : AO_{ai}(SM) \to AO_{int}(SM)$ из предложения 4.25. □

Отметим, что выделение аспектной структуры естественно относительно этого морфизма: функтор $str : AOSM \to \mathbf{Set}$ совпадает с $(codom \circ isurj) \circ asp$.

Результат связывания аспекта ведения паспорта с основным циклом обработки событий обладает еще одним показательным свойством: он не допускает разделения ответственности. Действительно, напомним, что аспектное ядро в *AOSM* (см. определение 4.14) состоит из всех помеченных сценариев, в которых разметка разбивает свою область на хорошо упорядоченную совокупность прообразов точек. Сценарии *B* и *W* из нашего примера удовлетворяют этому условию, однако результат их связывания – нет. Показательным примером немодуляризируемого сценария является также интерливинг (interleaving) – сценарий *Il* вида $\bullet \to \circ \to \bullet \to \circ \to \ldots$, который мы построили в разд.4.4 для доказательства отсутствия у технологии *SM* полного AO-расширения. Он характерен для распределенных систем, аспекты которых взаимодействуют в режиме клиент-сервер. Его нельзя собрать из двух составляющих его аспектов путем модульной компоновки, поскольку копредел любой конфигурации, составленной из них, изоморфен одному из них либо их прямой сумме. Нельзя получить этот сценарий и путем связывания двух аспектов, т.к. оно порождает аспект (утверждение (ii) предложения 4.9). Эти факты иллюстрируют трудности, возникающие при создании даже простой клиент-серверной распределенной системы.

Ясно, что любой модуляризируемый сценарий допускает разделение ответственности. Отсюда непосредственно вытекает следующее утверждение.

Предложение 5.12. Технология *SM* аспектно отражает некоторый оптимизируемый класс разбиений тогда и только тогда, когда он состоит из неразрушающих хорошо упорядоченных разбиений. □

Примером применения предложения 5.12 служит класс *SMI* всех неразрушающих хорошо упорядоченных разбиений. Зафиксируем произвольное непустое частично упорядоченное множество *S*. Инициальным объектом в категории $SMI \downarrow S$ служит дискретное разбиение (т.е. разбиение на одноэлементные подмножества), а терминальным – однокомпонентное неразрушающее «разбиение» (т.е. изоморфизм, см. предложение 2.16). Также *SM* аспектно отражает класс *SMS* всех сумм, так что любой непомеченный сценарий можно оптимально разметить взаимно независимыми (параллельными) аспектами. Инициальным объектом в $SMS \downarrow S$ является разбиение на связные компоненты, состоящее из всех классов отношения эквивалентности,

порожденного отношением порядка. Терминальным объектом в $SMS \downarrow S$ служит однокомпонентное разбиение.

Еще одним важным примером служит класс *SML* всех хорошо упорядоченных разбиений частично упорядоченных множеств на линейно упорядоченные подмножества. Дело в том, что разделение ответственности на линейно упорядоченные подаспекты описывает акт компоновки аспектно-ориентированной системы из модуляризированных аспектов, каждый из которых исполняется строго последовательно. Оптимизируемость совокупности всех таких разделений оправдывает создание аспектно-ориентированных расширений традиционных языков программирования, позволяющих создавать только последовательно исполняемые программы.

Теорема 5.2. Технология *SM* аспектно отражает класс *SML*. □

Для доказательства этой теоремы ввиду предложения 5.12 достаточно показать, что класс *SML* оптимизируем, поскольку любой **Pos**-мономорфизм с линейно упорядоченной областью является регулярным (это свойство может служить категорной характеризацией линейно упорядоченных множеств). Действительно, инициальным объектом в $SML \downarrow S$ является дискретное разбиение. Для построения терминального объекта в нем нам необходима конструкция эквисравнимости, введенная в [62]. Будем называть (симметричным замкнутым) *интервалом* множество $[t, t']$ элементов S, находящихся между точками $t, t' \in S$, включая сами эти точки:

$$[t, t'] = \{t, t'\} \bigcup \{t'' \in S \mid (t \le t'' \le t') \vee (t' \le t'' \le t)\}.$$

Будем называть подмножество в S *выпуклым*, если оно содержит интервал, заключенный между любыми двумя его точками. Очевидным примером выпуклого множества служит произвольный интервал. Другим примером является компонента любого хорошо упорядоченного неразрушающего разбиения: если точки t, t' принадлежат одной компоненте, то условия $t \le t''$ и $t'' \le t'$ могут выполняться только для точки t'', принадлежащей этой же компоненте.

Определение 5.3. Произвольные точки $t, t' \in S$ называются:

- *сравнимыми* (обозначается $t \sim t'$), если $t \le t'$ либо $t' \le t$;
- *эквисравнимыми* (обозначается $t \approx t'$), если для любого $t'' \in S$ условие $t \sim t''$ равносильно $t' \sim t''$;
- *интервально эквисравнимыми* (обозначается $t \,[\approx]\, t'$), если любые две точки из интервала $[t, t']$ эквисравнимы. □

Лемма 5.1. Отношение эквисравнимости является отношением эквивалентности, содержащимся (как множество пар) в сравнимости. □

Лемма 5.2. Интервальная эквисравнимость является отношением эквивалентности, содержащимся в эквисравнимости.

Доказательство. Доказательства заслуживает транзитивность отношения интервальной эквисравнимости. Если t [≈] t' и t' [≈] t'', то любая точка из интервала [t, t''] сравнима с t', поэтому она попадает либо в интервал [t, t'], либо в [t', t'']. В любом из этих случаев она эквисравнима с t', поэтому в силу леммы 5.1 любые две точки из интервала [t, t''] эквисравнимы. □

Лемма 5.3. Разбиение множества S на классы эквивалентности отношения интервальной эквисравнимости хорошо упорядочено.

Доказательство. Для несравнимых точек несравнимость их классов интервальной эквисравнимости непосредственно следует из определения эквисравнимости; остается показать, что если одна из интервально неэквисравнимых точек меньше другой, то ее класс интервальной эквисравнимости также меньше класса интервальной эквисравнимости другой, т.е. проверить выполнение следующего соотношения для любых t, $t' \in S$:

$$(t \le t' \wedge \neg\, t\ [\approx]\ t') \Rightarrow \forall t''\ (t\ [\approx]\ t'' \Rightarrow t'' \le t').$$

Действительно, поскольку $t \sim t'$, имеем $t'' \sim t'$ ввиду эквисравнимости точек t и t'' (лемма 5.2). Но если $t' \le t''$, то $t' \in [t, t'']$, поэтому обязательно t [≈] t', что противоречит выбору точки t'. Поэтому $t'' \le t'$. □

Лемма 5.4. Любые две точки из одной компоненты произвольного разбиения множества S, принадлежащего классу *SML*, интервально эквисравнимы.

Доказательство. Выберем произвольно: разбиение $\pi \in SML \downarrow S$, область P некоторого ребра коконуса π, точки t, $t' \in P$ и p, $p' \in [t, t']$. Ввиду выпуклости P имеем p, $p' \in P$. Для произвольной точки $p'' \in P$ имеем $p \sim p''$ и $p' \sim p''$ ввиду линейной упорядоченности P; в частности, $p \sim p'$. С учетом этого, для произвольной точки $p'' \in S \setminus P$ условие $p \le p''$ равносильно $p' \le p''$, а $p'' \le p$ равносильно $p'' \le p'$, ввиду хорошей упорядоченности разбиения π. Таким образом, $p \approx p'$, так что t [≈] t'. □

В силу этих лемм, разбиение множества S на классы интервальной эквисравнимости представляет собой терминальный объект в $SML \downarrow S$. Доказательство теоремы 5.2 завершено.

5.4 Процессные модели архитектуры

Выявление отдельных сценариев поведения системы и выделение аспектов в них является начальной стадией ее анализа. На следующем шаге определяются и формализуются правила их порождения, в результате чего формируется процессная модель архитектуры системы. Широко применяемым способом ее описания является указание состава состояний системы и переходов между ними. Формальные системы состояний и переходов допускают автоматическую трансформацию в программный код [188], что позволяет замкнуть цикл автоматизации процес-

сов, управляемый моделями, в соответствие с концепцией MDE. В качестве примера рассмотрим помеченные структуры событий [246], часто применяемые для моделирования реактивных систем, в том числе информационно-управляющих. В них состояние представляет собой наполнение журнала событий, а переход происходит по факту регистрации каждого нового события. Семантика событий задается их метками. Ограничения, выделяющие допустимые сценарии, сводятся к причинно-следственному упорядочению событий и указанию конфликтов – пар взаимоисключающих событий, которые не могут одновременно входить ни в один сценарий. Примером конфликта служит реакция оператора на рекомендацию информационно-управляющей системы выполнить некоторое управляющее воздействие на объект: оператор может либо одобрить выполнение воздействия, либо отклонить его, но не может дать оба ответа.

Определение 5.4 [246]. Помеченной *структурой событий* называется четверка $\langle E, \leq, \#, l \rangle$, где:

- E – множество, элементы которого называются событиями;
- $\leq$ – частичный порядок на E такой, что любой начальный отрезок $[\{e\}] = \{e' \mid e' \leq e\}$ является конечным множеством;
- $\#$ – иррефлексивное симметричное бинарное отношение на E, называемое отношением конфликта, такое, что $e \# e' \leq e''$ влечет $e \# e''$;
- l – отображение E во множество, элементы которого называются метками событий.

Множество $C \subseteq E$ называется *конфигурацией*, если выполнены следующие условия:

(i) $(e \in C \wedge e' \leq e) \Rightarrow e' \in C$ (замкнутость слева).

(ii) $(e \in C \wedge e' \# e) \Rightarrow e' \notin C$ (бесконфликтность).

Событие e называется *разрешенным* в конфигурации C (обозначается $C \vdash e$), если $e \notin C$ и $C \bigcup \{e\}$ – конфигурация. Структура событий называется *детерминированной*, если условия $C \vdash e$, $C \vdash e'$ и $l(e) = l(e')$ влекут $e = e'$. □

Структуру событий можно формально превратить в *AOSM*-объект путем «забывания» отношения конфликта. Однако такое представление является малосодержательным. Согласно [77], рассматривать как *AOSM*-объекты следует совокупности конфигураций, представляющих собой состояния моделируемой системы. В частности, рассмотрим множество FC_{ES} всех конечных конфигураций структуры *ES* как *AOSM*-объект следующего вида. Отношением порядка на нем служит теоретико-множественное включение, а меткой конфигурации C – $l(e)$, если $C = [\{e\}]$ для некоторого $e \in E$, и дополнительная метка $\perp \notin l(E)$ в противном случае. Ясно, что структура событий *ES* определяется объектом FC_{ES} однозначно с точностью до переименования событий. Действительно, конфигурация вида $[\{e\}]$ взаимно однозначно определяет событие e,

включение $[\{e\}] \subseteq [\{e'\}]$ эквивалентно соотношению $e \le e'$, а отсутствие конфигурации, содержащей как $[\{e\}]$, так и $[\{e'\}]$, эквивалентно конфликту $e \# e'$. В свою очередь, любая конфигурация вида $[\{e\}]$ может быть идентифицирована в частично упорядоченном множестве $mod(FC_{ES})$, которое является нижней полурешеткой, как неприводимый элемент (т.е. имеющий максимального собственного предшественника). Этот факт позволяет строить структуры событий и в случаях, когда для моделирования состояний используются какие-либо формальные конструкции, отличные от конфигураций. Нужно лишь, чтобы они были определенным образом упорядочены отношением предшествования.

Введем обозначения: $[X]$ – идеал, порожденный подмножеством X заданной нижней полурешетки; $Ev(X)$ – множество неприводимых элементов, содержащихся в $[X]$; $MaxEv(X) = \{y \mid y \in Ev(X) \wedge (\forall z\,(z \in Ev(X) \wedge y \in [\{z\}]) \Rightarrow z = y)\}$. Ясно, что если x неприводим, то $MaxEv(\{x\}) = \{x\}$.

Определение 5.5. *AOSM*-объект $AS = \langle S, \le, l\rangle$ называется *потенциалом*, если выполняются следующие условия:

(i) $\langle S, \le\rangle$ является полной нижней полурешеткой;

(ii) идеал $[\{x\}]$ конечен для любого $x \in S$;

(iii) если $x \notin Ev(S)$, то $(l(x) = l(y)) \Leftrightarrow (y \notin Ev(S))$;

(iv) для любого конечного множества $E \subseteq Ev(S)$, если для любых $e, e' \in E$ существует x такой, что $\{e, e'\} \subseteq [\{x\}]$, то существует y такой, что $Ev(\{y\}) = Ev(E)$.

При этом четверка $ES(A) = \langle Ev(S), \le \restriction_{Ev(S)}, \#, l \restriction_{Ev(S)}\rangle$, где $e \# e'$ означает, что не существует такого $x \in S$, что $\{e, e'\} \subseteq [\{x\}]$, называется *структурой* потенциала S. □

Предложение 5.13. Структура $ES(AS)$ любого потенциала AS является помеченной структурой событий, причем $FC_{ES(AS)} \cong AS$. Обратно, FC_{ES} является потенциалом для любой структуры событий ES.

Доказательство. Неочевидным является только существование изоморфизма между $FC_{ES(AS)}$ и AS. Чтобы задать его, докажем, что элемент y из условия (iv) определения 5.5 однозначно определяется множеством $MaxEv(E)$. Воспользуемся индукцией по $|MaxEv(E)|$. Если $MaxEv(E) = \varnothing$, то в качестве y можно выбрать только нуль нижней полурешетки $\langle S, \le\rangle$. Допустим, что y однозначно определяется по $MaxEv(E)$ при условии $|MaxEv(E)| < n$, и существует подходящее E с $|MaxEv(E)| = n$. Пусть $Y = \{y \mid Ev(\{y\}) = Ev(E)\}$, $y_0 = \min Y$. Предположим, что существует $y_1 \in Y \setminus \{y_0\}$. Поскольку y_1 приводим, он должен иметь предшественника u, несравнимого с y_0. Но если $MaxEv(\{u\}) \subset MaxEv(E)$, то $u \in [\{y_0\}]$ по предположению индукции, поэтому $MaxEv(\{u\}) \setminus E \ne \varnothing$. Таким образом, $MaxEv(\{y_1\}) \setminus E \ne \varnothing$, что противоречит выбору y_1. □

В контексте нашего подхода интерес представляет ситуация, когда состояния описываются аспектно-расширенными моделями некоторой формальной технологии проектирования $AR = \langle c\text{-}DESC, Conf, sig, r\text{-}DESC\rangle$. Потребуем, чтобы переходы подчинялись системным законам этой технологии, т.е. чтобы отношение предшествования можно было задать в виде подкатегории $FC \subseteq c\text{-}DESC$, имеющей форму частичного порядка. Тогда оно опускается на уровень интерфейсов путем сужения на дискретные *c-DESC*-объекты: класс морфизмов $\mathrm{FS} = sig(\mathrm{Mor}\ FC \bigcap sig^*(\mathrm{Mor}\ SIG))$ задает частичный порядок в *SIG*. В свою очередь, частичный порядок в категории АО-моделей *AO*, порожденной технологией *AR*, задается классом морфизмов $\{\langle f, b\rangle \mid f \in \mathrm{Mor}\ FC, b \in \mathrm{FS}\}$.

Частный случай, когда потенциалы состоят из конфигураций, рассматриваемых как *AOSM*-объекты, проанализирован в [246]. Здесь в качестве отношения предшествования в категории **Pos** естественным образом выбирается теоретико-множественное включение префикса – подмножества $Y \subseteq X$ такого, что $\neg x \leq y$ для любых $x \in X \setminus Y$ и $y \in Y$. В категории **Set** префиксами множества являются в точности все его подмножества. Использование префиксов при конструировании потенциалов позволяет упростить формулировку условия (iv) определения 5.5. Кроме того, удается сформулировать критерии, которым должны удовлетворять потенциалы детерминированных структур событий. Ключевую роль в них играет линейная упорядоченность аспектов (ср. теорему 5.2).

Введем обозначения: $Pref(X)$ – класс всех включений собственных (т.е. отличных от X) префиксов объекта X; *FPref* – подкатегория в *AOSM*, состоящая из всех *AOSM*-объектов с конечным числом префиксов и всех включений их префиксов. В качестве метки $lme(X)$ *FPref*-объекта $X \in Ev(FPref)$ выбирается единственный **Set**-морфизм $e : \mathbf{1} \to int(X)$ такой, что:

- $int(m) \circ v \neq e$ для любого **Set**-морфизма v, где m – включение максимального собственного префикса объекта X;
- $l \circ e$, где l – аспектная разметка объекта X, является **Set**-включением.

Если же $X \notin Ev(FPref)$, то полагаем $lme(X) = 1_\varnothing$. Будем называть *FPref*-объект X *аспектно линейным*, если для любого *AOSM*-морфизма $m : A \to X$ такого, что $mod(m) \in \mathrm{RegMono}\ \mathbf{Pos}$ и $str(A) = \mathbf{1}$, выполняется $A \in Ev(FPref)$ (это условие эквивалентно тому, что каждый аспект сценария X линейно упорядочен).

Предложение 5.14. Категория *FC* вместе с отображением $l : \mathrm{Ob}\ FC \to L$ является объектом конфигураций FC_{ES} для некоторой структуры событий *ES* тогда и только тогда, когда выполняются условия:

(i) *FC* является подкатегорией в *FPref* (финитность);

(ii) если $X \in \mathrm{Ob}\ FC$, то $Pref(X) \subseteq \mathrm{Mor}\ FC$ (замкнутость слева);

(iii) $L = lme(\mathrm{Ob}\ FC)$, $l(X) = lme(X)$ для любого FC-объекта X (разметка);

(iv) если $X \in \mathrm{Ob}\ FPref$, $Pref(X) \subseteq \mathrm{Mor}\ FC$ и $|MaxEv(X)| > 2$, то $X \in \mathrm{Ob}\ FC$ (когерентность).

Она определяет детерминированную структуру событий тогда и только тогда, когда дополнительно выполняются следующие условия:

(v) любой FC-объект аспектно линеен;

(vi) если для некоторых $X, Y \in \mathrm{Ob}\ FC$ существует пара $AOSM$-морфизмов $f : X \leftarrow Z \rightarrow Y : g$ такая, что $int(f)$ и $int(g)$ являются изоморфизмами, $str(f) = str(g) = 1_{str(Z)}$, и Z аспектно линеен, то $X = Y$. □

Первое утверждение вытекает из предложения 5.13, справедливость второго установлена в [246]. Отметим, что там оно сформулировано в терминах внутренней структуры $AOSM$-объектов, в то время как наши формулировки имеют чисто теоретико-категорный вид. Это открывает возможности для обобщения процедуры построения структур событий на другие аспектно-ориентированные технологии проектирования информационно-управляющих систем (отличные от АО-технологий над SM). Отметим также, что и класс всех детерминированных структур событий, и класс всех их потенциалов рассматриваются в [246] как категории, причем морфизмы в них определяются так, что преобразование $ES(-)$, переводящее потенциал в структуру событий, расширяется до функтора, задающего эквивалентность этих категорий.

В заключение раздела отметим, что структуры событий являются лишь одним из возможных представлений процессной модели архитектуры. Например, их можно преобразовать в системы переходов специального вида (transition systems with independence) [245] посредством функтора, имеющего правый сопряженный с тождественной единицей. Для анализа систем переходов применяются, в частности, методы теории гомологий [128].

5.5 Модели предметной области ТЭК

Показательной областью применения технологий аспектно-ориентированного моделирования является проектирование информационно-управляющих систем для объектов топливно-энергетического комплекса (ТЭК) [63]. Как указывалось в разд.1.1, здесь традиционно высока потребность в системах большого и сверхбольшого масштаба. Тенденция к укрупнению объектов ТЭК диктуется фундаментальными физико-техническими соображениями: главные энергоресурсы (электрическая и тепловая энергия) не могут накапливаться для хранения в промышленных масштабах, поэтому в каждый момент времени подлежат потреблению в точности в том объеме, в каком они произведены. Чтобы избежать потерь и перебоев энергоснабжения, процессы производителей и потребителей энергии должны идти синхронно. Ключевую роль при-

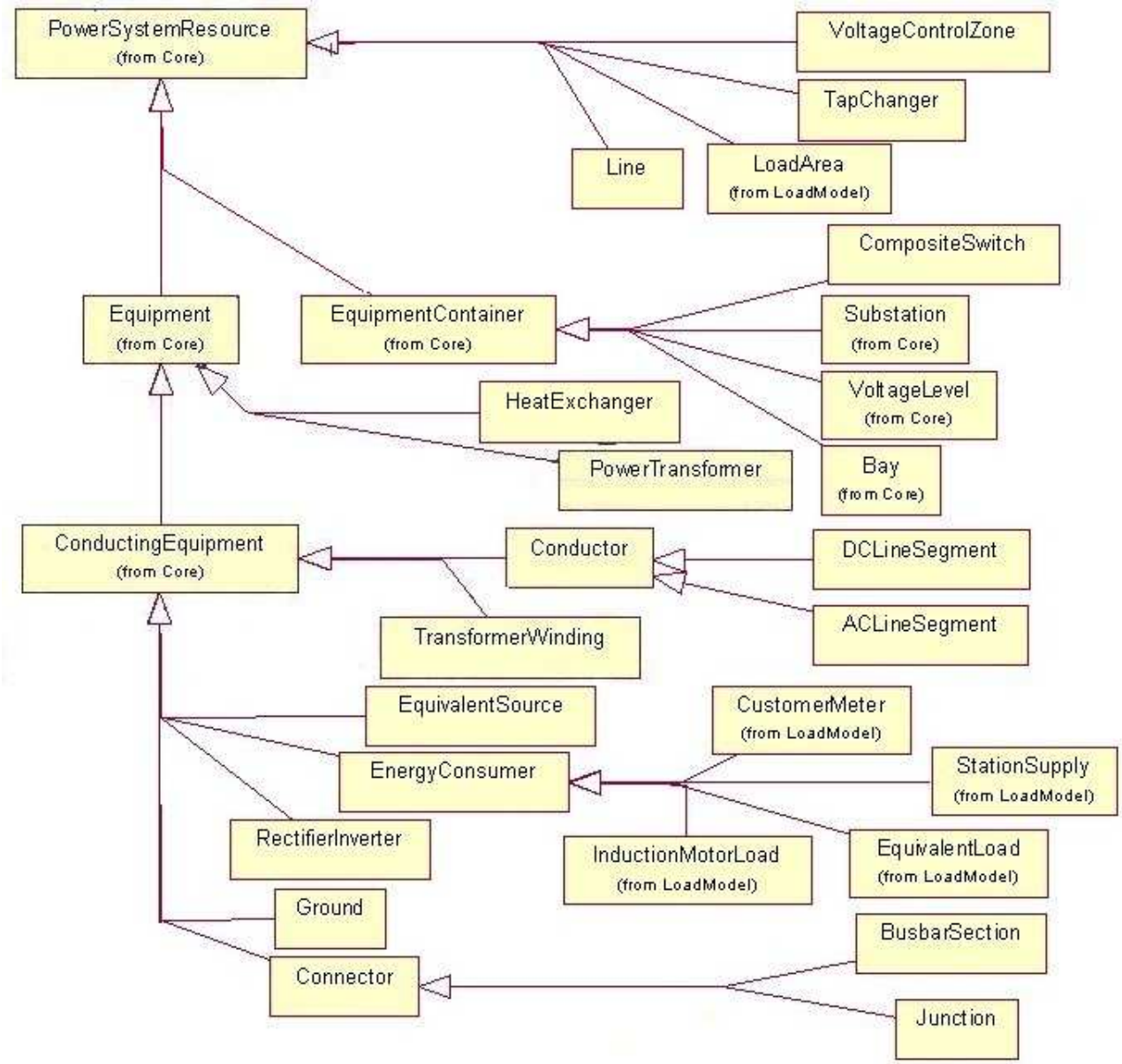
обретает оперативно-диспетчерское управление – специфический сквозной непрерывный управляющий технологический процесс, охватывающий все энергоустановки, между которыми происходит обмен энергией. Субъекты диспетчерского управления входят в число основных заказчиков и потребителей средств комплексной автоматизации в ТЭК [11]. Для них актуальны все требования качества к большим информационно-управляющим системам, перечисленные в разд.1.2.

Приведем обзор анализа предметной области ТЭК: кратко опишем основные части пространства задач, выделенные в разд.1.4 (см. Таблица 4). Поскольку предметом автоматизации здесь является организованная деятельность больших групп людей, первичными источниками информации для анализа предметной области, в том числе понятий и отношений онтологии, являются нормативно-правовые акты – законы, постановления, приказы органов государственного управления. В России к числу основополагающих в сфере ТЭК относятся следующие федеральные законы:

- №35-ФЗ «Об электроэнергетике»;
- №190-ФЗ «О теплоснабжении»;
- №69-ФЗ «О газоснабжении в Российской Федерации»;
- №2395-1 «О недрах»;
- №81-ФЗ «О государственном регулировании в области добычи и использования угля, об особенностях социальной защиты работников организаций угольной промышленности»;
- №170-ФЗ «Об использовании атомной энергии»;
- №117-ФЗ «О безопасности гидротехнических сооружений»;
- №102-ФЗ «Об обеспечении единства измерений»;
- №147-ФЗ «О естественных монополиях»;
- №210-ФЗ «Об основах регулирования тарифов организаций коммунального комплекса»;
- №256-ФЗ «О безопасности объектов топливно-энергетического комплекса»;
- №7-ФЗ «Об охране окружающей среды»;
- №261-ФЗ «Об энергосбережении и о повышении энергетической эффективности и о внесении изменений в отдельные законодательные акты Российской Федерации»;
- №382-ФЗ «О государственной информационной системе топливно-энергетического комплекса».

Законы регулируют преимущественно отношения между субъектами ТЭК, для автоматизации которых традиционно привлекаются средства уровня АСУП. Эти отношения неразрывно

связаны с деятельностью на производственно-технологических объектах ТЭК (энергоустановках), которая традиционно автоматизируется средствами уровня АСУТП. Она регламентируется многочисленными нормативно-техническими документами, такими как стандарты (ГОСТ), руководящие документы (РД), технические условия (ТУ), стандарты организаций (СТО) и др. Во многих из них приводятся термины технологического характера вместе с определениями, из которых извлекаются онтологические отношения. Широкий набор понятий и отношений в сфере электроэнергетики, подготовленный к применению при разработке автоматизированных систем, описан в международном стандарте IEC 61970-301 «Energy management system application program interface (EMS-API) – Part 301: Common information model (CIM) base» [197]. В качестве нотации в нем используется язык объектно-ориентированного моделирования UML, и существует его перевод на язык описания онтологий OWL [256]. Приведем фрагмент CIM, охватывающий некоторые виды электроэнергетического оборудования.



Видно, что понятия организованы в глубокую классификационную иерархию «абстрактное–конкретное» (наследование). Это позволяет модели CIM «поглощать» в себя онтологии небольших фрагментов предметной области, составленные согласно частным нормативным документам. Например, понятие средства измерения из онтологии метрологии, приведенной в разд.1.2, становится частным случаем технологического ресурса энергосистемы (`PowerSystemResource`), наряду с оборудованием, вычислительной техникой, персоналом. Понятие измерительного канала, следуя подходу CIM, объединяется с точками поставки ресурсов, подключения коммутационных аппаратов и т.д. в общую иерархию, базовым понятием которой является точка локализации управляющих воздействий [9].

В то же время в CIM отсутствуют понятия, необходимые для ведения истории изменения состава и характеристик оборудования энергоустановок [9]. Трудность ведения истории состоит в том, что его нельзя свести к указанию дат начала и окончания периода актуальности каждого значения каждого информационного элемента. Как указывалось в разд.1.3, необходимо сохранить связь с процессами, вызвавшими изменения значений. Для этого в онтологию вводится понятие конфигурации энергоустановки – это стабильное состояние комплекса оборудования, изменение которого означает переход к следующему этапу ее жизненного цикла (реконструкцию, перемаркировку и др.). Паспорт энергоустановки состоит из последовательности исторических конфигураций, одной актуальной и произвольного множества проектируемых. Характеристики (свойства и отношения) единиц оборудования разделяются на несколько категорий, предполагающих разные режимы ведения истории:

- характеристики типа/марки, установленные в проектной документации завода-изготовителя (например, коэффициент трансформации трансформатора);
- характеристики экземпляра, изменяющиеся только при переходе к новой конфигурации (например, длина линии электропередачи, которая может измениться при реконструкции энергоустановки);
- эксплуатационные характеристики, способные изменяться в результате работ по техническому обслуживанию и ремонту (например, коэффициент потерь тепла в теплотрассе);
- режимные характеристики, изменяющиеся при изменении режима объекта (например, уставка температуры теплоносителя на выходе котла);
- оперативные характеристики, непрерывно изменяющиеся в ходе функционирования объекта (например, показания измерительного прибора).

Еще одной трудностью применения модели CIM является недостаточная проработка понятий уровня АСУП. Часть этих понятий введена в пакете стандартов IEC 61968, однако он не в

полной мере отражает специфику таких сложных процессов управления, как тарифное регулирование или слияние/поглощение субъектов. Более перспективным представляется применение онтологий, построенных на базе стандартов электронного обмена деловой информацией типа ebXML [210]. Здесь возникают трудности, вызванные концептуальным разрывом между уровнями АСУТП и АСУП, о котором мы упоминали в разд.1.1: например, экземпляры технологического понятия «Единица оборудования» находятся в запутанных многозначных отношениях с экземплярами бухгалтерского понятия «Основное средство». В онтологию вводятся правила соотнесения понятий, вызываемые при переходе потоков исполнения сквозных процессов между двумя уровнями.

В корпус нормативно-технической документации ТЭК входит международный стандарт, проводящий процессный подход к управлению – это ISO 50001:2011 «Energy Management Systems – Requirement with Guidance for Use» (в России он принят под названием ГОСТ Р 50001-2012 «Системы энергетического менеджмента. Требования и руководства по применению» [34]). Он предписывает постоянно выполнять стандартный управленческий цикл PDCA (Plan – Do – Check – Act, планирование – выполнение – проверка – управление (исправление)) [40, с.100] в целях повышения энергетической результативности – полезного эффекта от использования энергоресурсов. Модель такого цикла энергетического менеджмента выглядит следующим образом [34, с.VII].

Наиболее значимые процессы, эффективность которых прямо или косвенно зависит от энергетической результативности, перечислены в таблице (Таблица 7), в привязке к видам владельцев – субъектов ТЭК.

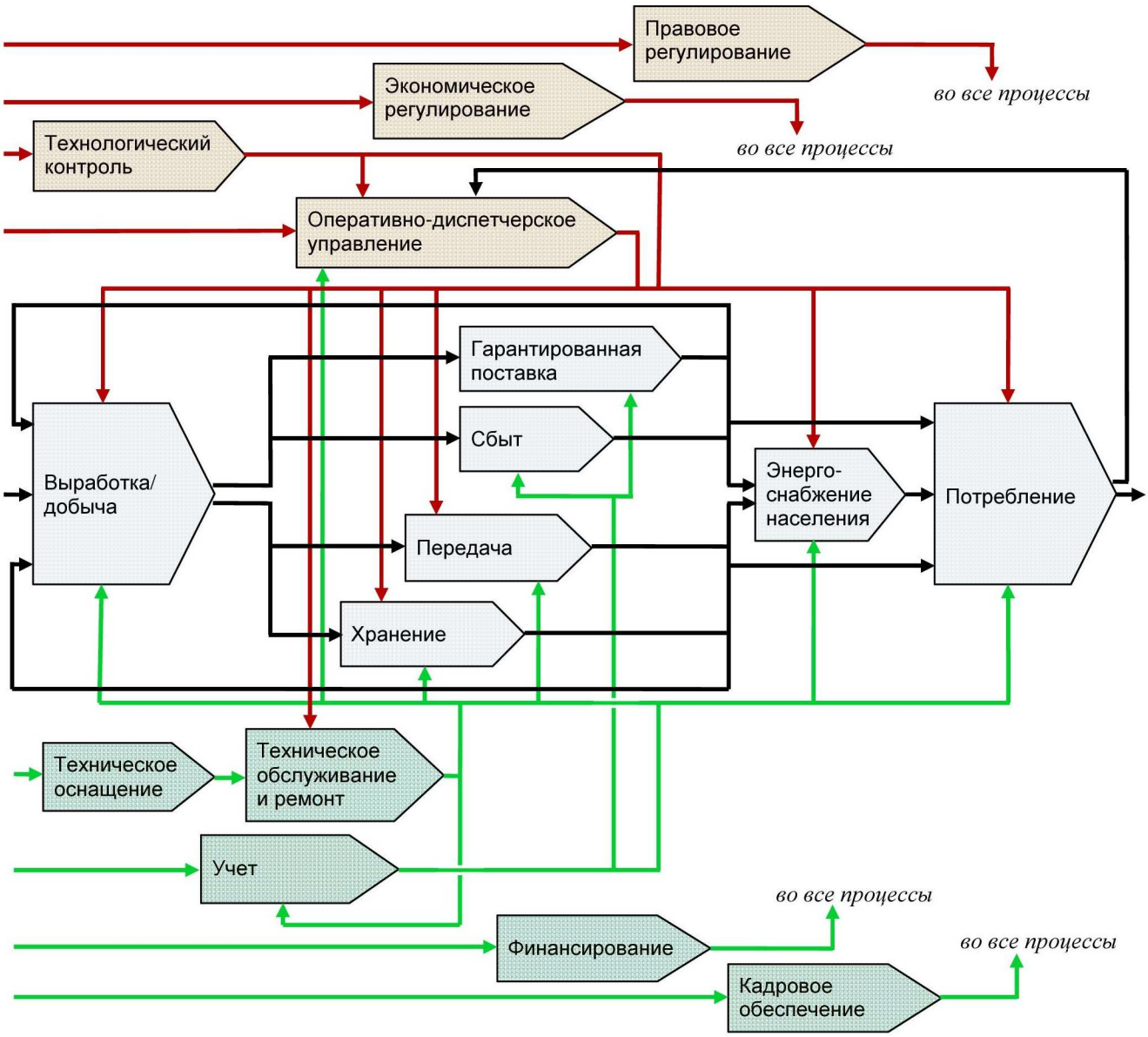
Таблица 7

Процессы	**Владельцы**	**Критерии эффективности**	**Ограничения** *(кроме стоимостных)*
Основные процессы			
Потребление энергии	Потребители	Снижение затрат на энергоресурсы	Потребность собственных нужд в энергии, диспетчерские указания
Сбыт энергии потребителям и покупка у производителей	Сбытовые организации	Повышение прибыли от поставки энергоресурсов	Требования потребителей
Гарантированное энергообеспечение потребителей	Гарантирующие поставщики	Увеличение объемов поставки энергоресурсов	Обязанность обслужить каждого потребителя в своей зоне деятельности
Энергоснабжение населения	Исполнители коммунальных услуг	Повышение качества энергоснабжения населения	Диспетчерские указания
Передача энергии и присоединение энергоустановок к сетям	Сетевые организации	Увеличение объема полезного отпуска энергоресурсов	Диспетчерские указания
Хранение топлива	Операторы топливохранилищ	Повышение полезной загрузки топливохранилищ	Диспетчерские указания
Выработка или добыча энергоресурсов	Производители	Повышение прибыли от продажи энергоресурсов	Диспетчерские указания
Обеспечивающие процессы			
Техническое обслуживание и ремонт оборудования энергоустановок	Сервисные службы и организации	Снижение затрат на техническое обслуживание и ремонт	Диспетчерские указания
Техническое оснащение энергоустановок	Поставщики оборудования, приборов, программных средств	Повышение прибыли от продажи технических средств	Потребность операторов энергоустановок в технических средствах
Учет энергоресурсов	Операторы учета энергоресурсов	Повышение точности учета энергоресурсов	Метрологические характеристики методов учета
Финансирование деятельности субъектов энергетики	Инвестиционные и финансовые организации	Повышение доходности вложенного капитала	Потребность субъектов в финансовых средствах, риски невозврата инвестиций
Кадровое обеспечение энергетических процессов	Кадровые службы и саморегулируемые организации	Снижение затрат на персонал	Нормы квалификации персонала
Управляющие процессы			

Оперативно-диспетчерское управление энергоустановками	Оперативно-диспетчерские службы	Повышение надежности и качества поставки энергоресурсов	Объемы отпуска энергоресурсов, технические характеристики оборудования
Нормирование и технологический контроль энергоустановок	Органы технологического контроля и регулирования	Повышение безопасности энергоустановок	Полномочия по вмешательству в деятельность субъектов
Экономическое регулирование деятельности субъектов энергетики	Органы тарифного регулирования	Повышение доступности продукции регулируемой деятельности	Рентабельность регулируемой деятельности
Правовое регулирование и программно-целевое управление	Органы государственной власти	Устойчивое развитие территорий и отраслей	Социальные гарантии, экологические нормы

Подчеркнем, что в дополнение к ограничениям, указанным в последней колонке, на каждый процесс накладываются стоимостные ограничения. В конкурентных секторах, в которых деятельность направлена на максимизацию прибыли, они возникают вследствие естественной балансировки спроса и предложения. В свою очередь, в регулируемых секторах, где главным критерием эффективности служит рост натуральных показателей количества и качества продукции, ее стоимость устанавливается государственным регулятором (в России это Федеральная служба по тарифам и региональные службы).

Наиболее значимые взаимосвязи перечисленных процессов, предписываемые нормативно-правовыми актами, указаны на следующей схеме (ср. модель розничного рынка электроэнергии [107, с.81]).

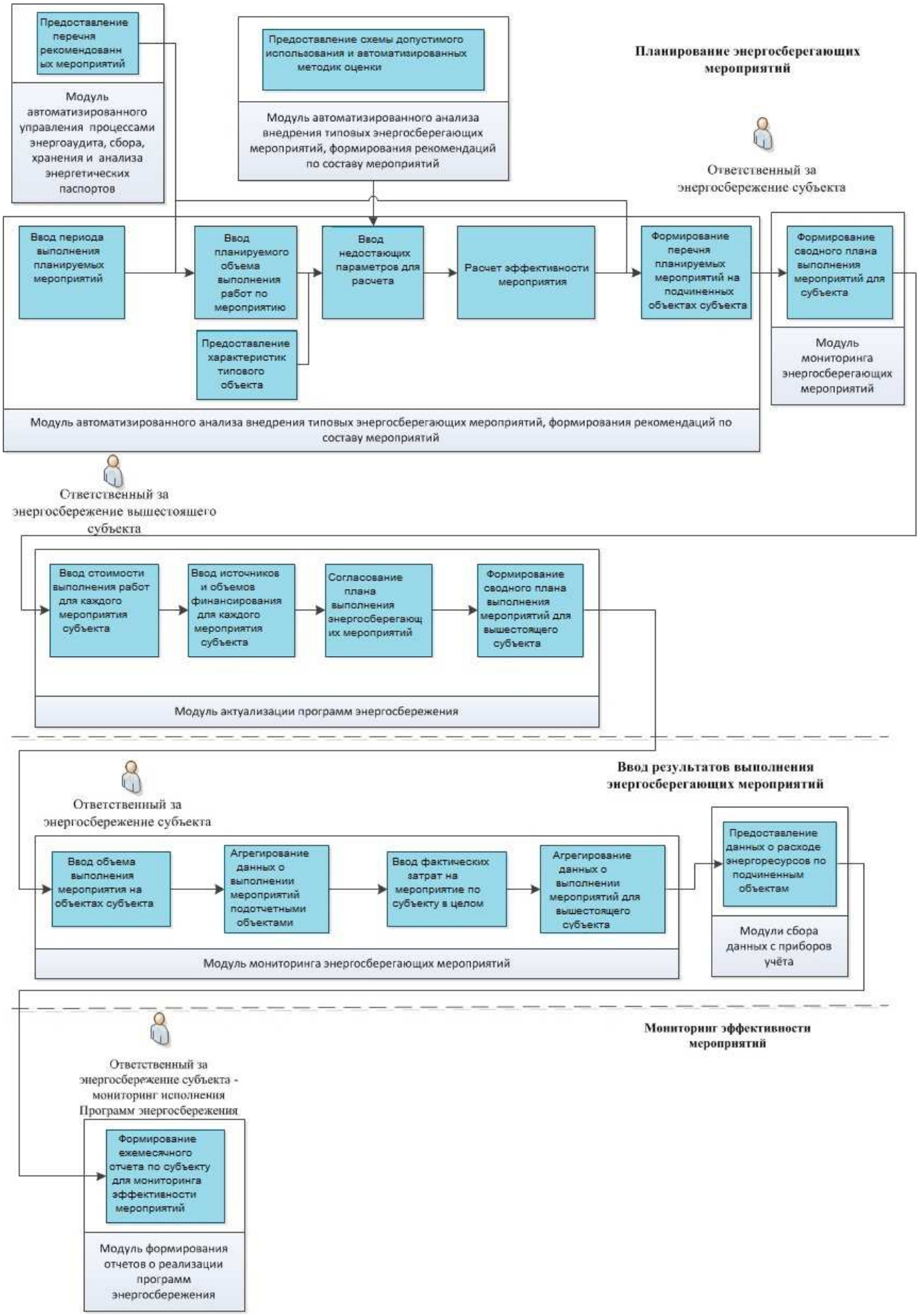


В число управляющих и обеспечивающих процессов включены только непосредственно связанные с основными (будем называть их первичными). Иными словами, наша модель процессов не замкнута относительно телеологических связей (вида «цель–средство», см. разд.1.1) между функциями, подлежащими (полной либо частичной) автоматизации. Прослеживание таких связей приводит к выявлению многочисленных дополнительных процессов, обеспечивающих и управляющих по отношению к первичным (вторичные процессы), на следующем шаге – дополнительных по отношению к вторичным, и т.д. Многие из них встречаются почти во всех системах управления: целеполагание, уплата налогов, хранение запасных частей и принадлежностей, ведение всевозможных информационных реестров, обучение и подготовка кадров, и т.д. (На некотором шаге перечисления таких процессов появляется даже процесс подготовки настоящей работы). Кроме них, в ТЭК присутствует и специфический блок – процессы, направленные на энергосбережение и повышение энергетической эффективности. Они относятся к до-

полнительным, поскольку уменьшение объема расхода энергоресурсов в натуральном выражении не входит в число критериев эффективности основных процессов и процессов первого уровня. Наиболее прямым образом этот критерий связан со стремлением потребителей к снижению энергозатрат, однако такая связь порождает управляющие воздействия на процесс потребления только при дефиците либо достаточно высокой стоимости энергоресурсов. В противном случае потребитель предпочитает отказаться от выполнения медленно окупаемых энергосберегающих мероприятий, а также сохранить сложившийся уклад деятельности. А поскольку производители почти всегда естественным образом заинтересованы в увеличении объема потребления, критерий энергоэффективности остается только как производный для процессов государственного управления – в виде целевого показателя безопасности, доступности и развития отрасли. В России основу правового регулирования в сфере энергосбережения составляет федеральный закон №261-ФЗ «Об энергосбережении…». Согласно этому закону и подзаконным актам, в блок энергосбережения входят следующие вторичные процессы:

- проведение энергетических обследований;
- организация расчета за потребленные энергоресурсы по показаниям измерительных приборов;
- организация выполнения энергосберегающих мероприятий;
- энергосервисная деятельность;
- установление требований энергетической эффективности товаров, работ, услуг для государственных или муниципальных нужд;
- присвоение класса энергетической эффективности товарам и оборудованию;
- проверка соответствия зданий, строений, сооружений требованиям энергетической эффективности;
- утилизация осветительных устройств;
- экономическое стимулирование энергосбережения;
- пропаганда и обучение в сфере энергосбережения;
- деятельность координационных советов по энергосбережению;
- программно-целевое управление энергосбережением;
- размещение информации в Государственной информационной системе «Энергоэффективность».

Эти процессы носят ярко выраженный сквозной характер. Приведем в качестве примера процесс организации выполнения энергосберегающих мероприятий, изображенный на следующих схемах вместе с наименованиями компонентов, автоматизирующих выполнение отдельных шагов [124]. Видно, что этот процесс затрагивает практически все первичные обеспечивающие процессы.

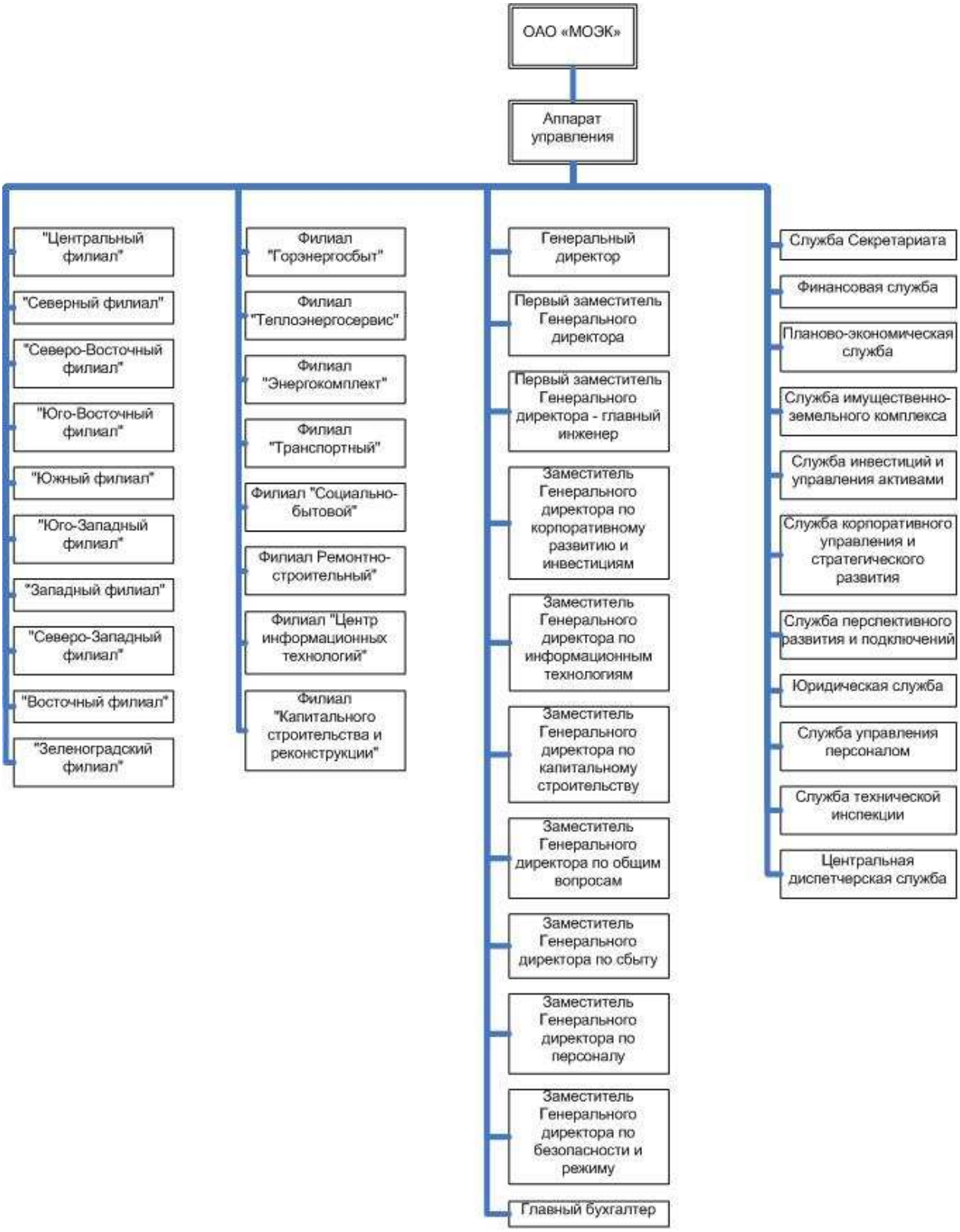

Планирование энергосберегающих мероприятий
Предоставление перечня рекомендованных мероприятий
Модуль автоматизированного управления процессами энергоаудита, сбора, хранения и анализа энергетических паспортов
Предоставление схемы допустимого использования и автоматизированных методик оценки
Модуль автоматизированного анализа внедрения типовых энергосберегающих мероприятий, формирования рекомендаций по составу мероприятий
Ответственный за энергосбережение субъекта
Ввод периода выполнения планируемых мероприятий
Ввод планируемого объема выполнения работ по мероприятию
Ввод недостающих параметров для расчета
Расчет эффективности мероприятия
Формирование перечня планируемых мероприятий на подчиненных объектах субъекта
Формирование сводного плана выполнения мероприятий для субъекта
Модуль мониторинга энергосберегающих мероприятий
Предоставление характеристик типового объекта
Модуль автоматизированного анализа внедрения типовых энергосберегающих мероприятий, формирования рекомендаций по составу мероприятий
Ответственный за энергосбережение вышестоящего субъекта
Ввод стоимости выполнения работ для каждого мероприятия субъекта
Ввод источников и объемов финансирования для каждого мероприятия субъекта
Согласование плана выполнения энергосберегающих мероприятий
Формирование сводного плана выполнения мероприятий для вышестоящего субъекта
Модуль актуализации программ энергосбережения
Ввод результатов выполнения энергосберегающих мероприятий
Ответственный за энергосбережение субъекта
Ввод объема выполнения мероприятия на объектах субъекта
Агрегирование данных о выполнении мероприятий подотчетными объектами
Ввод фактических затрат на мероприятие по субъекту в целом
Агрегирование данных о выполнении мероприятий для вышестоящего субъекта
Предоставление данных о расходе энергоресурсов по подчиненным объектам
Модули сбора данных с приборов учёта
Модуль мониторинга энергосберегающих мероприятий
Мониторинг эффективности мероприятий
Ответственный за энергосбережение субъекта - мониторинг исполнения Программ энергосбережения
Формирование ежемесячного отчета по субъекту для мониторинга эффективности мероприятий
Модуль формирования отчетов о реализации программ энергосбережения

Перейдем к рассмотрению организационной структуры ТЭК. У потребителей организация процессов в области энергетики в большой степени зависит от вида их основной деятельности, по которому их часто классифицируют дополнительно. Например, приказ Министерства энергетики России №600 от 14 декабря 2011 г. «Об утверждении Порядка составления топливно-энергетических балансов субъектов Российской Федерации, муниципальных образований» предписывает раздельно учитывать энергопотребление на объектах сельского хозяйства, промышленности, строительства, транспорта и связи, сферы услуг, населения. (Правда, здесь неясно, куда отнести специальные виды деятельности, такие как оборона или научные исследования). Отдельно учитывается использование топливно-энергетических ресурсов в качестве сырья и на нетопливные нужды (например, получение аммиака из природного газа). Особо выделяется потребление энергоресурсов самими объектами ТЭК (порождающее на схеме процессов обратные связи, входящие в процесс выработки и добычи), в том числе:

- выработка электрической и тепловой энергии из ископаемого топлива;
- переработка нефти и газа (в другие виды топлива), обогащение угля;
- питание энергетического оборудования (собственные нужды);
- потери энергоресурсов при передаче.

Объемы производства, переработки и потребления учитываются раздельно по видам энергоресурсов, ископаемых (нефть, газ, уголь и прочее твердое топливо), вырабатываемых (нефтепродукты, электрическая и тепловая энергия), даровых (атомная энергия, гидроэнергия и нетрадиционные возобновляемые источники энергии). Ряд нормативных актов предписывает рассматривать в качестве энергоресурса также воду, пар, сжатые газы. Вид энергоресурса предопределяет не только технологическое содержание, но и организационную форму процессов. В России наиболее мелкое дробление субъектов на самостоятельные юридические лица имеет место в электроэнергетике, где даже на законодательном уровне запрещено совмещать регулируемые и конкурентные виды деятельности в рамках одной организации. Здесь есть выделенные юридические лица диспетчерского управления (самое крупное из них – ОАО «СО ЕЭС») и оператор учета (на оптовом рынке – это ОАО «АТС»). Поставщики делятся на два непересекающихся вида: сбытовые организации и гарантирующие поставщики. Они выполняют одну и ту же функцию, однако руководствуются совершенно разными критериями эффективности. Сбыт представляет собой коммерческую посредническую деятельность между производителями и потребителями, которые свободно выбираются в целях максимизации прибыли, а гарантированная поставка – услугу, оказываемую любому платежеспособному потребителю за счет фиксированной надбавки, установленной органами тарифного регулирования.

В сфере оборота других энергоресурсов (кроме электроэнергии) функции поставщиков часто берут на себя производители – такие комплексные субъекты называются энерго- или ресурсоснабжающими организациями. Существуют и субъекты, деятельность которых включает почти все энергетические процессы. Приведем в качестве примера организационную структуру ОАО «МОЭК» – основной теплоснабжающей организации города Москвы [100].



Отметим, что высокий уровень комплексности не обязательно связан с большим масштабом объектов: он характерен также для исполнителей коммунальных услуг, которым приходится и производить энергию (например в котлах автономного отопления), и передавать ее по внутридомовым сетям, и поставлять жителям домов, и проводить ремонтные работы, и даже инвестировать целевые взносы жильцов в развитие домашнего энергохозяйства. Сложность этих задач компенсируется малым масштабом объектов ЖКХ.

Управление энергетическими процессами требует колоссального потока документов. Организация документооборота в ТЭК основана на тех же принципах и сталкивается с теми же проблемами, что и в других отраслях производства. Большинство энергетических процессов сопровождается регулярными циклами планирования и отчетности, в ходе которых формируются документы следующих видов:

- планы, программы, проекты выполнения функций/услуг с различными горизонтами (от суток до десятилетий);
- заявки (задания) на выполнение одного шага плана (а также на однократное внеплановое выполнение функции/услуги);
- акты выполнения функций/услуг;
- отчеты о выполнении планов за период.

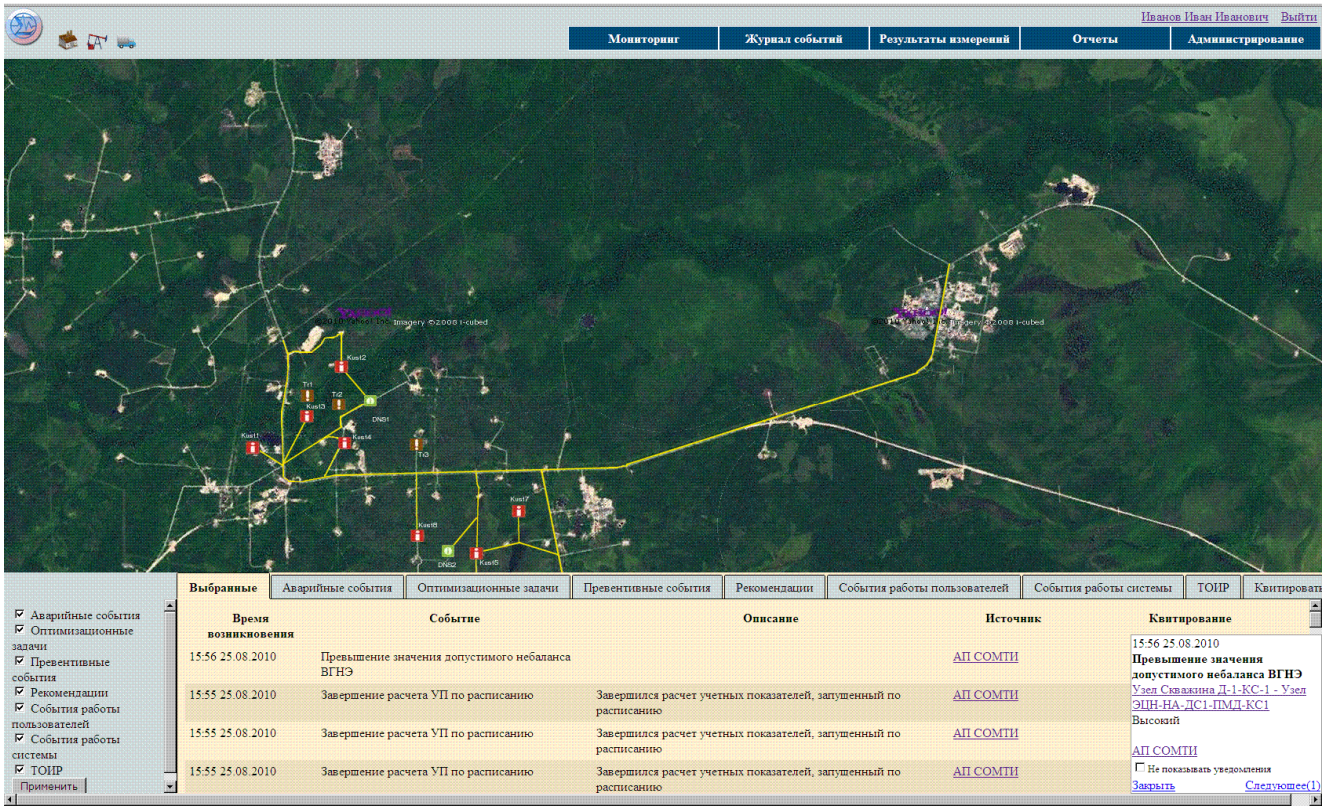
В дополнение к ним по ходу выполнения функций/услуг создается много других документов: организационно-правовых, распорядительных, информационно-справочных, кадровых [88]. Например, процессы в области энергосбережения оперируют следующими специфическими документами:

- энергетический паспорт объекта;
- отчет об энергетическом обследовании;
- программа энергосбережения;
- энергосервисный контракт;
- отчет об исполнении программ энергосбережения;
- информация для включения в Государственную информационную систему «Энергоэффективность»;
- решение координационного совета;
- нормативно-техническая документация в области энергосбережения;
- рекламные и информационные материалы по энергосбережению.

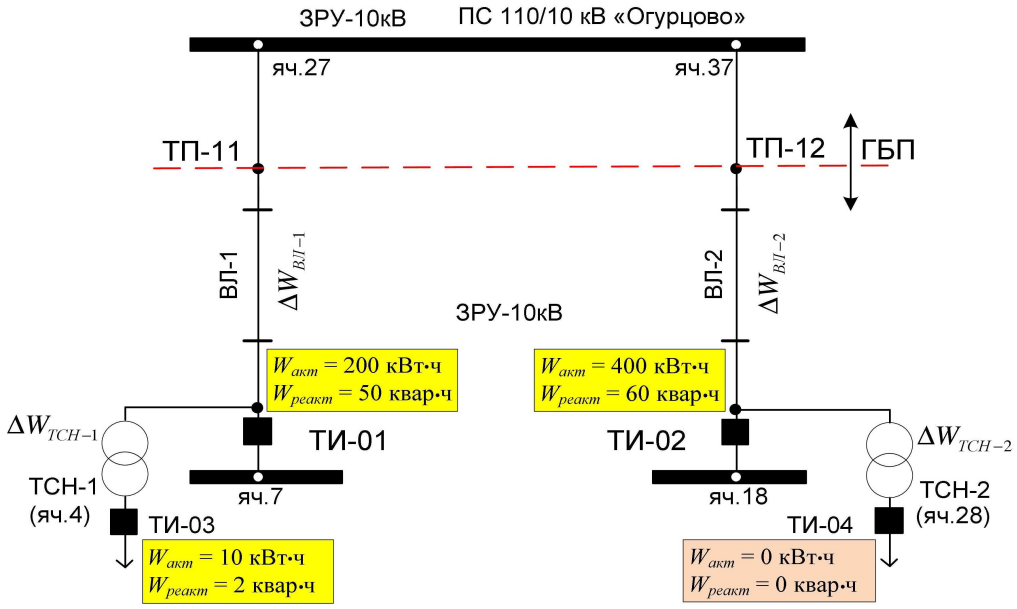
В документах многократно дублируется одна и та же информация, причем не вследствие ошибок их составления, а в целях обеспечения самодостаточности каждого документа. Это совершенно необходимо при размещении документов на бумажном носителе, но неизбежно при-

водит к появлению многих значений одного и того же информационного элемента, с неясной процедурой выбора достоверного значения. При автоматизации документооборота строятся модели документов, содержащие описание состава и расположения их полей, форматы полей, привязку к элементам онтологии и модели данных, критерии корректного заполнения, разграничение прав участников процессов на чтение/изменение/согласование значений полей. Здесь много сил уходит на нормирование документов, включая отделение информационных дублей от случаев омонимии в наименованиях полей, возникающей в том числе вследствие концептуального разрыва между уровнями АСУТП и АСУП. Составление моделей всех значимых видов документов служит важнейшей предпосылкой к достижению единства информационной базы деятельности субъекта. Тогда документ становится ее вторичным артефактом – одной из форм обмена информацией между участниками процессов. Посредством электронных документов в машинно-читаемых форматах, в первую очередь основанных на языке XML, взаимодействуют средства автоматизации процессов [97]. Например, в России для обмена информацией об объеме энергопотребления широко применяется XML-формат 80020, разработанный ОАО «АТС» [1].

Как форма представления информации, документ обладает двумя известными недостатками: статичностью и низкой наглядностью [94]. Они затрудняют использование документов как носителей исходных данных для принятия оперативных решений. В связи с этим в управлении ТЭК сложились традиции наглядного интерактивного изображения объектов на плоскости видеомонитора вместе с оперативной индикацией их состояния [11]. Протяженные объекты изображаются на цифровой географической карте, например видеокадр мониторинга участка сети промысловых нефтепроводов может выглядеть следующим образом [52].



Недостатком картографического изображения является линейность масштаба, входящая в эргономическое противоречие с неоднородностью расположения энергоустановок: около крупных населенных пунктов на небольшом участке размещается много объектов, в то время как на карте большой малонаселенной области трудно разыскать изображения редких разбросанных объектов. Кроме того, картографическая нотация не содержит средств для изображения внутреннего устройства локальных объектов, таких как цех или жилой дом. Поэтому часто объект изображается в виде мнемосхемы [115] – графа, в узлах которого располагаются стандартные условные обозначения локальных энергоустановок и единиц оборудования, а ребра изображают участки энергетических сетей. Направления и длины ребер графа выбирают исходя из удобства восприятия мнемосхемы, а не из географической ориентации и протяженности сети. Около узлов размещаются всплывающие окна, в которых оперативно показываются текущие значения основных режимных параметров и сигналы. Предусматривается возможность выбрать любой элемент и вызвать дополнительную информацию по нему: таблицу его паспортных характеристик, журнал состояния, график нагрузки, историю технического обслуживания и ремонта. Приведем пример мнемосхемы питания промышленной энергоустановки класса напряжения 10 кВ (ее оборудование описывается фрагментом модели CIM, приведенным в начальной части настоящего раздела).



Первичным источником оперативных данных, визуализируемых на мнемосхемах, служат технические средства измерения, сигнализации и регулирования. Перечислить их виды и протоколы взаимодействия с ними в настоящей работе невозможно ввиду их многочисленности, поэтому мы только напомним общие принципы их применения. Измерения выполняются периодически, формируя в информационно-управляющей системе временны*е* ряды значений расхода энергоресурсов, а также параметров их качества (напряжение электрической сети, температура теплоносителя, теплотворная способность газа и др.), состояния оборудования (температура масла в кожухе трансформатора, положение подвижного объекта и др.) и окружающей среды (температура, загазованность и др.). Сигналы, сообщающие о переключениях коммутационных аппаратов, коротких замыканиях, порывах трубопроводов и т.д., приходят с регистрирующих устройств по мере возникновения. «Навстречу» результатам измерений и сигналам из информационно-управляющей системы к техническим средствам поступают команды опроса, регулирования, параметрирования. В целях соблюдения требований реального времени, система взаимодействует с устройствами, непосредственно подключенными к энергетическому оборудованию, не напрямую, а через многофункциональные контроллеры управления энергоустановками. Контроллеры автоматически выполняют функции сбора данных и регулирования в реальном времени, согласно задаваемым системой уставкам, таким как период опроса приборов, режимный температурный график, максимальная выходная электрическая нагрузка, уровень заполнения нефтехранилища и др. Повышение уровня интеллектуальности технических средств и развитие протоколов взаимодействия с ними входит в число основных направлений разработки систем класса Smart Grid [58]. Приведем в качестве примера спецификацию обмена данными с контроллером ФИОРД, основанную на стандарте IEC 60870-5-104 [51] (Таблица 8).

Таблица 8

Тип	Название	Обозначение
Информация о процессе в направлении контроля		
1	Одноэлементная информация	M_SP_NA_1
9	Значение измеряемой величины, нормализованное значение	M_ME_NA_1
11	Значение измеряемой величины, масштабированное значение	M_ME_NB_1
13	Значение измеряемой величины, короткий формат с плавающей запятой (4 байта)	M_ME_NC_1
15	Интегральные суммы	M_IT_NA_1
30	Одноэлементная информация с меткой времени CP56Время 2a	M_SP_TB_1
34	Значение измеряемой величины, нормализованное значение с меткой времени CP56Время 2a	M_ME_TD_1
35	Значение измеряемой величины, масштабированное значение с меткой времени CP56Время 2a	M_ME_TE_1
36	Значение измеряемой величины, короткий формат с плавающей запятой с меткой времени CP56Время 2a	M_ME_TF_1
37	Интегральные суммы с меткой времени CP56Время 2a	M_IT_TB_1
Информация о процессе в направлении управления		
45	Однопозиционная команда	C_SC_NA_1
48	Команда уставки, нормализованное значение	C_SE_NA_1
49	Команда уставки, масштабированное значение	C_SE_NB_1
50	Команда уставки, короткий формат с плавающей запятой	C_SE_NC_1
58	Однопозиционная команда с меткой времени CP56Время 2a	C_SC_TA_1
61	Команда уставки, нормализованное значение с меткой времени CP56Время 2a	C_SE_TA_1
62	Команда уставки, масштабированное значение с меткой времени CP56Время 2a	C_SE_TB_1
63	Команда уставки, короткий формат с плавающей запятой с меткой времени CP56Время 2a	C_SE_TC_1

Информация, поступающая в систему от средств измерения, а также из входящих документов и с автоматизированных рабочих мест пользователей, используется в качестве входной для разнообразных задач расчета и анализа. В первую очередь оперативно вычисляются режимные показатели – результаты распространения значений параметров количества и качества энергоресурсов вдоль оси времени (статистический анализ, прогнозирование), потоков энергии (замещение отсутствующих значений, расчет отпуска из сети по приему в сеть и др.), иерархии управления объектом (например формирование сводного топливно-энергетического баланса города). Решение таких задач сводится к суперпозиции базовых алгоритмов, определяющих один шаг распространения (прогноз изменения потребности энергоустановки в энергоресурсе за один расчетный период, объем потери энергоресурса в элементе энергетической сети и т.п.). Порядок применения суперпозиции задается каузальными, топологическими и мереологическими связями между сущностями, составляющими объект. Рассмотрим в качестве примера задачу расчета объема потребления электроэнергии энергоустановкой, изображенной на вышеприведенной мнемосхеме. По определению, потребление представляет собой суммарный переток по границе балансовой принадлежности (ГБП), отделяющей объекты потребителя от смежного объекта энергетической сети [107]. На схеме ГБП, изображенная красной пунктирной линией, состоит из двух точек поставки ТП-11 и ТП-12 – наконечников линий, отходящих на подстанцию «Огурцово», принадлежащую сетевой организации. Счетчики, позволяющие узнать

переток электроэнергии путем непосредственного измерения, находятся не на ГБП, а в точках ТИ-01–ТИ-04, поэтому объем потребления энергоустановки можно установить только расчетным путем. Алгоритм расчета потребления в ТП-11 состоит из трех шагов: сначала вычисляется объем потребления на первичной обмотке трансформатора ТСН-1 (путем суммирования показаний счетчика ТИ-03 с объемом потерь в трансформаторе), затем к нему прибавляются показания счетчика ТИ-01 (правило Кирхгофа), и наконец – объем потерь в воздушной линии ВЛ-1. Используя явные формулы расчета потерь активной энергии в единицах оборудования [113], имеем:

$$W_{\text{ТП-11}} = W' + \Delta W_{ВЛ\text{-}1}(W'),$$

$$W' = W'' + W_{\text{ТИ-01}},$$

$$W'' = W_{\text{ТИ-03}} + \Delta W_{ТСН\text{-}1}(W_{\text{ТИ-03}}),$$

$$\Delta W_{ВЛ\text{-}1}(W) = \frac{P^2 + Q^2}{U_{ном}^{\ 2}} \cdot r \cdot l \cdot T,$$

$$\Delta W_{ТСН\text{-}1}(W) = \Delta P_{XX} \cdot T + \Delta P_{КЗ} \cdot \frac{P^2 + Q^2}{S^2} \cdot T,$$

где $P = P(W) = W_{акт}/T$ – текущая активная мощность (кВт), $Q = Q(W) = W_{реакт}/T$ – текущая реактивная мощность (квар), T – период измерения (ч), остальные параметры берутся из паспорта объекта: $U_{ном}$ – номинальное напряжение воздушной линии (кВ), r – погонное сопротивление линии (Ом/км), l – длина (км), ΔP_{XX} – потери активной мощности в трансформаторе на холостом ходу (кВт), $\Delta P_{КЗ}$ – потери активной мощности короткого замыкания (кВт), S – номинальная мощность трансформатора (кВА). Точно так же рассчитывается объем потребления в ТП-12 $W_{\text{ТП-12}}$ по показаниям счетчиков ТИ-02 и ТИ-04. Сумма $W_{\text{ТП-11}} + W_{\text{ТП-12}}$ и составляет объем потребления энергоустановки.

Расчет путем навигации по истории, топологии и иерархии объектов требуется также для показателей технического состояния, определяющих уровень износа, надежность, потребность в ремонте и/или модернизации. Многие режимные и технические показатели переводятся в стоимостную форму, причем алгоритмы перевода согласно разнообразным тарифным и сметным нормативам могут быть достаточно сложными. Массивы значений всех показателей служат входными данными для выявления необходимых управляющих воздействий на энергообъект. Вычисляются не только их фактические значения, но и возможные: плановые, требуемые, прогнозные, предельные, усредненные, оптимальные с точки зрения того или иного критерия, нормативы, оценки, уставки. Расчет многих видов возможных значений требует привлечения мощных методов анализа данных [172]. Далее выявляются расхождения между фактическими и возможными значениями, выходящие за рамки допустимых. Выясняются их причины: либо в данных или алгоритмах расчета (в том числе в паспорте объекта) допущены ошибки, либо ре-

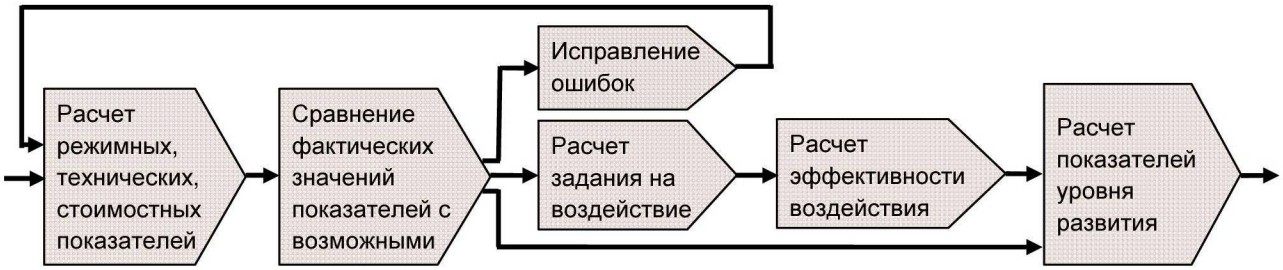
жим объекта выходит за пределы нормального. В первом случае ошибки исправляются, а во втором рассчитывается воздействие, способное наиболее эффективно привести к возврату значений в целевые рамки.

Для расчета воздействий применяются методы математического программирования [141], экспертные системы [49], имитационное моделирование [3] и др. В результате формируется документ – задание на воздействие, которое после утверждения диспетчером при наличии технической возможности исполняется автоматически, а в противном случае передается обслуживающему персоналу. В нашем примере энергоустановки можно по результатам расчета потерь автоматически обнаружить, что КПД трансформатора ТСН-1 чрезмерно низок вследствие нерациональной (пониженной либо повышенной) нагрузки. В этом случае рассчитывается карта переключений, приводящих к ее оптимизации (алгоритм для этого можно найти в [15, разд.4.2]). Перераспределение нагрузки на трансформаторы относится к числу энергосберегающих мероприятий, способных привести к заметному снижению потерь электроэнергии при невысоких затратах [87].

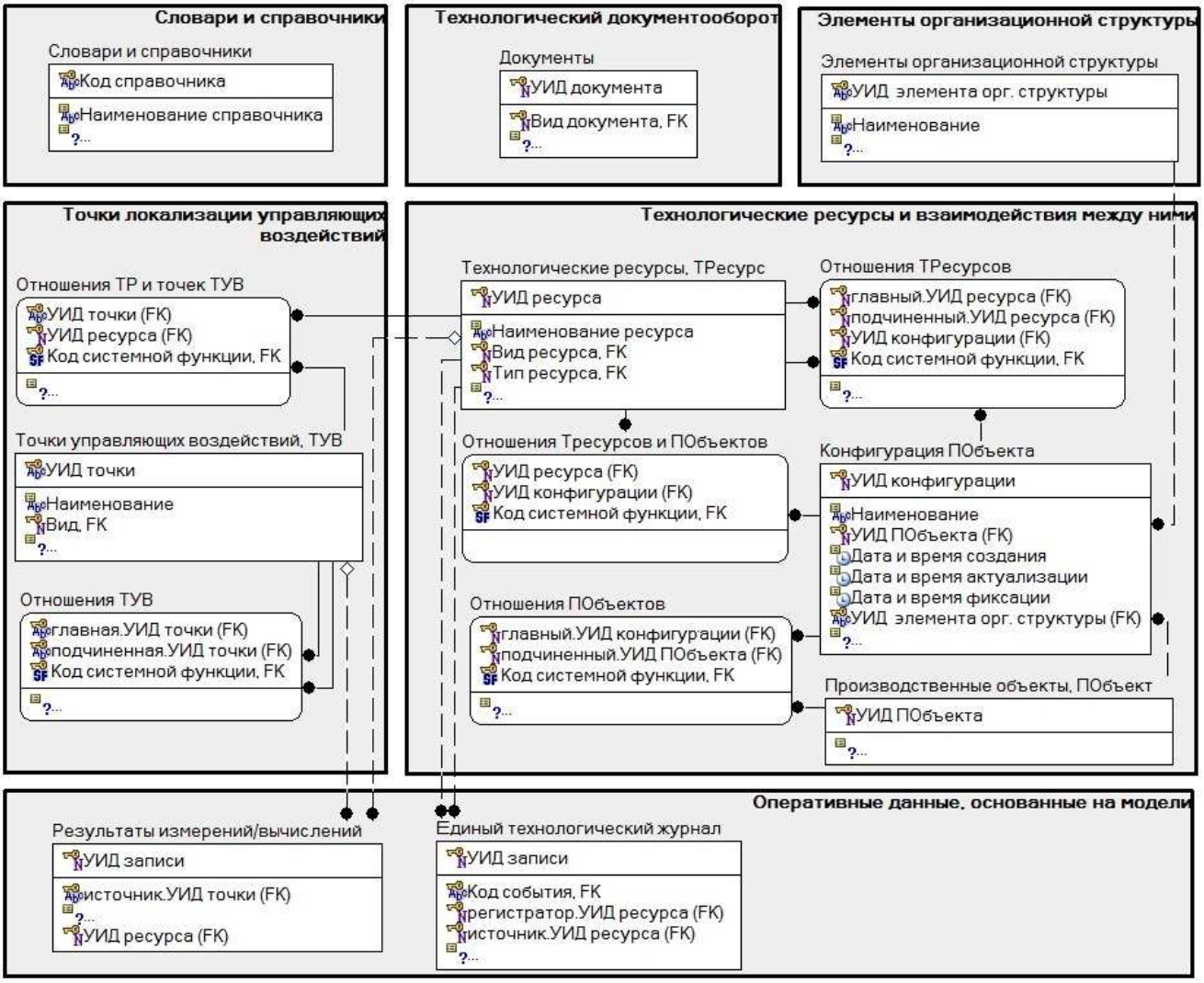
На следующем шаге цикла расчета, в соответствие со стандартом энергетического менеджмента, вычисляются показатели эффективности исполнения задания. Они делятся на производственно-технологические (достигнутые значения целевых показателей воздействия – показатели уровня АСУТП) и организационно-финансовые (сроки и стоимость работ – показатели уровня АСУП). Геометрическое среднее отношений их фактических значений к ожидаемым образует безразмерную комплексную оценку [108]. Неудовлетворительно низкая оценка может служить основанием для выполнения корректирующих воздействий на процессы, направленных на их совершенствование.

В завершение цикла расчета вычисляются агрегатные показатели уровня развития объекта управления, характеризующие его способность удовлетворять потребности, новизну техники, эффективность технологий (в том числе энергетическую) [37]. Эти показатели используются в процессах стратегического управления, в том числе в качестве целевых в долгосрочных программах развития ТЭК. Например, целью Государственной программы «Энергосбережение и повышение энергетической эффективности на период до 2020 года» является «снижение энергоемкости валового внутреннего продукта Российской Федерации на 40% в 2007-2020 годах» (см. распоряжение Правительства Российской Федерации от 27 декабря 2010 г. №2446-р).

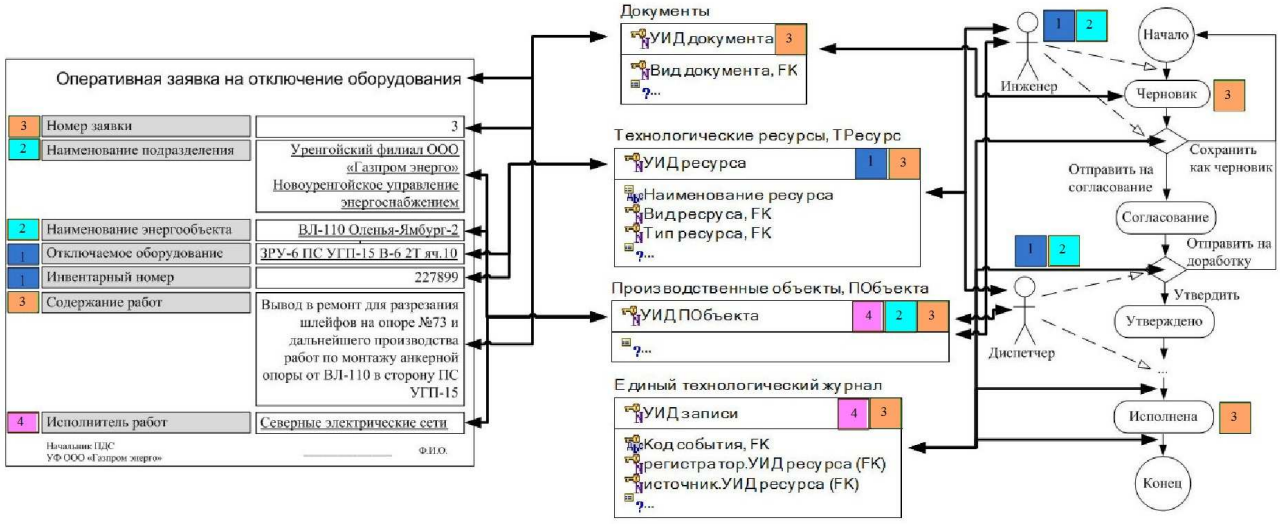
Таким образом, получается следующая схема типового цикла расчета.



На этом мы завершим обзор пространства задач ТЭК. Нормированные наименования всех составляющих его классов автоматизируемых функций (concerns) собираются в справочник, записи которого выступают в качестве меток единиц хранения данных и шагов процессов. Некоторые массивы данных размечаются (классифицируются) дополнительно, если это предусмотрено онтологией: выделяются виды объектов и оборудования, формы (виды) документов. В результате базовые онтологические понятия и отношения порождают следующую схему данных, организованную в соответствие с принципами разд.5.2 [2].



Эта модель данных пригодна (после необходимой детализации) к применению в качестве типовой для информационно-управляющих систем ТЭК. Физическая структура базы данных генерируется по ней автоматически, причем в целях повышения быстродействия доступа к данным генерируются правила секционирования таблиц по значениям атрибута-метки. Сущности связаны (в смысле АОП) разнообразными отношениями вида «многие-ко-многим», метками в которых выступают коды функций системы. Использование списка функций в качестве единого массива меток позволяет прозрачно трассировать элементы данных к другим артефактам проектирования. Приведем пример трассирования в ходе автоматизации процесса технического обслуживания и ремонта оборудования [2].



Здесь цифрами в квадратиках обозначены классы задач: (1) ведение реестра оборудования, (2) ведение реестра объектов, (3) документооборот, (4) мониторинг состояния объекта. Они выполняются поочередно в ходе процесса, образованного событиями жизненного цикла заявки на ремонт воздушной линии электропередачи: составление, согласование, утверждение и исполнение (правая часть схемы). Показано, как задачи перемешиваются и рассеиваются по сущностям типовой модели данных (средняя часть схемы) и далее по модели документа «Оперативная заявка на отключение оборудования» (левая часть). Тем не менее, взаимосвязь между ними легко восстанавливается благодаря разметке, поэтому сквозное трассирование не требует значительных затрат труда. Как указывалось в разд.5.3, по журналу событий можно восстановить историю изменений объекта, вызванных ремонтом. С событиями можно связать (в смысле АОП) и другие обработчики, например проверку соответствия фактического хода работ нормальному/плановому, чтобы автоматически оперативно обнаруживать опасные отклонения (такие как попытки производить работы под напряжением) и вносить необходимые коррективы. Такое динамическое связывание аспектов открывает широкие возможности для тонкой настройки порядка выполнения перемешанных и рассеянных задач непосредственно в работающей системе, без дублирования их программного кода и вообще без участия программистов

[8]. А если разработчики привлекаются для модернизации системы, то благодаря трассированию они изменяют все компоненты синхронно, сохраняя их взаимную совместимость.

Архитектурный стиль взаимодействия воплощается в компонентах информационно-управляющих систем ТЭК согласно типовым подходам разд.1.3. Каждый сеанс взаимодействия проходит под контролем многоуровневой системы защиты информации, включающей разграничение прав доступа пользователей как по ролям (должностям), так и по участкам объекта согласно организационной структуре. Сложные контрольные списки доступа (access control lists) оформляются как профили выборки данных [89]. Для взаимодействия с техническими средствами управления и контроллерами энергоустановок создаются драйверы, функционирующие под управлением общего ядра [10]. Компоненты взаимодействия с пользователями встраиваются в единый портал на базе web-технологий [81]. Взаимодействие со смежными системами и между подсистемами организуется посредством интеграционной платформы на базе сервисно-ориентированной архитектуры.

Обратимся к вопросам проектирования вычислительных задач. Как было показано выше, компоненты расчета режимных и технических показателей путем навигации по топологическим и каузальным связям представляют собой проблемно-ориентированные модели вычислений, в которых совокупность примитивов и вид вычисляемых функций задаются паспортом объекта управления. Для анализа корректности (в смысле отсутствия переполнения и грубого округления) и эффективности таких моделей применяется подход, изложенный в гл.3. Например, расчет объема потребления электроэнергии производится на множестве всех пар вида ($W_{акт}$, $W_{реакт}$), состоящих из машинных чисел с подходящей фиксированной точностью (обычно до целых Вт·ч). Объем потребления в точке ТП-11 вычисляется путем суперпозиции машинных реализаций функций, образующих множество {+, $\Delta W_{ВЛ\text{-}1}$, $\Delta W_{ТСН\text{-}1}$}. Более сложные схемы энергоснабжения требуют привлечения операций вычитания, умножения на фиксированный коэффициент, расчета потерь в оборудовании других видов.

В соответствие с подходом MDE, программная реализация таких моделей автоматически генерируется по наполнению паспорта в виде динамически подгружаемых модулей с фиксированным интерфейсом [64]. Генератор поставляется в составе системы, поскольку он вызывается при изменениях паспорта для обновления изменившихся алгоритмов. Генератор обходит граф участка энергетической сети, показатели которого требуется рассчитать, и порождает конструкторы классов, вычисляющих значения показателей отдельных элементов, с подстановкой значений их паспортных характеристик. Например, программный код на языке Java, сгенериро-

ванный для описанного выше алгоритма расчета потребления электроэнергии в точке ТП-11, состоит из следующих двух классов[2] (комментарии добавлены нами).

```
// Алгоритм расчета объема потребления
// на первичной обмотке трансформатора ТСН-1

public class Algorithm2 extends LinearAlgorithmWithLosses
{
  public Algorithm2()
  {
    // Регистрация алгоритма расчета потерь в двухобмоточном
    // трансформаторе ТСН-1

    energyConsumers = new EnergyConsumer[] {

      // Transformer2(S (кВА), Uвн (кВ), Uнн (кВ),
      //              dPхх (кВт), dPкз (кВт))

      new Transformer2(63, 10, 0.4, 0.26, 1.28)
    };
  }
}

// Алгоритм расчета объема потребления в точке ТП-11

public class Algorithm1 extends LinearAlgorithmWithLosses
{
  public Algorithm1()
  {
    // Регистрация алгоритма Algorithm2 для расчета потерь
    // в трансформаторе ТСН-1
    // по результатам измерений расхода энергии в точке ТИ-03

    accPointGroupCodes.put("03", Algorithm2.class.getName());

    // Регистрация алгоритма для расчета потерь в воздушной линии ВЛ-1
    // по результатам суммирования объема потребления
    // на первичной обмотке трансформатора ТСН-1
    // с результатами измерений расхода энергии в точке ТИ-01

    energyConsumers = new EnergyConsumer[] {

      // AerialCircuit(ro (кв.мм), r (Ом/км), l (км), Uном (кВ))

      new AerialCircuit(394, 0.073, 1.33, 10)
    };
  }
}
```

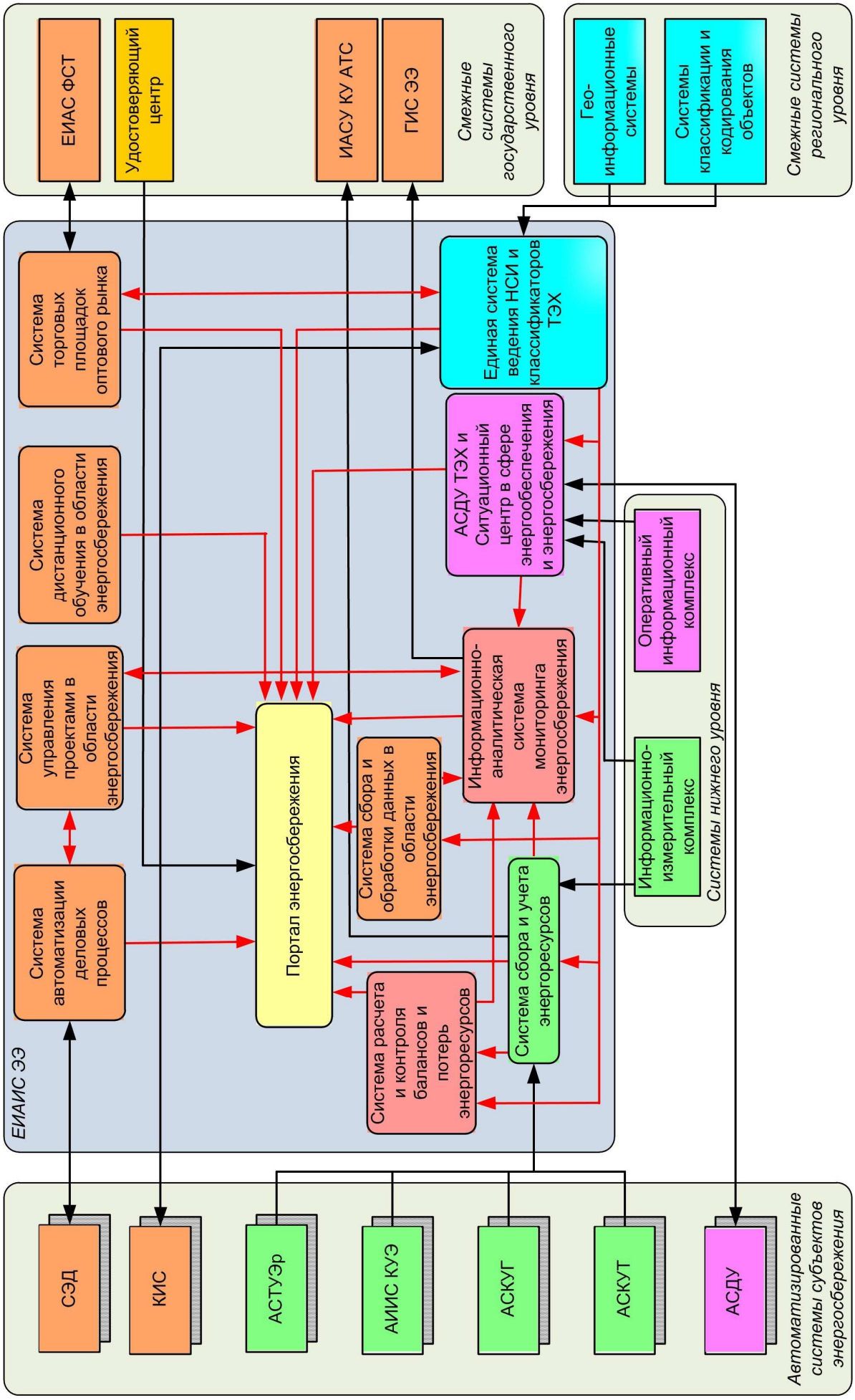
Модули такого типа развертываются в распределенной вычислительной среде, организованной на базе технологий J2EE [129], и исполняются в потоковом режиме. Для решения остальных расчетных и аналитических задач привлекаются специализированные средства, которые также оформляются как компоненты для развертывания в распределенной вычислитель-

[2] Программная реализация выполнена К.Кучерявым.

ной среде. Циклы расчета инициируются разнообразными способами: по расписанию, по запросу оператора, по возникновению потребности в результатах, по поступлению входных данных, по наличию свободных вычислительных ресурсов. Разные проходы по циклу могут отличаться составом показателей, алгоритмов, участков объекта, исторических значений. Шаблоны проходов задаются профилями выборки данных [64] (см. разд.1.3).

Таким образом, формируется полноценное пространство решений для ТЭК. При создании и развитии больших информационно-управляющих систем его компоненты непрерывно порождаются из пространства задач (в том числе автоматически) и комплексируются друг с другом, согласно потребностям заказчиков и особенностям объектов. Приведем в качестве примера ЕИАИС ЭЭ – систему управления энергосбережением города Москвы [84]. Эта система предназначена для комплексной автоматизации процессов реализации Государственной программы «Энергосбережение в городе Москве», рассчитанной на перспективу до 2020 года. Соисполнителями программы являются департаменты правительства и префектуры административных округов города Москвы. Мероприятия программы по состоянию на 2012 г. охватывают более 100 тыс. объектов энергосбережения, в том числе 40 тыс. жилых домов, 15 тыс. нежилых зданий и сооружений, 15 тыс. электроподстанций, 80 тыс. км линий электропередачи, 9 тыс. км тепловых сетей, 8 тыс. км газовых сетей (все значения примерные). Классификатор видов объектов и оборудования насчитывает более 600 позиций, распределенных по 7 уровням иерархии наследования [38]. Согласно программе, совокупное энергопотребление объектов составляет около 40 млн. тонн условного топлива в год. Хотя этот и другие показатели рассчитываются ежеквартально/ежегодно в рамках отчетности об исполнении программы, расчет потребления отдельных объектов при подборе их оптимального энергетического режима требуется выполнять несколько раз в час. При этом для анализа долговременных тенденций требуются временные ряды энергетических показателей начиная с 2007 г. Такие требования порождают высокую нагрузку как на средства хранения и обработки данных ЕИАИС ЭЭ, так и на разработчиков системы, принуждая к глубокой автоматизации процедур создания баз данных и функциональных компонентов.

Разнообразие функциональных задач ЕИАИС ЭЭ наглядно показано на приведенной ниже схеме архитектуры. Здесь цветовая раскраска указывает вхождение компонентов в подсистемы типовой архитектуры большой информационно-управляющей системы из разд.1.3. Видно, что ввиду сквозного характера процессов энергосбережения, ЕИАИС ЭЭ вобрала в себя средства автоматизации всех процессов ТЭК.


ЕИАС ФСТ
Удостоверяющий центр
ИАСУ КУ АТС
ГИС ЭЭ
Смежные системы государственного уровня
Гео-информационные системы
Системы классификации и кодирования объектов
Смежные системы регионального уровня
ЕИАИС ЭЭ
Система торговых площадок оптового рынка
Единая система ведения НСИ и классификаторов ТЭХ
Система дистанционного обучения в области энергосбережения
АСДУ ТЭХ и Ситуационный центр в сфере энергообеспечения и энергосбережения
Система управления проектами в области энергосбережения
Информационно-аналитическая система мониторинга энергосбережения
Оперативный информационный комплекс
Портал энергосбережения
Система сбора и обработки данных в области энергосбережения
Система автоматизации деловых процессов
Система сбора и учета энергоресурсов
Информационно-измерительный комплекс
Системы нижнего уровня
Система расчета и контроля балансов и потерь энергоресурсов
СЭД
КИС
АСТУЭр
АИИС КУЭ
АСКУГ
АСКУТ
АСДУ
Автоматизированные системы субъектов энергосбережения

Проектные решения, изложенные в настоящей работе, апробированы автором и его коллегами в ходе создания программной платформы учета и управления энергообеспечением «Энергиус» [4, 5, 79, 80, 82, 83]. На ее базе были созданы компоненты больших автоматизированных систем диспетчерского управления [2], интеллектуального учета электроэнергии [10], управления энергосбережением (в том числе ЕИАИС ЭЭ) [125] и др. В целях снижения сроков и стоимости работ, были масштабированы и задействованы следующие технологические подходы:

- полный цикл инженерии предметной области ТЭК, в ходе которого был построен большой комплекс моделей, обеспечивающий концептуальный баланс между уровнями АСУТП и АСУП;
- приемы разработки, управляемой моделями (MDE), для автоматической генерации структур данных и программного кода их обработки;
- аспектно-ориентированный подход к проектированию хранения данных и мониторинга состояния объекта, в целях обеспечения трассируемости компонентов к моделям предметной области и устранения дублирования программного кода;
- организация расчета и анализа данных в среде распределенных вычислений, позволившая обеспечить информационную поддержку принятия решений в темпе процесса оперативно-диспетчерского управления.

ЗАКЛЮЧЕНИЕ

В диссертационной работе предложены новые теоретико-категорные модели и методы проектирования больших информационно-управляющих систем, предназначенные для повышения эффективности жизненного цикла таких систем. Основные научные и практические результаты, выводы и рекомендации состоят в следующем:

1. Для задачи управления жизненным циклом больших информационно-управляющих систем с целью уменьшения затрат при соблюдении ограничений на допустимые значения показателей качества, показана возможность решения путем привлечения масштабируемых технологий инженерии предметной области, разработки, управляемой моделями, распределенных вычислений, аспектно-ориентированного подхода.

2. Построен и теоретически обоснован аппарат для формального анализа и синтеза технологий проектирования систем на основе теории категорий и концепции формальной технологии, предоставляющий методы подбора рациональных архитектурных решений, интеграционных интерфейсов, способов трансформации результатов проектирования.

3. Построена и теоретически обоснована формальная технология проектирования вычислительных компонентов информационно-управляющих систем путем отображения алгоритмов на архитектуру вычислительной среды, включая редукцию на конечные множества доступных ресурсов, подбор рациональных средств управления потоками вычислений, рациональное комплексирование вычислительных компонентов в распределенные системы, оценку сложности вычислений с привлечением аппарата конечнозначной логики Лукасевича.

4. Построена и теоретически обоснована универсальная теоретико-категорная семантика расширения модульных технологий проектирования систем аспектно-ориентированными приемами и инструментами, с обеспечением трассирования результатов проектирования к классам задач и разделения ответственности путем экспликации аспектной структуры на уровень модулей.

5. Доказано, что разметка данных и сценариев поведения систем классами задач представляет собой частный случай перехода от модульного проектирования к аспектно-ориентированному и что необходимым и достаточным условием существования аспектного связывания помеченных сценариев является совместимость с конкуррентностью, а разделения ответственности – хорошая упорядоченность аспектов.

6. Доказано, что оснащение компонентов интеграционными интерфейсами, предназначенными для сборки систем, представляет собой частный случай перехода от модульного

проектирования к аспектно-ориентированному, что позволяет применять аспектно-ориентированный подход для повышения эффективности синтеза технологий создания систем.

7. Разработаны модели данных, процессов, вычислений и других составляющих предметной области топливно-энергетического комплекса, на базе которых при помощи предложенных в работе приемов рационально построены большие системы диспетчерского управления, интеллектуального учета электроэнергии, управления энергосбережением.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

АИИС	Автоматизированная информационно-измерительная система
АЛУ	Арифметико-логическое устройство
АОП	Аспектно-ориентированное программирование
АО-модель	Аспектно-ориентированная модель
АО-расширение	Аспектно-ориентированное расширение
АО-технология	Аспектно-ориентированная технология
АСДУ	Автоматизированная система диспетчерского управления
АСКУГ	Автоматизированная система коммерческого учета газа
АСКУТ	Автоматизированная система коммерческого учета тепла
АСКУЭр	Автоматизированная система коммерческого учета энергоресурсов
АСОД	Автоматизированная система обработки данных
АСУ	Автоматизированная система управления
АСУП	Автоматизированная система управления предприятием
АСУТП	Автоматизированная система управления технологическим процессом
АТД	Абстрактный тип данных
БИХ-фильтр	Фильтр с бесконечной импульсной характеристикой
ВАК	Высшая аттестационная комиссия
ГБП	Граница балансовой принадлежности
ЕИАИС	Единая интегрированная автоматизированная информационная система
ЕИАС	Единая информационно-аналитическая система
ЕЭС	Единая энергетическая система России
ЖКХ	Жилищно-коммунальное хозяйство
ИМОУ	Информационная модель объекта управления
ИПУ	Институт проблем управления
КИС	Корпоративная информационная система
КПД	Коэффициент полезного действия
КУЭ	Коммерческий учет электроэнергии
МЗАЛУ	Многозначное арифметико-логическое устройство
МЗИ	Модули защиты информации
НСИ	Нормативно-справочная информация
ОАО	Открытое акционерное общество
ООО	Общество с ограниченной ответственностью

ПМС	Подсистема мониторинга состояния
ПОД	Подсистема обмена документами
ППИ	Подсистема пользовательских интерфейсов
ПРАД	Подсистема расчета и анализа данных
ПСИ	Подсистема сбора информации и телеуправления
РАМ-машина	Равнодоступная адресная машина
РАН	Российская академия наук
РД	Руководящий документ
СТО	Стандарт организации
СУБД	Система управления базами данных
СЭД	Система электронного документооборота
ТУ	Технические условия
ТЭК	Топливно-энергетический комплекс
ТЭХ	Топливно-энергетическое хозяйство
ФЗ	Федеральный закон
ФСТ	Федеральная служба по тарифам
ЭВМ	Электронно-вычислительная машина
ЭЭ	Энергетическая эффективность
AOSD	Aspect-oriented software development
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
CALS	Continuous acquisition and lifecycle support
CASE	Computer-aided software engineering
CIM	Common information model
CMOS	Complementary-symmetry/metal-oxide semiconductor
DSL	Domain-specific language
EJB	Enterprise Java Beans
FODA	Feature-Oriented Domain Analysis
IASTED	International Association of Science and Technology for Development
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
J2EE	Java 2 Enterprise Edition
MathML	Mathematical Markup Language
MDE	Model-driven engineering

ODE	Ontology Domain Engineering
ORM	Object-relational mapping
OWL	Ontology Web Language
PDCA	Plan – Do – Check – Act
PIM	Platform independent model
PSM	Platform specific model
RCT	Requirement Composition Table
RSA	(cryptographical algorithm by) Rivest, Shamir, and Adleman
RSEB	Reuse-driven Software Engineering Business
RUP	Rational Unified Process
SCADA	Supervisory control and data acquisition
SOA	Service-oriented architecture
SQL	Structured Query Language
ULS	Ultra-large scale
UML	Unified Modeling Language
WSDL	Web Service Definition Language
XML	eXtensible Markup Language
XP	eXtreme Programming

СПИСОК ЛИТЕРАТУРЫ

1. *Андреева Л.В., Осика Л.К., Тубинис В.В.* Коммерческий учет электроэнергии на оптовом и розничном рынках. М.: «АВОК-ПРЕСС», 2010.
2. *Андрюшкевич С.К.* Построение информационной модели крупномасштабных объектов технологического управления с применением аспектно-ориентированного подхода // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2010. Т. 8, №3. С. 34–45.
3. *Андрюшкевич С.К., Журавлев С.С., Золотухин Е.П., Ковалёв С.П., Окольнишников В.В., Рудометов С.В.* Разработка системы мониторинга с использованием имитационного моделирования // Проблемы информатики. 2010. №4. С. 65–75.
4. *Андрюшкевич С.К., Золотухин Е.П., Ковалёв С.П.* Многоуровневая автоматизированная система управления производственно-технологическими процессами с управлением затратами на основе мониторинга, анализа и прогноза состояния технологической инфраструктуры нефтегазодобывающего предприятия. Патент на изобретение №2435188, ноябрь 2011.
5. *Андрюшкевич С.К., Золотухин Е.П., Ковалёв С.П.* Экспериментальный образец программного комплекса системы оперативного мониторинга технологической инфраструктуры «ЭО СОМТИ» (ЭО ПК СОМТИ). Свидетельство о государственной регистрации программы для ЭВМ №2010613850, июнь 2010.
6. *Андрюшкевич С.К., Ковалёв С.П.* Динамическое связывание аспектов в крупномасштабных системах технологического управления // Вычислительные технологии. 2011. Т. 16, №6. С. 3–12.
7. *Андрюшкевич С.К., Ковалёв С.П.* Интеллектуальный мониторинг распределенных технологических объектов с использованием информационных моделей состояния // Известия Томского политехнического университета. 2010. Т. 317, №5. Управление, вычислительная техника и информатика. С. 35–39.
8. *Андрюшкевич С.К., Ковалёв С.П.* О формировании аспектно-ориентированного связывания производственных задач в системах технологического управления // Материалы XIII Российской конференции «Распределенные информационные и вычислительные ресурсы» DICR'2010. Новосибирск: ИВТ СО РАН, 2010. С. 32.
9. *Андрюшкевич С.К., Ковалёв С.П.* Опыт адаптации стандартных информационных моделей для распределенных объектов технологического управления // Высокие технологии, фундаментальные исследования, образование: сборник трудов Седьмой международной

научно-практической конференции «Исследование, разработка и применение высоких технологий в промышленности». Т. 1. СПб.: Изд-во Политехнического университета, 2009. С. 56–57.

10. *Андрюшкевич С.К., Ковалёв С.П., Кубышкин А.С., Трегубов А.М.* Проблемы автоматизации управления процессами розничного рынка электроэнергии // Труды XIV Международной конференции «Проблемы управления и моделирования в сложных системах» ПУМСС-2012. Самара: СамНЦ РАН, 2012. С. 376–386.
11. *Арзамасцев Д.А.* АСУ и оптимизация режимов энергосистем. М.: Высшая школа, 1983.
12. *Арестова М.Л., Быковский А.Ю.* Методика реализации оптоэлектронных схем многопараметрической обработки сигналов на основе принципов многозначной логики // Квантовая электроника. 1995. Т. 22, №10. С. 980–984.
13. *Астелс Д., Миллер Г., Новак М.* Практическое руководство по экстремальному программированию. М.: Вильямс, 2002.
14. *Ахо А.В., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.
15. *Байков И.Р., Смородов Е.А., Ахмадуллин К.Р.* Методы анализа надежности и эффективности систем добычи и транспорта углеводородного сырья. М.: ООО «Недра-Бизнесцентр», 2003.
16. *Барахнин В.Б., Клименко О.А., Ковалёв С.П.* Сбор и систематизация информации для портала математических ресурсов MATHTREE // Труды международной конференции «Вычислительные и информационные технологии в науке, технике и образовании». Т. II. Павлодар: ТОО НПФ «ЭКО», 2006. С. 381–389.
17. *Бауэр Ф.Л., Гооз Г.* Информатика. Вводный курс. Т. 1-2. М.: Мир, 1990.
18. *Бениаминов Е.М.* Алгебраические методы в теории баз данных и представлении знаний. М.: Научный мир, 2003.
19. Большой энциклопедический словарь. М.: Изд-во «Большая Российская энциклопедия», 2000.
20. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2-е изд. М.: Бином; СПб.: Невский диалект, 1999.
21. *Важенин А.П.* Параллельные вычисления с динамической длиной операндов: проблемы и перспективы. В кн.: Системная информатика. Вып. 7. Новосибирск: Наука, 2000. С. 225–274.
22. *Васильев С.Н., Жерлов А.К., Федосов Е.А., Федунов Б.Е.* Интеллектное управление динамическими системами. М.: ФИЗМАТЛИТ, 2000.

23. *Вендров А.М.* CASE-технологии. Современные методы и средства проектирования информационных систем. М.: Финансы и статистика, 1998.
24. *Виноградова М.В.* Свойство однозначности построения алгоритма при проектировании мультиаспектных информационных систем // Наука и образование: электронное научно-техническое издание МГТУ им. Н.Э. Баумана. 2011. Вып.10. С. 13.
25. *Воеводин В.В.* Отображение проблем вычислительной математики на архитектуру вычислительных систем // Вычислительные методы и программирование. 2000. Т.1. С. 37–44.
26. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002.
27. *Гамма Э., Хелм Р., Джонсон Р., Влассидес Дж.* Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001.
28. *Глушков В.М.* Введение в АСУ. Киев: Техника, 1972.
29. *Голдблатт Р.* Логика времени и вычислимость. М.: Мир, 1993.
30. *Голдблатт Р.* Топосы. Категорный анализ логики. М.: Мир, 1983.
31. ГОСТ 19.701-90 (ИСО 5807-85). Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. М.: Госстандарт СССР, 1990.
32. ГОСТ Р ИСО 9126-93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению. М.: Госстандарт России, 1994.
33. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. М.: Госстандарт России, 2010.
34. ГОСТ Р 50001-2012. Системы энергетического менеджмента. Требования и руководства по применению. М.: Стандартинформ, 2012.
35. ГОСТ Р 8.596-2002. Метрологическое обеспечение измерительных систем. Основные положения. М.: Госстандарт России, 2002.
36. *Гордов Е.П., Ковалёв С.П., Молородов Ю.И., Федотов А.М.* Web-система управления знаниями об окружающей среде // Вычислительные технологии. Т. 10, ч.2. Специальный выпуск: Труды Международной конференции и школы молодых ученых «Вычислительные и информационные технологии для наук об окружающей среде» (CITES-2005). Новосибирск, 2005. С. 12–19.
37. *Гребенюк Г.Г., Лубков Н.В., Никишов С.М.* Информационные аспекты управления муниципальным хозяйством. М.: ЛЕНАНД, 2011.

38. *Гребенюк Г.Г., Никишов С.М., Лубков Н.В., Крыгин А.А., Середа Л.А.* Городской отраслевой классификатор видов объектов топливно-энергетического хозяйства города Москвы для решения задач управления. ГОК Москвы 001 – 2012. М.: ДепТЭХ, 2012.
39. *Гуськов А.Е., Дубров И.С., Ковалёв С.П., Молородов Ю.И., Прокопов Н.А., Сударикова И.А.* Принципы построения распределённой среды атласа «Атмосферные аэрозоли Сибири» // Вычислительные технологии. Т. 11. Специальный выпуск: Труды X Российской конференции с участием иностранных ученых «Распределенные информационно-вычислительные ресурсы» DICR-2005. Новосибирск, 2005. С. 77–87.
40. *Дал У., Дейкстра Э., Хоор К.* Структурное программирование. М.: Мир, 1975.
41. Данные в языках программирования. М.: Мир, 1982.
42. Дискретная математика и математические вопросы кибернетики. Т. I. М.: Наука, 1974.
43. *Елиферов В.Г., Репин В.В.* Бизнес процессы: регламентация и управление. М.: ИНФРА-М, 2012.
44. *Журков С.В., Пинус А.Г.* Об автоморфизмах шкал потенциалов вычислимости *n*-элементных алгебр // Сибирский математический журнал. 2003. Т. 44, №3. С. 606–621.
45. *Завертайлов А.В., Ковалёв С.П.* Система поддержки деятельности распределённых экспертных групп по разработке онтологий предметных областей // Труды Международной конференции по вычислительной математике «МКВМ-2004». Рабочие совещания. Новосибирск: ИВМиМГ СО РАН, 2004. С. 56–65.
46. *Загоруйко Н.Г., Гусев В.Д., Завертайлов А.В., Ковалёв С.П., Налётов А.М., Саломатина Н.В.* Автоматизация процессов построения онтологий // Proc. XI International Conference “Knowledge-Dialogue-Solution” (KDS-2005). Vol. 1. Varna, Bulgaria, 2005. P. 53–59.
47. *Загоруйко Н.Г., Гусев В.Д., Завертайлов А.В., Ковалёв С.П., Налётов А.М., Саломатина Н.В.* Система «ONTOGRID» для построения онтологий // Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции «Диалог'2005». М.: Наука, 2005. С. 146–152.
48. *Загоруйко Н.Г., Гусев В.Д., Завертайлов А.В., Ковалёв С.П., Налётов А.М., Саломатина Н.В.* Система ONTOGRID для автоматизации процессов построения онтологий предметных областей // Автометрия. 2005. Т. 41, №5. С. 13–25.
49. *Загорулько Ю.А., Ануреев И.А., Загорулько Г.Б.* Подход к разработке системы поддержки принятия решений на примере нефтегазодобывающего предприятия // Известия Томского политехнического университета. 2010. Т. 316, №5. С. 127–131.
50. *Замулин А.В.* Формальные методы спецификации программ. Новосибирск: НГУ, 2002.

51. *Золотарев С.В.* Некоторые особенности реализации стандарта IEC-60870-5-104 в системе программирования контроллеров ISaGRAF: от теории к практике // Информатизация и системы управления в промышленности. 2010. №4(28). С. 26–31.

52. *Золотухин Е.П., Ковалёв С.П. Андрюшкевич С.К., Васильева Е.С., Воробьева Д.Б., Долгих Е.В., Кислицына Т.А., Лапшинов М.Ф., Окольнишников В.В., Романов Р.А., Шульженко Л.М.* Разработка интеллектуальной системы пространственно-технологического мониторинга на базе глобального спутникового позиционирования с целью повышения энергоэффективности и экологической безопасности существующих методов добычи углеводородов. Отчет о НИР по государственному контракту № 02.514.11.4126 от 30.09.2010 г. Минобрнауки РФ. № гос. рег. 1200962854. Новосибирск: КТИ ВТ СО РАН, 2010.

53. *Камер Д.* Сети TCP/IP. Том 1. Принципы, протоколы и структура. М.: «Вильямс», 2003.

54. *Карпенко А.С.* Логики Лукасевича и простые числа. М.: Наука, 2000.

55. *Карпова Т.С.* Базы данных: модели, разработка, реализация. СПб.: Питер, 2001.

56. *Кейслер Г., Чэн Ч.Ч.* Теория моделей. М.: Мир, 1977.

57. *Коберн А.* Современные методы описания функциональных требований. М.: «Лори», 2002.

58. *Кобец Б., Волкова И.* Smart Grid // Энергорынок. 2010. №3(75). С. 66–72.

59. *Ковалёв С.П.* Алгебраические спецификации в экстремальном программировании // Материалы Международной конференции «Алгебра, логика и кибернетика-2004». Иркутск: ИГПУ, 2004. С. 155–157.

60. *Ковалёв С.П.* Алгебраический подход к проектированию распределенных вычислительных систем // Сибирский журнал индустриальной математики. 2007. Т. X, №2(30). С. 70–84.

61. *Ковалёв С.П.* Аналитические модели машинной арифметики // Сибирский журнал индустриальной математики. 2003. Т. 6, №3(15). С. 88–102.

62. *Ковалёв С.П.* Архитектура времени в распределенных информационных системах // Вычислительные технологии. 2002. Т. 7, №6. С. 38–53.

63. *Ковалёв С.П.* Аспектное проектирование интегрированных систем управления энергопотреблением // Сборник материалов II международной научно-технической конференции «Новые информационные технологии в нефтегазовой отрасли и образовании». Тюмень: ТюмГНГУ, 2006. С. 97–99.

64. *Ковалёв С.П.* Аспектно-ориентированный подход к проектированию систем мониторинга крупномасштабных объектов // Проблемы информатики. 2009. №3(4). С. 5–18.

65. *Ковалёв С.П.* Диаграммное описание комплексирования программных систем // Вестник Новосибирского государственного университета. Серия: Математика, механика, информатика. 2012. Т. 12, №3. С. 103–126.

66. *Ковалёв С.П.* Категория вычислительных систем // Международная конференция «Алгебра и логика: теория и приложения». Тезисы докладов. Красноярск: СФУ, 2013. С. 64–66.

67. *Ковалёв С.П.* Логика Лукасевича как архитектурная модель арифметики // Сибирский журнал индустриальной математики. 2003. Т. 6, №4(16). С. 32–50.

68. *Ковалёв С.П.* Математические основания компьютерной арифметики // Математические труды. 2005. Т. 8, №1. С. 3–42.

69. *Ковалёв С.П.* Полупримальные модели компьютерных вычислений // Международная конференция «Алгебра и ее приложения». Тезисы докладов. Красноярск: ИВМ СО РАН, 2007. С. 73–74.

70. *Ковалёв С.П.* Применение аспектно-ориентированного подхода для автоматизации крупномасштабных объектов и процессов управления // Труды VI Международной конференции «Управление развитием крупномасштабных систем» MLSD'2012. Т. II. М.: ИПУ РАН, 2012. С. 314–323.

71. *Ковалёв С.П.* Применение логики Лукасевича для разработки алгоритмов. В кн.: Логические исследования. Вып. 12. М.: Наука, 2005. С. 194–206.

72. *Ковалёв С.П.* Применение онтологий при разработке распределенных автоматизированных информационно-измерительных систем // Автометрия. 2008. Т. 44, №2. С. 41–49.

73. *Ковалёв С.П.* Применение формальных методов для обеспечения качества вычислительных систем // Вестник Новосибирского государственного университета. Серия: Математика, механика, информатика. 2004. Т. 4, №2. С. 49–74.

74. *Ковалёв С.П.* Принципы профессионального программирования // 3 Международная конференция «Смирновские чтения». М.: ИФРАН, 2001. С. 42–44.

75. *Ковалёв С.П.* Семантика аспектно-ориентированного моделирования данных и процессов // Информатика и ее применения. 2013. Т. 7, Вып.3. С. 70–80.

76. *Ковалёв С.П.* Системный анализ жизненного цикла больших информационно-управляющих систем // Автоматика и телемеханика. 2013. №9. С. 98–118.

77. *Ковалёв С.П.* Формальный подход к аспектно-ориентированному моделированию сценариев // Сибирский журнал индустриальной математики. 2010. Т. 13, №3. С. 30–42.

78. *Ковалёв С.П.* Формальный подход к разработке программных систем. Учебное пособие. Новосибирск: НГУ, 2004.

79. *Ковалёв С.П., Андрюшкевич С.К.* Автоматизированная система ведения нормативно-справочной информации в области энергетики «Энергиус-НСИ». Свидетельство о государственной регистрации программы для ЭВМ №2011613800, май 2011.

80. *Ковалёв С.П., Андрюшкевич С.К.* Автоматизированная система учета энергоносителей «Энергиус-Учёт». Свидетельство о государственной регистрации программы для ЭВМ №2010613308, май 2010.

81. *Ковалёв С.П., Андрюшкевич С.К.* Автоматизированный инструмент компонентной интеграции информационных порталов // Труды Всероссийской научной конференции «Научный сервис в сети Интернет». М.: МГУ, 2004. С. 98–99.

82. *Ковалёв С.П., Андрюшкевич С.К., Гуськов А.Е.* Интеграционная платформа учета и управления энергообеспечением «Энергиус». Свидетельство о государственной регистрации программы для ЭВМ №2009613359, июнь 2009.

83. *Ковалёв С.П., Андрюшкевич С.К., Яковченко К.Н.* Система автоматизации оперативно-диспетчерского документооборота «Энергиус-Диспетчер». Свидетельство о государственной регистрации программы для ЭВМ №2009611387, март 2009.

84. *Ковалёв С.П., Пароджанов С.С.* Концепция создания автоматизированной системы мониторинга и управления энергоэффективностью на объектах города Москвы // Информационно-измерительные и управляющие системы. 2011. Т. 9, №6. С. 50–58.

85. *Ковалёв С.П., Прокопов Н.А.* Автоматизация процессов измерения физико-химических величин на базе онтологии // Вычислительные технологии. 2007. Т. 12, Специальный выпуск 1. С. 79–86.

86. *Ковалёв С.П., Яковченко К.Н.* Организация информационных порталов на основе канальной интеграции // Труды Международной конференции по вычислительной математике «МКВМ-2004». Рабочие совещания. Новосибирск: ИВМиМГ СО РАН, 2004. С. 66–72.

87. *Комков В.А., Тимахова Н.С.* Энергосбережение в жилищно-коммунальном хозяйстве. М.: ИНФРА-М, 2010.

88. *Кублашвили О.В.* Документационное обеспечение управления. М.: Изд-во МГУП, 2004.

89. *Кубышкин А.С.* Разработка модели разграничения прав доступа для автоматизированных систем технологического управления // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2012. Т. 10, №3. С. 26–33.

90. *Кузнецов А.А., Ковалёв С.П.* Мониторинг состояния крупномасштабных систем технологического управления // Высокие технологии, фундаментальные исследования, образование: сборник трудов Седьмой международной научно-практической конференции

«Исследование, разработка и применение высоких технологий в промышленности». Т. 1. СПб.: Изд-во Политехнического университета, 2009. С. 101–103.

91. *Кузнецов А.А., Ковалёв С.П.* Тестирование и мониторинг в распределенных автоматизированных системах технологического управления // Вычислительные технологии. 2009. Т. 14, №4. С. 57–69.
92. *Леоненков А.* Самоучитель UML. СПб.: БХВ-Петербург, 2002.
93. *Маклейн С.* Категории для работающего математика. М.: Физматлит, 2004.
94. *Мамиконов А.Г.* Методы разработки автоматизированных систем управления. М.: Энергия, 1973.
95. *Мамиконов А.Г., Кульба В.В., Косяченко С.А.* Типизация разработки модульных систем обработки данных. М.: Наука, 1989.
96. *Матросов В.М., Анапольский Л.Ю., Васильев С.Н.* Метод сравнения в математической теории систем. Новосибирск: Наука, 1980.
97. *Мельников П.П.* Обмен данными в разнородных бизнес-приложениях. М.: Финакадемия, 2010.
98. *Месарович М., Такахара Я.* Общая теория систем: математические основы. М.: Мир, 1978.
99. *Михеев А., Орлов М.* Перспективы workflow-систем. Сравнение workflow-языков // PC Week/RE. 2005. Т. 36. С. 498.
100. МОЭК – Московская объединенная энергетическая компания. http://www.oaomoek.ru/ru/.
101. *Непейвода Н.Н.* Уровни знаний и умений // Труды научно-исследовательского семинара Логического центра ИФ РАН. Вып. XIV. М.: ИФ РАН, 2000. С. 9–35.
102. *Никаноров С.П.* Теоретико-системные конструкты для концептуального анализа и проектирования. М.: «Концепт», 2008.
103. *Новоселов Д.Ю., Ковалёв С.П.* Аспектно-ориентированный подход к созданию GRID приложений // Труды Международной конференции по вычислительной математике «МКВМ-2004». Рабочие совещания. Новосибирск: ИВМиМГ СО РАН, 2004. С. 95–102.
104. *Новоселов Д.Ю., Ковалёв С.П.* LDAP-каталог СО РАН // Вычислительные технологии. 2005. Т. 10. Специальный выпуск: Труды IX рабочего совещания по электронным публикациям El-Pub2004. С. 102–107.
105. *Норенков И.П., Кузьмик П.К.* Информационная поддержка наукоемких изделий (CALS-технологии). М.: Изд-во МГТУ, 2002.
106. *Оптнер С.* Системный анализ для решения деловых и промышленных проблем. М.: Советское радио, 1966.

107. *Осика Л.К., Макаренко И.Г.* Промышленные потребители на рынке электроэнергии. Принципы организации деловых отношений. М.: ЭНАС, 2010.
108. *Пароджанов С.С., Ковалёв С.П.* Система комплексного управления проектами в области энергосбережения // Научная сессия НИЯУ МИФИ-2012. Аннотации докладов. Т. 2. Проблемы фундаментальной науки. Стратегические информационные технологии. М.: НИЯУ МИФИ, 2012. С. 354.
109. *Пинус А.Г.* Условные термы и их применение в алгебре и теории вычислений. Новосибирск: Изд-во НГТУ, 2002.
110. *Плоткин Б.И.* Универсальная алгебра, алгебраическая логика и базы данных. М.: Наука, 1991.
111. *Пьявченко Т.А., Финаев В.И.* Автоматизированные информационно-управляющие системы. Таганрог: Изд-во ТРТУ, 2007.
112. *Раскин Д.* Интерфейс: новые направления в проектировании компьютерных систем. СПб.: Символ-Плюс, 2003.
113. *Рокотян С.С., Шапиро И.М.* Справочник по проектированию электроэнергетических систем. М.: Энергия, 1985.
114. *Рябко Б.Я., Фионов Л.Н.* Основы современной криптографии. М.: Научный мир, 2004.
115. *Савчук В.Л.* Электронные средства сбора, обработки и отображения информации. Томск: Томский государственный университет систем управления и радиоэлектроники, 2007.
116. *Скотт Д.С.* Области в денотационной семантике. В кн.: Математическая логика в программировании. М.: Мир, 1991. С. 58–118.
117. *Смелянский Р.Л.* Методы анализа и оценки производительности вычислительных систем. М.: МГУ, 1990.
118. *Солонина А.И., Улахович Д.А., Яковлев Л.А.* Алгоритмы и процессоры цифровой обработки сигналов. СПб.: БХВ-Петербург, 2012.
119. *Соренсен И.* Язык спецификаций. В кн.: Требования и спецификации в разработке программ. М.: Мир, 1984. С. 223–239.
120. *Страуструп Б.* Введение в язык C++. 1995. http://citforum.ru/programming/cpp/aglav.shtml.
121. *Таненбаум Э.* Архитектура компьютера. 4-е изд. СПб.: Питер, 2002.
122. *Таненбаум Э., ван Стеен М.* Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003.
123. *Тарский А.* Введение в логику и методологию дедуктивных наук. Биробиджан: ИП «ТРИВИУМ», 2000.

124. *Трегубов А.М., Ковалёв С.П.* Особенности проектирования систем мониторинга и управления энергосбережением // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2012. Т. 10, №3. С. 80–91.

125. *Трегубов А.М., Ковалёв С.П.* Системы мониторинга и управления энергосбережением – новый класс крупномасштабных систем // Сборник статей XI Международной научно-практической конференции «Фундаментальные и прикладные исследования, разработка и применение высоких технологий в промышленности». Т. 1. СПб.: Изд-во Политехнического университета, 2011. С. 132.

126. *Туманов В.Е.* Основы проектирования реляционных баз данных. М.: Бином, 2010.

127. *Хоар Ч.* Взаимодействующие последовательные процессы. М.: Мир, 1989.

128. *Хусаинов А.А., Лопаткин В.Е., Трещев И.А.* Исследование математической модели параллельных вычислительных процессов методами алгебраической топологии // Сибирский журнал индустриальной математики. 2008. Т. 11, №1(33). С. 141–152.

129. *Цимбал А.А., Аншина М.Л.* Технологии создания распределенных систем. СПб.: Питер, 2003.

130. *Чарнецки К., Айзенекер У.* Порождающее программирование: методы, инструменты, применение. СПб.: Питер, 2005.

131. *Юркевич Е.Н.* Логика определения в контексте мереологии понятий // Ученые записки Таврического национального университета им. В.И. Вернадского. Серия «Философия. Культурология. Политология. Социология». 2011. Т. 24(63), № 3–4. С. 418–425.

132. *Юэ Ж., Оппен Д.* Равенства и правила переписывания. Обзор. В кн.: Математическая логика в программировании. М.: Мир, 1991. С. 176–232.

133. *Яблонский С.В., Гаврилов Г.П., Набебин А.А.* Предполные классы в многозначных логиках. М.: МЭИ, 1997.

134. *Якобсон А., Буч Г., Рамбо Дж.* Унифицированный процесс разработки программного обеспечения. СПб.: Питер, 2002.

135. *van der Aalst W.M.P.* Challenges in business process management: verification of business processes using Petri nets // Bulletin EATCS. 2003. N 80. P. 174–198.

136. *Adámek J., Herrlich H., Strecker G.* Abstract and Concrete Categories. N. Y.: Wiley and Sons, 1990.

137. *Adams B., De Schutter K., Zaidman A., Demeyer S., Tromp H., De Meuter W.* Using aspect orientation in legacy environments for reverse engineering using dynamic analysis – an industrial experience report // Journal of Systems and Software. 2009. Vol. 82(4). P. 668–684.

138. *Aizenbud-Reshef N., Nolan B., Rubin J., Shaham-Gafni Y.* Model traceability // IBM Systems Journal. 2006. Vol. 45(3). P. 515–526.

139. *Allen R. J., Garlan D.* A formal basis for architectural connection // ACM Transactions on Software Engineering and Methodology. 1997. Vol. 6(3). P. 213–249.

140. *Andrews J.H.* Process-algebraic foundations of aspect-oriented programming // Lecture Notes in Computer Science. 2001. V. 2192. P. 187–209.

141. *Andryushkevich S.K., Kovalyov S.P.* Distributed plants intelligent monitoring using information models of states // Proc. IASTED International Conference on Automation, Control and Information Technology ACIT-2010. Control, Diagnostics, and Automation. Novosibirsk, 2010. P. 250–255.

142. *Anvaari M., Cruzes D.S., Conradi R.* Smart Grid software applications as an ultra-large-scale system: challenges for evolution // Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES. 2012. P. 1–6.

143. *Appelrath H.-J., Uslar V., Lucks A., Schmedes N., Winkels L.* Interaction of EMS related systems by using the CIM standard // Proc. 2nd International ICSC Symposium on Information Technologies in Environmental Engineering. Magdeburg: Otto-von-Guericke-Universität, 2005. P. 596–610.

144. Aspect-Oriented Software Development. Reading: Addison Wesley, 2004.

145. *Barr M., Wells C.* Category Theory for Computing Science. London: Prentice Hall, 1990.

146. *Batory D.S., Azanza M., Saraiva J.* The objects and arrows of computational design // Lecture Notes in Computer Science. 2008. V. 5301. P. 1–20.

147. *Beavers G.* Automated theorem proving for Łukasiewicz logics // Studia Logica. 1993. Vol. 52, No. 2. P. 183–195.

148. *Bednarczyk M.A., Borzyszkowski A.M., Pawlowski W.* Epimorphic functors. Gdansk: Institute of Computer Science, 2007. http://www.ipipan.gda.pl/~andrzej/papers/categ-ipi.ps.gz.

149. *Belov S.D., Bredikhin S.V., Kovalyov S.P., Kulagin S.A., Musher S.L., Nikultsev V.A., Scherbakova N.G., Shabalnikov I.V., Shokin Yu.I.* Siberia: Internet is coming // Proc. 7th Joint European Networking Conference JENC7. Budapest, 1996. P. 173/1–173/4.

150. *Belov S.D., Bredikhin S.V., Kovalyov S.P., Kulagin S.A., Musher S.L., Nikultsev V.A., Scherbakova N.G., Shabalnikov I.V., Shokin Yu.I.* Siberia: putting the virgin lands to the Internet plough // Proc. 8th Joint European Networking Conference JENC8. Edinburg, 1997. P. 523/1–523/4.

151. *Belov S.D., Bredikhin S.V., Kovalyov S.P., Kulagin S.A., Musher S.L., Scherbakova N.G., Shabalnikov I.V.* The emerging Internet landscape in Siberia // Computer Networks. 1998. Vol. 30. P. 1657–1662.

152. *van den Berg K.G., Tekinerdogan B., Nguyen H.* Analysis of crosscutting in model transformations // Proc. ECMDA-TW Traceability Workshop. Bilbao, 2006. P. 51–64.

153. *Bergman C., Berman J.* Morita equivalence of almost-primal clones // Journal of Pure and Applied Algebra. 1996. Vol. 108. P. 175–201.

154. *Bergmans L., Aksit M.* Principles and design rationale of composition filters. In: Aspect-Oriented Software Development. Reading: Addison Wesley, 2004. P. 63–96.

155. *Bertoni A., Mauri G., Miglioli P.* On the power of model theory to specify abstract data types and to capture their recursiveness // Fundamenta Informaticae. 1983. Vol. IV.2. P. 129–170.

156. *Botella P., Burgués X., Franch X., Huerta M., Salazar G.* Modeling non-functional requirements. In: Applying Requirements Engineering. Sevilla: Catedral Publicaciones, 2001. P. 13–34.

157. *Breitman K.K., Leite J.C.S.P., Finkelstein A.* The world's a stage: a survey on requirements engineering using a real-life case study // Journal of the Brazilian Computer Society. 1999. Vol. 6(1). P. 13–37.

158. *Brichau J., Chitchyan R., Rashid A., D'Hondt T.* Aspect-oriented software development: an introduction. In: Wiley Encyclopedia of Computer Science and Engineering. Vol. 1. N. Y.: Wiley & Sons, 2008. P. 188–198.

159. *Carboni A., Janelidze G., Kelly G.M., Paré R.* On localization and stabilization for factorization systems // Applied Categorical Structures. 1997. Vol. 5. P. 1–58.

160. *Chai Y., Zhou Y., Wang Y.* A collaborative strategy for manufacturing execution systems // Proc. 11th International Conference on Computer Supported Cooperative Work in Design CSCWD'2007. Melbourne, 2007. P. 904–907.

161. *Charfi A., Mezini M.* AO4BPEL: An aspect-oriented extension to BPEL // World Wide Web. 2007. No. 10. P. 309–344.

162. *Charfi A., Müller H., Mezini M.* Aspect-oriented business process modeling with AO4BPMN // Lecture Notes in Computer Science. 2010. V. 6138. P. 48–61.

163. *Chernak Y.* Building a foundation for structured requirements. Aspect-oriented requirement engineering explained // Better Software. 2009. Vol. 99, №1. P. 90–96.

164. CIDR Report. http://www.cidr-report.org/as2.0/.

165. *Clarke S., Baniassad E.* Aspect-Oriented Analysis and Design: The Theme Approach. Reading: Addison Wesley, 2005.

166. *Colyer A., Clement A., Harley G., Webster M.* Eclipse AspectJ. Reading: Addison-Wesley, 2004.

167. *Dadam P., Reichert M., Kuhn K.* Clinical workflows – the killer application for process-oriented information systems? // Proc. 4th International Conference on Business Information Systems BIS'2000. Poznań, 2000. P. 36–59.

168. *Deng G., Gray J., Schmidt D., Lin Y., Gokhale A., Lenz G.* Evolution in model-driven software product-line architectures. In: Software Applications: Concepts, Methodologies, Tools, and Applications. Hershey: IGI Global, 2009. P. 1280–1312.

169. *Diskin Z., Maibaum T.S.E.* Category theory and model-driven engineering: from formal semantics to design patterns and beyond // Proc. 7th Workshop ACCAT'2012. Electronic Proceedings in Theoretical Computer Science. 2012. Vol. 93. P. 1–21.

170. *Douence R., Fradet P., Südholt M.* Trace-based aspects. In: Aspect-Oriented Software Development. Reading: Addison Wesley, 2004. P. 201–218.

171. *Egyed A., Grünbacher P., Heindl M., Biffl S.* Value-based requirements traceability: lessons learned // Lecture Notes in Business Information Processing. 2009. Vol. 14. P. 240–257.

172. Energy Management Systems. Rijeka Shanghai N. Y.: INTECH, 2011.

173. *Falbo R.A., Guizzardi G., Duarte K.C.* An ontological approach to domain engineering // Proc. 14th International Conference on Software Engineering and Knowledge Engineering SEKE'2002. Ischia, Italy, 2002. P. 351–358.

174. *Fiadeiro J.L.* Categories for Software Engineering. Berlin Heidelberg N. Y.: Springer, 2005.

175. *Fiadeiro J.L., Lopes A., Wermelinger M.* A mathematical semantics for architectural connectors // Lecture Notes in Computer Science. 2003. V. 2793. P. 190–234.

176. *Filman R.E., Friedman D.P.* Aspect-oriented programming is quantification and obliviousness. In: Aspect-Oriented Software Development. Reading: Addison Wesley, 2004. P. 21–36.

177. *Foster A.L., Pixley A.F.* Semi-categorical algebras I. Semi-primal algebras // Mathematische Zeitschrift. 1964. Vol. 83, N 2. P. 147–169.

178. *Frankel D.S.* Model Driven Architecture: Applying MDA to Enterprise Computing. N. Y.: Wiley & Sons, 2003.

179. *Glabbeek R.J. van, Goltz U.* Refinement of actions and equivalence notions for concurrent systems // Acta Informatica. 2000. V. 37, N 4–5. P. 229–327.

180. *Goguen J.* A categorical manifesto // Mathematical Structures in Computer Science. 1991. Vol. 1(1). P. 49–67.

181. *Goguen J.* Categorical foundations for general systems theory. In: Advances in Cybernetics and Systems Research. London: Transcripta Books, 1973. P. 121–130.

182. *Goguen J., Burstall R.* Institutions: abstract model theory for specification and programming // Journal of the ACM. 1992. Vol. 39(1). P. 95–146.

183. Grid Computing: Making the Global Infrastructure a Reality. N. Y.: Wiley & Sons, 2003.

184. *Groher I., Völter M.* Aspect-oriented model-driven software product line engineering // Lecture Notes in Computer Science. 2009. V. 5560. P. 111–152.

185. *Gruber T.R.* Toward principles for the design of ontologies used for knowledge sharing // International Journal of Human-Computer Studies. 1995. Vol. 43. P. 907–928.

186. *Guitart R., van den Bril L.* Décompositions et lax-complétions // Cahiers de Topologie et Géométrie Différentielle Catégoriques. 1977. Tome 18, no 4. P. 333–407.

187. *Gurevich Y.* Evolving algebras 1993: Lipari guide. In: Specification and Validation Methods. Oxford: Oxford University Press, 1995. P. 9–36.

188. *Hagihara S., Arai T., Shimakawa M., Yonezaki N.* Developing embedded systems from formal specifications written in temporal logic // Lecture Notes in Electrical Engineering. 2013. Vol. 150. P. 107–113.

189. *Hailpern B., Tarr P.* Model-driven development: the good, the bad, and the ugly // IBM Systems Journal. 2006. Vol. 45(3). P. 451–461.

190. *Halbwachs N., Caspi P., Raymond P., Pilaud D.* The synchronous data flow programming language LUSTRE // Proceedings of IEEE. 1991. Vol. 79. P. 1305–1320.

191. *Hanenberg S., Unland R.* Roles and aspects: similarities, differences, and synergetic potential // Lecture Notes in Computer Science. 2002. V. 2425. P. 507–520.

192. *Healy M.J., Caudell T.P.* Ontologies and worlds in category theory: implications for neural systems // Axiomathes. 2006. Vol. 16. P. 165–214.

193. *Hermosillo G., Seinturier L., Duchien L.* Using complex event processing for dynamic business process adaptation // Proc. 7th IEEE International Conference on Services Computing SCC'2010. Miami, 2010. P. 466–473.

194. *Herrmann C., Krahn H., Rumpe B., Schindler M., Volkel S.* Scaling-up model-based-development for large heterogeneous systems with compositional modeling // Proc. International Conference on Software Engineering Research & Practice SERP'2009. Las Vegas, 2009. P. 172–176.

195. *Hey T., Trefethen A.* The data deluge: an e-science perspective. In: Grid Computing: Making the Global Infrastructure a Reality. N. Y.: Wiley & Sons, 2003. P. 809–824.

196. *Hruška T., Kolenčík P.* Comparison of categorical foundations of object-oriented database model // Lecture Notes in Computer Science. 1997. V. 1341. P. 302–319.

197. IEC 61970-301 «Energy management system application program interface (EMS-API) – Part 301: Common information model (CIM) base». Geneva: IEC, 2011.

198. ISC Domain Survey. Internet Systems Consortium, 2012. http://www.isc.org/solutions/survey.

199. ISO/IEC/IEEE 42010 «Systems and software engineering – Architecture description». Geneva: ISO, New York: IEEE, 2011.

200. *Jacobs B., Rutten J.* A tutorial on (co)algebras and (co)induction // EATCS Bulletin. 1997. Vol. 62. P. 222–259.

201. *Jacobson I., Griss M., Jonsson P.* Software Reuse: Architecture, Process, and Organization for Business Success. Reading: Addison-Wesley, 1997.

202. *Jagadeesan R., Pitcher C., Riely J.* Open bisimulation for aspects // Proc. International Conference AOSD'2007. Vancouver, 2007. P. 107–120.

203. *Jouault F., Vanhooff B., Bruneliere H., Doux G., Berbers Y., Bezivin J.* Inter-DSL coordination support by combining megamodeling and model weaving // Proc. 2010 ACM Symposium on Applied Computing. Sierre, 2010. P. 2011–2018.

204. *Kahan W.* Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic. Berkeley: 1996. http://www.cs.berkeley.edu/~wkahan/ieee754status/IEEE754.pdf.

205. *Kannenberg A., Saiedian H.* Why software requirements traceability remains a challenge // Journal of Defense Software Engineering. July/August 2009. P. 14–19.

206. *Katz E., Katz S.* Verifying scenario-based aspect specifications // Lecture Notes in Computer Science. 2005. V. 3582. P. 432–447.

207. *Khurshid N., Ormandjieva O., Klasa S.* Towards a tool support for specifying complex software systems by Categorical Modeling Language // Studies in Computational Intelligence. 2010. Vol. 296. P. 133–149.

208. *Kiczales G., Lamping J., Mendhekar A., Maeda C., Lopes C.V., Loingtier J.-M., Irwin J.* Aspect-oriented programming // Lecture Notes in Computer Science. 1997. V. 1241. P. 220–242.

209. *Kiczales G., Mezini M.* Separation of concerns with procedures, annotations, advice and pointcuts // Lecture Notes in Computer Science. 2005. V. 3586. P. 195–213.

210. *Ko R.K.L., Lee E.W., Lee S.G.* Business-OWL (BOWL) – a hierarchical task network ontology for dynamic business process decomposition and formulation // IEEE Transactions on Services Computing. 2012. Vol. 5(2). P. 246–259.

211. *Kolovos D.S., Paige R.F., Polack F.A.C.* The grand challenge of scalability for model driven engineering // Lecture Notes in Computer Science. 2009. V. 5421. P. 48–53.

212. *Kovalyov S.P.* Architecture of distributed information-computing system for exploring atmospheric aerosol // Proc. SPIE. 2005. V. 6160. Part I: International Conference “Molecular spectroscopy and atmospheric radiative processes”. P. 21–26.

213. *Kovalyov S.P.* Model-theoretic methods of analysis of computer arithmetic // Proc. 9th Asian Logic Conference “Mathematical Logic on Asia”. Singapore: World Scientific Publishing Co., 2006. P. 145–155.

214. *Kovalyov S.P.* Modeling aspects by category theory // Proc. 9th Workshop on Foundations of Aspect-Oriented Languages. Rennes, France, 2010. P. 63–68.

215. *Kovalyov S.P., Molorodov Yu.I.* Data integration methodology for atlas “Atmospheric aerosols of Siberia” // Proc. SPIE. 2005. V. 6160. Part I: International Conference “Molecular spectroscopy and atmospheric radiative processes”. P. 9–14.

216. *Kovalyov S.P., Ozhiganova E.A.* Formal modeling of software quality with xNoFun language // Proc. 2nd IASTED International Conference on Automation, Control and Information Technology ACIT-2005. Software Engineering. Novosibirsk, 2005. P. 48–53.

217. KSL Protégé Project. Stanford University. http://protege.stanford.edu.

218. *Lambek J., Scott P.J.* Introduction to Higher Order Categorical Logic. Cambridge: Cambridge University Press, 1986.

219. *Le D. T. M., Janicki R.* A categorical approach to mereology and its application to modelling software components // Lecture Notes in Computer Science. 2008. V. 5084. P. 146–174.

220. *Liskov B.H., Zilles S.N.* Specification techniques for data abstractions // IEEE Transactions on Software Engineering. 1975. SE-1, 1. P. 7–19.

221. *Lopes A., Fiadeiro J.L.* Revisiting the categorical approach to systems // Lecture Notes in Computer Science. 2002. V. 2422. P. 426–440.

222. *Luckham D., Vera J., Bryan D., Augustin L., Belz F.* Partial orderings of event sets and their application to prototyping concurrent, timed systems // Journal of Systems and Software. 1993. Vol. 21(3). P. 253–265.

223. Mathematical Markup Language (MathML) version 2.0. W3 Consortium, 2003. http://www.w3.org/TR/MathML2/.

224. *Mills J.W.* Polymer Processors. Indiana University, Computer Science Dept, Technical Report TR580. Indiana University, 2003. http://www.cs.indiana.edu/pub/techreports/TR580.pdf.

225. *Mohagheghi P., Dehlen V.* Where is the proof? – a review of experiences from applying MDE in industry // Lecture Notes in Computer Science. 2008. V. 5095. P. 432–443.

226. *Morin B., Barais O., Jézéquel J.M.* Weaving aspect configurations for managing system variability // Proc. 2nd International Workshop on Variability Modelling of Software-Intensive Systems VaMoS'08. Essen, 2008. P. 53–62.

227. *Morin B., Barais O., Nain G., Jézéquel J.-M.* Taming Dynamically Adaptive Systems using models and aspects // Proc. 31st International Conference on Software Engineering ICSE'09. Vancouver, 2009. P. 122–132.

228. *Murch R.* Autonomic computing. N. Y.: IBM Software press, 2004.

229. *Nakajima S., Tamai T.* Weaving in role-based aspect-oriented design models // Proc. Workshop on Early Aspects EA'2004. Vancouver, 2004. http://trese.cs.utwente.nl/workshops/oopsla-early-aspects-2004/Papers/NakajimaEtAl.pdf

230. *Onneweer S.P., Kerkhoff H.G.* High-radix current-mode CMOS circuits based on the truncated-difference operator // Proc. 17th International Symposium on Multiple-Valued Logic. Boston: IEEE, 1987. P. 188–195.

231. OWL Web Ontology Language guide. W3 Consortium, 2003. http://www.w3.org/TR/2003/WD-owl-guide-20030331/.

232. *Paige R.F., Drivalos N., Kolovos D.S., Fernandes K.J., Power C., Olsen G.K., Zschaler S.* Rigorous identification and encoding of trace-links in model-driven engineering // Software and Systems Modeling. 2011. Vol. 10. P. 469–487.

233. *Pfalzgraf J., Soboll T.* On a general notion of transformation for multiagent systems and its implementation // Electronic Communication of the European Association of Software Science and Technology. 2008. Vol. 12.

234. *Pinto M., Fuentes L., Troya J.M.* DAOP-ADL: an architecture description language for dynamic component and aspect-based development // Lecture Notes in Computer Science. 2003. V. 2830. P. 118–137.

235. *Pohl C., Charfi A., Gilani W., Gobel S., Grammel B., Lochmann H., Rummler A., Spriestersbach A.* Adopting aspect-oriented software development in business application engineering // Proc. International Conference AOSD'2008. Industry track. Brussels, Belgium, 2008. http://www.aosd.net/2008/program/industry.php.

236. *Pratt V.R.* Modeling concurrency with partial orders // International Journal of Parallel Programming. 1986. V. 15(1). P. 33–71.

237. *Qureshi N.A., Jureta I., Perini A.* Requirements engineering for self-adaptive systems: core ontology and problem statement // Lecture Notes in Business Information Processing. 2011. V. 83. P. 33–47.

238. *Rashid A., Chitchyan R.* Aspect-oriented requirements engineering: a roadmap // Proc. 13th International Workshop on Early Aspects EA'2008. Leipzig, 2008. P. 35–41.

239. *Rashid A., Moreira A.* Domain models are not aspect free // Lecture Notes in Computer Science. 2006. V. 4199. P. 155–169.

240. *Rashid A., Sawyer P., Moreira A., Araújo J.* Early aspects: a model for aspect-oriented requirements engineering // Proc. International Conference RE'2002. Essen, 2002. P. 199–202.

241. *Raynal M., Singhal M.* Logical time: capturing causality in distributed systems // IEEE Computer. 1996. V. 29(2). P. 49–56.

242. *Rosenberg I.G.* Completeness properties of multiple-valued logic algebras. In: Computer Science and Multiple-Valued Logic. Amsterdam: North Holland, 1977. P. 144–186.

243. *Rutle A., Rossini A., Lamo Y., Wolter U.* A formalisation of the copy-modify-merge approach to version control in MDE // Journal of Logic and Algebraic Programming. 2010. Vol. 79, Issue 7. P. 636–658.

244. *Sannella D.* A survey of formal software development methods. In: Software Engineering: A European Prospective. IEEE Computer Society Press, 1993. P. 281–297.

245. *Sassone V., Nielsen M., Winskell G.* A classification of models for concurrency // Lecture Notes in Computer Science. 1993. V. 715. P. 82–96.

246. *Sassone V., Nielsen M., Winskell G.* Deterministic behavioural models for concurrency // Lecture Notes in Computer Science. 1993. V. 711. P. 682–692.

247. *Schauerhuber A., Schwinger W., Kapsammer E., Retschitzegger W., Wimmer M.* Towards a common reference architecture for aspect-oriented modeling // Proc. 8th International Workshop on Aspect-Oriented Modeling. Bonn, 2006. http://publik.tuwien.ac.at/files/pub-inf_3843.pdf.

248. *Schmidt D.C.* Model-driven engineering // IEEE Computer. 2006. Vol. 39(2). P. 25–32.

249. *Smith D.R.* Aspects as invariants. In: Automatic Program Development. Berlin Heidelberg N. Y.: Springer, 2008. P. 247–263.

250. *Smith D.R.* Composition by colimit and formal software development // Lecture Notes in Computer Science. 2006. V. 4060. P. 317–332.

251. *Spivey J. M.* The Z Notation: a Reference Manual. London: Prentice Hall, 1992.

252. *Srinivas Y.V., Jüllig R.* SPECWARE: formal support for composing software // Lecture Notes in Computer Science. 1995. Vol. 947. P. 399–422.

253. *Steimann F.* Domain models are aspect free // Lecture Notes in Computer Science. 2005. V. 3713. P. 171–185.

254. *Steimann F.* The paradoxical success of aspect-oriented programming // Proc. International Conference OOPSLA'2006. Portland, 2006. P. 481–497.

255. *Sutton S.M., Rouvellou I.* Concern modeling for aspect-oriented software development. In: Aspect-Oriented Software Development. Reading: Addison Wesley, 2004. P. 479–505.

256. *Textor A., Stynes J., Kroeger R.* Transformation of the Common Information Model to OWL // Lecture Notes in Computer Science. 2010. V. 6385. P. 163–174.

257. *Tudorache T., Noy N.F., Tu S., Musen M.A.* Supporting collaborative ontology development in Protégé // Lecture Notes in Computer Science. 2008. V. 5318. P. 17–32.

258. Ultra-Large-Scale Systems: The Software Challenge of the Future. Pittsburgh: Carnegie Mellon Software Engineering Institute, 2006.

259. *Vogel T., Seibel A., Giese H.* The role of models and megamodels at runtime // Lecture Notes in Computer Science. 2011. V. 6627. P. 224–238.

260. *Whittle J., Jayaraman P.* MATA: A tool for aspect-oriented modeling based on graph transformation // Lecture Notes in Computer Science. 2008. V. 5002. P. 16–27.

261. *Wimmer M., Schauerhuber A., Kappel G., Retschitzegger W., Schwinger W., Kapsammer E.* A survey on UML-based aspect-oriented design modeling // Journal ACM Computing Surveys. 2011. V. 43, No. 4. P. 28:1–28:33.

262. *Wolfengagen V.E.* Object-oriented solutions // Proc. 2nd International Workshop on Advances in Databases and Information systems ADBIS'95. Moscow, 1995. P. 117–128.

263. *Yu Y., Niu N., González-Baixauli B., Mylopoulos J., Easterbrook S., Leite J.* Requirements engineering and aspects // Lecture Notes in Business Information Processing. 2009. V. 14. P. 432–452.

264. *Zhang J., Cottenier T., van den Berg A., Grey J.* Aspect composition in the Motorola aspect-oriented modeling weaver // Journal of Object Technology. 2007. Vol. 6, No. 7. P. 89–108.

Приложение 1. АКТЫ ВНЕДРЕНИЯ РЕЗУЛЬТАТОВ ДИССЕРТАЦИОННОЙ РАБОТЫ

Получены акты внедрения научных и практических результатов диссертационной работы при создании следующих информационно-управляющих систем регионального и федерального масштаба.

1. Автоматизированная информационно-измерительная система коммерческого учета электроэнергии ООО «Транснефтьсервис-С» для предприятий ОАО «АК «Транснефть» (АИИС КУЭ ТНС, г. Москва, 2007).
2. Автоматизированная информационно-измерительная система коммерческого учета электроэнергии ОАО «Томусинское энергоуправление», (АИИС КУЭ ТЭУ, г. Междуреченск, 2008).
3. Автоматизированная система диспетчерского управления энергохозяйством ООО «Газпром энерго» (АСДУ ГПЭ, г. Москва, 2009).
4. Единая интегрированная автоматизированная информационная система мониторинга и управления эффективностью энергосбережения на объектах города Москвы (ЕИАИС ЭЭ, г. Москва, 2011).

«УТВЕРЖДАЮ»
Первый заместитель генерального директора
ООО «Транснефтьсервис С»
Семенов В.Н.

«29» января 2007 г.

АКТ
о внедрении научных и практических результатов
докторской диссертационной работы
Ковалёва С.П.

Комиссия в составе:

- председатель: Шихин Владимир Анатольевич, заместитель главного инженера ООО «Транснефтьсервис С», к.т.н.

- член комиссии: Фадеев Евгений Александрович, начальник отдела программного обеспечения ООО «Транснефтьсервис С»

- член комиссии: Ковалёв Сергей Протасович, соискатель, к.ф.-м.н.

составили настоящий акт о том, что результаты диссертационной работы Ковалёва С.П., представленной на соискание ученой степени доктора физико-математических наук, использованы ООО «Транснефтьсервис С» при разработке специального программного обеспечения (СПО) автоматизированной информационно-измерительной системы коммерческого учета электроэнергии ООО «Транснефтьсервис С» для предприятий ОАО «АК «Транснефть».

Объектами внедрения являются:

1. Архитектура СПО, состав и функциональное наполнение подсистем.
2. Формальная модель качества СПО, соответствующая стандарту ISO/IEC 9126.
3. Принципы комплексирования распределенных компонентов СПО в единую систему, протоколы и расписания взаимодействия между компонентами.
4. Схема непрерывного мониторинга функционирования системы на базе реактивной модели сбора и обработки событий.
5. Модель распределенной вычислительной среды для расчета учетных показателей энергопотребления в составе СПО.

Использование указанных результатов позволяет:
- реализовать непрерывный автоматический 30-минутный цикл сбора и анализа величин энергопотребления предприятий ОАО «АК «Транснефть», включающих около 1300 точек учета, распределенных по 49 регионам РФ;
- сократить время формирования регламентных отчетов об энергопотреблении согласно требованиям оптового рынка электроэнергии;
- повысить достоверность значений энергопотребления.

Председатель комиссии _____ Шихин В.А.

Член комиссии _____ Фадеев Е.А.

Член комиссии _____ Ковалёв С.П.

«УТВЕРЖДАЮ»
Директор
ОАО «Томусинское энергоуправление»
Шерстобитов В.Н.

«07» февраля 2008 г.

АКТ
о внедрении научных и практических результатов
докторской диссертационной работы
Ковалёва С.П.

Комиссия в составе:

- председатель: Муратов Михаил Васильевич, заместитель главного инженера ОАО «Томусинское энергоуправление».

- член комиссии: Легков Сергей Анатольевич, ведущий специалист по АИИС ОАО «Томусинское энергоуправление».

- член комиссии: Ковалёв Сергей Протасович, соискатель, к.ф.-м.н.

составили настоящий акт о том, что результаты диссертационной работы Ковалёва С.П., представленной на соискание ученой степени доктора физико-математических наук, использованы ОАО «Томусинское энергоуправление» при разработке специального программного обеспечения (СПО) автоматизированной информационно-измерительной системы коммерческого учета электроэнергии ОАО «Томусинское энергоуправление».

Объектами внедрения являются:

1. Концепция архитектуры СПО, состав и функциональное наполнение подсистем.
2. Формальная модель качества СПО, соответствующая стандарту ISO/IEC 9126.
3. Принципы организации распределенных компонентов СПО в единую систему, протоколы и расписания взаимодействия между компонентами.
4. Проект подсистемы непрерывного сквозного мониторинга функционирования системы на базе реактивной модели сбора и обработки событий.
5. Схема распределенной вычислительной среды для расчета учетных показателей энергопотребления в составе СПО.

Использование указанных результатов позволяет:
- реализовать непрерывный автоматический 30-минутный цикл сбора и анализа величин энергопотребления предприятий, подключенных к подстанциям ОАО «Томусинское энергоуправление», включающих около 100 точек коммерческого и технического учета, расположенных в Кемеровской области;
- сократить время формирования регламентных отчетов об энергопотреблении согласно требованиям оптового рынка электроэнергии;
- повысить достоверность значений энергопотребления

Председатель комиссии _____ Муратов М.В.

Член комиссии _____ Легков С.А.

Член комиссии _____ Ковалёв С.П.

«УТВЕРЖДАЮ»
Председатель комиссии
Заместитель генерального директора
ООО «Газпром энерго» по автоматизации

_____ А.В. Парамзин
«29» _____ 2009 г.

АКТ
о внедрении научных и практических результатов
докторской диссертационной работы
Ковалёва С.П.

Комиссия в составе:

- председатель: Парамзин Алексей Викторович, заместитель генерального директора по автоматизации ООО «Газпром энерго»

- член комиссии: Михасенко Антон Геннадьевич, ведущий инженер отдела автоматизации и телемеханизации управления АСУ ТП ООО «Газпром энерго»

- член комиссии: Ковалёв Сергей Протасович, соискатель, к.ф.-м.н.

составили настоящий акт о том, что результаты диссертационной работы Ковалёва С.П., представленной на соискание ученой степени доктора физико-математических наук, использованы ООО «Газпром энерго» при внедрении подсистем автоматизированной системы диспетчерского управления (АСДУ) энергохозяйством.

Объектами внедрения являются:

1. Архитектура комплекса подсистем АСДУ, состав и функциональное наполнение подсистем.
2. Схема оперативно-диспетчерского документооборота на базе реактивной модели сбора и обработки событий, отражающих согласование и исполнение оперативных заявок на проведение ремонтных работ.

Использование указанных результатов позволяет повысить надежность энергоснабжения объектов Единой системы газоснабжения РФ за счет следующих факторов:
- оперативное и достоверное планирование и контроль ремонтных работ оборудования, установленного на более 400 энергообъектов в 43 регионах РФ;
- оперативное оповещение диспетчеров Администрации ООО «Газпром энерго» о переключениях на энергообъектах;
- автоматическая запись диспетчерских телефонных переговоров в 10 филиалах ООО «Газпром энерго» в реальном времени и репликация в центральный архив записей.

Член комиссии _____ Михасенко А.Г.

Член комиссии _____ Ковалёв С.П.

«УТВЕРЖДАЮ»
Заместитель руководителя Департамента
топливно-энергетического хозяйства города Москвы
Татарников А.В.
«22» 04 2014 г.

АКТ
о внедрении научных и практических результатов
докторской диссертационной работы
Ковалёва С.П.

Комиссия в составе:
- председатель: Родзик Юрий Владимирович, советник по ИТ Департамента топливно-энергетического хозяйства города Москвы
- член комиссии: Черкасова Наталья Леонидовна, заведующая сектором ГБУ «Энергетика»
- член комиссии: Ковалёв Сергей Протасович, соискатель

составили настоящий акт о том, что результаты диссертационной работы Ковалёва С.П., представленной на соискание ученой степени доктора физико-математических наук, использованы Департаментом топливно-энергетического хозяйства города Москвы при разработке Единой интегрированной автоматизированной информационной системы мониторинга и управления эффективностью энергосбережения на объектах города Москвы (ЕИАИС ЭЭ).

Объектами внедрения являются:

1. Концептуальная модель предметной области «мониторинг энергосбережения».
2. Аспектно-ориентированная архитектура ЕИАИС ЭЭ, обеспечивающая распределение функциональных задач между компонентами системы, трассируемое к модели предметной области.
3. Модель распределенной вычислительной среды для расчета показателей энергопотребления и энергообеспечения объектов в составе ЕИАИС ЭЭ.

Использование указанных результатов позволяет:
- реализовать региональную политику в области энергосбережения и повышения энергоэффективности, включая мониторинг выполнения программ энергосбережения;
- собирать, хранить, верифицировать информацию о транспортировке и потреблении энергетических ресурсов объектов бюджетной сферы города Москвы;
- обеспечить ведение единой НСИ в области энергосбережения города Москвы;
- повысить эффективность управления топливно-энергетическим хозяйством и сократить потери энергоресурсов города Москвы.

Председатель комиссии _____ Родзик Ю.В.
Член комиссии _____ Черкасова Н.Л.
Член комиссии _____ Ковалёв С.П.