Аспектно-ориентированный подход к проектированию систем мониторинга крупномасштабных объектов

С. П. Ковалёв

Конструкторско-технологический институт вычислительной техники СО РАН, 630090, Новосибирск, Россия

Представлена теоретико-категорная метамодель аспектно-ориентированного подхода (АОП), описывающая обогащение формальных моделей программ трассируемыми трансформациями, порождающими их интеграционные интерфейсы. В качестве приложения метамодели рассмотрена формализация аспектно-ориентированных моделей сценариев сбора и обработки событий в виде помеченных частично упорядоченных множеств. Предложен формальный подход к анализу эффективности вычислительных аспектов систем мониторинга, основанный на аппарате полупримальных алгебр и конечнозначных логик. Описана типовая аспектная архитектура системы мониторинга, разработанная в рамках технологического цикла инженерии предметной области "мониторинг крупномасштабных объектов".

Ключевые слова: аспектно-ориентированный подход, формальная дисциплина проектирования, полупримальная алгебра, инженерия предметной области, система мониторинга, управление энергообеспечением.

A category-theoretical metamodel of aspect-oriented programming (AOP) is presented, which describes enrichment of program models by traceable refinements that produce their interfaces. Formalization of aspect-oriented event gathering and processing scenarios, as labeled partially ordered sets, is considered as an application of the metamodel. Formal approach to efficiency analysis of computational aspects of monitoring systems, utilizing semi-primal algebras and finite-valued logics, is proposed. Reference aspect architecture of a monitoring system, developed in the course of the domain engineering cycle applied to large-scale objects monitoring, is described.

Key words: aspect-oriented approach, architecture school, semi-primal algebra, domain engineering, monitoring system, energy management.

Введение. Основой эффективного управления технологическими и природными объектами является мониторинг – своевременное получение достоверной информации об их состоянии в форме, пригодной для принятия решений о воздействии на них. Для крупномасштабных объектов объем информации настолько велик, что ее сбор и обработка невозможны без сквозной автоматизации. Процессы мониторинга должны согласованно выполняться на множестве объектов и центров управления, часто удаленных друг от друга на тысячи километров, оснащенных разнородными техническими средствами, контролируемых различными юридическими лицами. Поэтому традиционное разделение задач автоматизации по единицам организационной структуры, типам технических средств и т. п. не позволяет создавать целостные системы мониторинга. В инженерии программного обеспечения данная проблема известна как спутанность классов задач (tangling of concerns). Для ее решения в конце 1990-х гг. была предложена новая методология – аспектно-ориентированный подход к разработке программного обеспечения (АОП) [1]. Целью АОП является явное разделение и комплексирование аспектов – логически замкнутых и технологически однородных единиц функциональности, реализующих конкретные классы задач. В качестве аспектов выступают средства реализации как функциональных возможностей (например, сбор данных, их хранение, вычислительная обработка и т.д.), так и нефункциональных требований (например, координация функционирования распределенных компонентов, защита информации и т. д.) Разделение аспектов и представление их в виде отдельных единиц комплексирования, связываемых в явно заданных точках, значительно упрощает компоновку и сопровождение системы. Многие технологии программирования специально ориентированы на реализацию отдельных типов аспектов, поэтому аспектная декомпозиция облегчает организацию деятельности коллектива разработчиков, являющихся узкими специалистами.

Однако практическое применение АОП сдерживается ввиду отсутствия единого непротиворечивого понимания способа достижения его цели и взаимосвязи с традиционным модульным подходом к проектированию систем.

Эту ситуацию может улучшить формальная метамодель, отражающая процесс обогащения технологий модульного проектирования средствами разработки аспектов. Такая метамодель может быть основана на концепции трассируемости шагов разработки системы, поскольку именно трассируемость более всего страдает от спутывания классов задач. Для формализации и верификации метамодели целесообразно использовать аппарат теории категорий, позволяющий единообразно описывать разнородные технологии проектирования систем. Заметим, что в литературе предлагается ряд подходов к формализации АОП (см., например, работы [2, 3] и др.), однако они представлены в терминах конкретных формализмов и поэтому могут применяться только в рамках конкретных парадигм проектирования.

Основным примером приложения АОП к задачам разработки систем мониторинга является моделирование сценариев сбора и обработки событий. Разделение сценария на аспекты приводит к разметке его шагов идентификаторами классов задач, для решения которых он выполняется. Разнообразные помеченные структуры часто фигурируют в литературе в качестве формальных моделей распределенных систем (см., например, [4]), однако природа меток и способы их синтеза обычно не рассматриваются.

К числу основных аспектов систем мониторинга относятся вычислительные задачи анализа данных. От результатов их проектирования зависит эффективность всей системы. Они реализуются на базе распределенных вычислительных сред, узлы которых поддерживают различные модели вычислений. Последние формально описываются как конечные алгебры, обладающие средствами управления потоком вычислений и диагностики переполнения. Процедуры сборки распределенных вычислительных систем формализуются при помощи теории категорий. Для анализа эффективности вычислительных алгоритмов привлекается аппарат дискретной математики и конечнозначной логики [5].

Вместе с интерфейсными и инфраструктурными вычислительные аспекты образуют типовую архитектуру системы мониторинга. Ввиду высокой сложности крупномасштабных объектов мониторинга для ее построения необходимо проводить специальный технологический цикл исследования и моделирования предметной области (Domain Engineering) [6]. В ходе него строится онтология предметной области "мониторинг крупномасштабных объектов", служащая источником терминов для других моделей и в дальнейшем трансформируемая в схему базы данных системы. Применение АОП в этом цикле позволяет обеспечить быструю инкрементальную неразрушающую разработку проблемно-ориентированных систем с высокой степенью многократного использования результатов. Примером системы, реализованной в рамках этого подхода, является интеграционная платформа учета и управления энергообеспечением "Энергиус".

Настоящая работа представляет собой краткий обзор результатов применения АОП к задачам разработки крупномасштабных систем мониторинга. Частично эти результаты подробно изложены в работах [7−12]; ряд статей по тематике работы находится в печати. Сведения из теории категорий, необходимые для понимания работы, можно найти, например, в [13], а описание основных принципов АОП – в работе [1].

1. Теоретико-категорное описание аспектно-ориентированного подхода. Формальный подход, представленный в настоящей работе, основан на теоретико-категорной модели процесса разработки систем, предложенной в работе [14]. В рамках этой модели каждому компоненту или системе сопоставляется абстрактный объект, а каждому действию по интеграции индивидуального компонента в систему – морфизм, т. е. абстрактный аналог функции, отображающий объект-область (компонент) в объект-кообласть (систему). Композиция морфизмов соответствует объединению действий в цепочки (конвейеры). Имеются тождественные морфизмы, означающие "ничегонеделание". Таким образом, получается категория, обозначаемая как *c-DESC*.

Наборы компонентов, из которых строятся системы, и взаимосвязи между ними представляются диаграммами этой категории. Акту сборки системы из набора Δ отвечает построение копредела – диаграммы в форме коконуса с основанием Δ, обладающего свойством универсальности [13]. Объект копредела обозначает систему, а ребра копредельного коконуса – вхождения компонентов. В простейшем случае, когда компоненты никак не взаимосвязаны, система представляет собой их прямую сумму, явным образом сохраняющую идентичность каждого компонента и не добавляющую ничего лишнего. Диаграммы, копределы которых действительно отвечают актам сборки систем, называются конфигурациями и образуют класс, обозначаемый как *Conf*.

Известно, что интеграционные возможности компонента определяются его интерфейсом – специально подготовленной частью. Примером интерфейса служит сигнатура программного модуля – список функций, реализованных в нем, с указанием параметров и возвращаемых значений. Интерфейсы образуют категорию, она обозначается как *SIG*, а операция выделения интерфейса формализуется как функтор $sig : c{-}DESC \to SIG$, называемый сигнатурным. Поскольку различные 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. Отсюда вытекает, что функтор *sig* должен быть инъективным на каждом множестве вида Mor (*A*, *B*), состоящем из всех морфизмов с областью *A* и кообластью *B*; такие функторы называются унивалентными (faithful).

Для каждого интерфейса гарантируется наличие хотя бы одной реализации, поддерживающей его интеграционные возможности в полном объеме. Она называется дискретной и определяется функтором $sig^* : SIG \to c\text{-}DESC$. Интерфейс дискретной реализации любого *SIG*-объекта *I* должен совпадать с *I*, поэтому функтор sig^* является правым обратным к *sig*, т. е. $sig \circ sig^* = 1_{SIG}$. 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^* является сопряженным слева к *sig* с тождественной единицей сопряжения [13].

Существует естественная взаимосвязь между интерфейсами и конфигурациями. Если две *c-DESC*-диаграммы имеют один и тот же *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 может содержать как конфигурации, так и наборы, не порождающие целостных систем. Кроме того, выделение интерфейса должно быть естественным относительно интеграции систем (т. е. перестановочным с ней) в том смысле, что копредел *sig*-образа любой конфигурации должен быть *sig*-образом ее копредела. Заметим, что из этого и других указанных выше требований, предъявляемых к функтору *sig*, вытекает естественность в обратную сторону: *sig* сохраняет копределы конфигураций.

Полноценный процесс создания систем в дополнение к актам интеграции содержит трансформации (refinements) – шаги разработки индивидуальных компонентов (такие, как реализация заданной спецификации на языке программирования). Классу всех трансформаций сопоставляется категория, обозначаемая *r-DESC* и обладающая теми же объектами, что и *c-DESC*, но другими морфизмами. Частным (тривиальным) случаем трансформации является *c-DESC*-изоморфизм. На класс *Conf* налагается условие, что любой набор трансформаций компонентов системы должен порождать трансформацию системы как целого. Четверка $\langle c\text{-}DESC, Conf, sig, r\text{-}DESC\rangle$ называется формальной дисциплиной проектирования (architecture school [14]). Она позволяет формализовать весь спектр отношений вида "часть-целое" (*c-DESC*-морфизмы) и "абстрактное-конкретное" (*r-DESC*-морфизмы), возникающих между единицами комплексирования программ.

Примеры формальных дисциплин можно найти в работах [7, 14]. Для целей настоящей работы представляют интерес дисциплины "над" **Set**, в которых в качестве *sig* выступает категория **Set**, состоящая из всех множеств и всех отображений. Формальными моделями в такой дисциплине служат множества, на которых задана некоторая структура (например, алгебраические системы, топологические пространства и т. п.), а действиями по интеграции – отображения множеств, совместимые со структурой (в подходящем смысле). *sig* является каноническим стирающим функтором, который "забывает" структуру, его действие на объект *A* обычно обозначается как $|A|$. Дискретная реализация интерфейса заключается в формировании структуры, наименьшей (тривиальной) с точки зрения интеграции. Частные случаи дисциплин над **Set** описаны ниже.

В контексте концепции дисциплины проектирования автором настоящей работы предложен теоретико-категорный подход к формализации АОП. Он заключается в том, чтобы представить добавление поддержки аспектов в технологию разработки программных систем как преобразование дисциплин проектирования. Действительно, в целом АОП можно рассматривать как оснащение формальных моделей систем разметкой, идентифицирующей классы задач, на решение которых направлены их составляющие. Изначальная проблема, решить которую

стремились создатели АОП, состояла в отсутствии конструкций, позволяющих разделять спутанный исходный код программ по классам задач, в традиционных языках программирования [15]. Различные технологии АОП различаются по способам разметки, но сходятся в стремлении обеспечить сквозную трассируемость – возможность точно определить, для чего в систему включен тот или иной фрагмент [1].

Известно, что наиболее часто трассируемость нарушается при трансформациях. Возможность трассирования результата трансформации к источнику означает, что обращение ее направления (теоретико-категорная дуализация) превращает ее в отображение результата в источник, т. е. в *c-DESC*-морфизм (назовем его трассой). Можно сказать, что результат трассируемой трансформации должен быть не только конкретизацией, но и частью источника. В то же время, для того чтобы трассируемость сохранялась при последующей интеграции результата в систему, на уровне интерфейсов трасса должна быть обратима справа. Правый обратный к ее *sig*-образу отвечает включению интерфейса источника трансформации в интерфейс результата, идентифицирующему его при интеграции в следующем смысле. Если имеется трансформация источника X в результат A, дуальная к трассе $t : A \to X$, и *SIG*-морфизм $s : sig(X) \to sig(A)$, такой что $sig(t) \circ s = 1_{sig(X)}$, то для любого действия $f : A \to S$ по интеграции A в систему S *SIG*-морфизм $sig(f) \circ s$ задает вхождение $sig(X)$ в $sig(S)$. В связи с этим *sig*-образы трасс называются разметками. Например, в дисциплине над **Set** разметки являются сюръективными отображениями, поэтому действие трассируемой трансформации X в A можно описать как раскрытие (expansion) точек множества $|X|$ в множества, образующие разбиение множества $|A|$, с частичным переносом структуры объекта X на них. Точку множества $|X|$ можно рассматривать как обозначение класса задач, реализуемого путем раскрытия, в соответствии с интуитивным пониманием трансформации. Это наводит на мысль, что цели АОП можно добиться путем "запоминания" трассируемых трансформаций, порождающих программы, хотя бы на уровне интерфейсов. Таким образом, аспектно-ориентированной моделью является пара вида $\langle A, l : sig(A) \to L\rangle$, где A – *c-DESC*-объект, а l – разметка его интерфейса. 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 пара $\langle f : A_1 \to A_2, b : L_1 \to L_2\rangle$, такая что $b \circ l_1 = l_2 \circ sig(f)$. В результате получается категория, обозначаемая *AO* и являющаяся частным случаем универсальной конструкции категории запятой (comma category [13]). В дисциплине над **Set** разметка l сопоставляет каждой точке множества $|A|$ элемент множества L, обозначающий аспект, к которому она относится. *AO*-объект с одноэлементным множеством L естественным образом называется аспектом. По существу (с точностью до изоморфизма), разметка является отношением эквивалентности на множестве $|A|$. *AO*-морфизмом является в точности любой *c-DESC*-морфизм, сохраняющий это отношение эквивалентности.

Диаграммы из класса *Conf*, построение копределов которых не разрушает аспектную структуру, порождают класс *AO*-конфигураций, а трассы – класс (трассируемых) трансформаций *AO*-объектов. В роли интерфейсов могут выступать объекты различных категорий, из которых состоят *AO*-объекты. В частности, можно показать, что сигнатурными могут служить следующие функторы:

− *mod*, переводящий *AO*-объект $\langle A, l\rangle$ в A (модульный интерфейс *AO*-объекта),

− $int = sig \circ mod$ (исходный интерфейс),

− *asp*, переводящий $\langle A, l\rangle$ в l (аспектный интерфейс).

Таким способом формальные дисциплины традиционного (модульного) проектирования обогащаются до полноценных аспектно-ориентированных дисциплин. Методы анализа и синтеза программ, предлагаемые в рамках АОП, задаются в них как универсальные конструкции категории *AO*. Например, операция сплетения (weaving) *AO*-объекта B (базы) с *AO*-объектом W формализуется следующим образом. Правила сплетения имеют вид указания точек соединения (join points) в базе B, в которых происходит обращение к W через соответствующие точки входа (entry points). Например, программа, написанная на аспектно-ориентированном расширении объектно-ориентированного языка (таком как AspectJ [16]), может быть сплетена с другой до или после вызова метода, обработчика исключительных ситуаций и т. д. Для специфицирования этих правил привлекается дополнительный *AO*-объект C, называемый связкой [17], такой что соответствие точек соединения точкам входа приобретает вид пары *AO*-морфизмов $j : B \leftarrow C \to W : e$. При сплетении сначала (виртуально) порождается достаточное количество копий объекта W, по числу точек соединения, с маркировкой соответствующих им точек входа. Сплетение состоит в склеивании этих точек друг с другом таким образом, чтобы не разрушить аспектную структуру исходных объек-

тов. Как легко убедиться на примере дисциплины над **Set**, этим действиям соответствует построение кодекартова квадрата (т. е. копределa) диаграммы $j : B \leftarrow C \rightarrow C \times W : \langle 1_C, e\rangle$. Корректное сплетение получается, если этот копредел существует (в частности, существует произведение $C \times W$) и сохраняется функтором аспектной структуры *str*, переводящим *AO*-объект $\langle A, l\rangle$ в *SIG*-объект codom *l*.

Другой важной аспектно-ориентированной операцией является экспликация аспектной структуры *AO*-объекта – подъем трансформации, породившей его интерфейс, на уровень категории *r-DESC*. А именно, *r-DESC*-морфизм называется экспликацией *AO*-объекта $\langle A, l\rangle$, если он дуален такой трассе $t : A \rightarrow X$, что $sig(t) = l$. Экспликация называется универсальной, если для любого *AO*-морфизма $\langle f, b\rangle : \langle A, l\rangle \rightarrow \langle A', l'\rangle$ и любой экспликации *r* объекта $\langle A', l'\rangle$ существует такой *c-DESC*-морфизм *q* (называемый экспликацией морфизма $\langle f, b\rangle$ вдоль *r*), что $q \circ t = t' \circ f$, где $t' : A' \rightarrow X'$ – трасса, дуальная к *r* (так что $sig(q) = b$). Универсальная экспликация определена однозначно (с точностью до изоморфизма). Ее существование является необходимым условием для разделения классов задач между единицами модульной архитектуры (separation of concerns), стремление к которому явилось главным стимулом к созданию АОП. С ее помощью можно извлекать из *AO*-объекта отдельные аспекты, используя теоретико-категорную конструкцию декартова квадрата для обобщения теоретико-множественной операции вычисления полного прообраза подмножества. Если оба объекта $\langle A, l\rangle$ и $\langle A', l'\rangle$ имеют универсальные экспликации, дуальные к трассам *t* и *t'* соответственно, то *AO*-морфизм $\langle f, b\rangle : \langle A, l\rangle \rightarrow \langle A', l'\rangle$ называется включением подмодели при выполнении следующих условий:

− его экспликация *q* является правой обратной к некоторой трассе;

− коммутативный квадрат, задаваемый условием экспликации $q \circ t = t' \circ f$, является декартовым квадратом в *c-DESC* (т. е. пределом пары морфизмов (q, t'), обладающих общей кообластью).

Из данных условий следует, что $\langle f, b\rangle$ является регулярным *AO*-мономорфизмом. В дисциплине над **Set**, если объект $\langle A, l\rangle$ является аспектом, то $sig(q)$ идентифицирует некоторую метку (элемент множества codom *l'*), а *A* является прообразом этой метки относительно отображения *t'*. Если прообраз каждой метки можно представить как включение подмодели, то объект $\langle A', l'\rangle$ оказывается полностью разделенным на отдельные аспекты.

2. Аспектно-ориентированное моделирование сценариев. В качестве основного примера приложения приведенной формальной метамодели АОП рассмотрим моделирование сценариев поведения систем – одну из основных процедур разработки требований к системам мониторинга. Базовым математическим представлением сценария является частично упорядоченное множество (poset) [18]. Его элементы отвечают атомарным событиям, составляющим ход его выполнения. Частичный порядок естественным образом возникает из причинно-следственных зависимостей между событиями. Действиями по интеграции сценариев компонентов в сценарии системы являются в точности все гомоморфизмы, поскольку ни события, ни взаимодействия не могут быть "забыты". 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, достаточные для построения мощных структурных правил комбинирования моделей. Интерфейсом сценария естественным образом выступает множество его событий, получаемое путем "забывания" их упорядочения. Трансформация сценария заключается в замене атомарных событий подсценариями с полным наследованием порядка [19]. Таким образом, получается формальная дисциплина проектирования $\langle$**Pos**, *CPos*, $|\!-\!|$, *r-Pos*$\rangle$, где **Pos** – категория всех частично упорядоченных множеств и всех их гомоморфизмов; *CPos* – класс всех раздельных объединений **Pos**-коконусов; $|\!-\!| : \mathbf{Pos} \rightarrow \mathbf{Set}$ – канонический функтор, забывающий порядок; *r-Pos* – категория, объектами которой являются все частично упорядоченные множества, а морфизмами – все дуальные к сюръективным отображениям, удовлетворяющим условию $\forall x\ \forall y\ f(x) \le f(y) \Leftrightarrow (x \le y \vee f(x) = f(y))$. Заметим, что все трассы в этой дисциплине обратимы справа. Более того, они позволяют трассировать включения сценариев – действия по интеграции, не разрушающие их внутреннюю структуру. Включениям отвечают в точности все регулярные **Pos**-мономорфизмы, а их трассированию вдоль трансформации, дуальной к трассе $t : A \rightarrow X$ – возможность продолжить любое включение $m : M \rightarrow X$ до включения $m' : M \rightarrow A$, такого что $t \circ m' = m$.

Как и в любой дисциплине над **Set**, аспекты сценария представляют собой не что иное, как метки, приписанные событиям и превращающие его в частично упорядоченное мультимножество (pomset) [18]. Помеченные сценарии хорошо поддаются аспектно-ориентированным операциям, однако их трудно строить средствами модульной сборки. Например, сценарий, состоящий из двух линейно упорядоченных аспектов, выполняющихся в перемежающемся режиме (interleaving), не имеет экспликации аспектной структуры. Это иллюстрирует трудности, возникающие при создании даже простой клиент-серверной распределенной системы. Тем не менее любой (непомеченный) сценарий можно разметить линейно упорядоченными аспектами таким образом, чтобы он полностью разделялся на них. Среди таких разметок существует порождающая как максимальное количество аспектов (каждому событию присваивается уникальная метка), так и минимальное (строится так называемое отношение линейной эквисравнимости [8]). Этим обусловлена разработка аспектно-ориентированных расширений традиционных языков программирования, способных описывать только последовательно исполняемые программы.

Обычно на этапе сбора требований к системам идентифицируются только основные сценарии поведения, состоящие из крупных шагов. Существуют методики, позволяющие определить достаточность состава и глубины детализации сценариев [20]. Тем не менее можно рассмотреть ситуацию, когда удалось каким-либо способом (например, аксиоматически) описать семейство всех допустимых сценариев поведения системы. В этом случае его можно преобразовать в формальную модель ее архитектуры – помеченную систему переходов, порождающую это семейство. Такое преобразование может быть описано на языке теории категорий [4]. Можно сформулировать следующие необходимые и достаточные условия того, что система переходов, порождаемая семейством аспектно-ориентированных сценариев *SF*, является детерминированной:

− каждый аспект каждого сценария из *SF* линейно упорядочен;

− если для некоторых $A, A' \in SF$ существует такая пара AO-морфизмов $f : A \leftarrow B \rightarrow A' : f'$, что $int(f)$ и $int(f')$ являются биекциями, $str(f) = str(f') = 1_{str(B)}$, и каждый аспект сценария B линейно упорядочен, то $A = A'$.

В общих чертах применение аспектно-ориентированного моделирования сценариев к разработке систем мониторинга выглядит следующим образом. Основной сценарий мониторинга состоит в повторяющемся выполнении следующей цепи классов задач по обработке событий: *регистрация* → *сохранение* → *анализ* → *рекомендация*. По мере разработки системы эти классы трансформируются в сложные аспекты, оставаясь отдельными единицами модульной архитектуры. Однако инфраструктурные аспекты, такие как модель объекта мониторинга и пользовательский интерфейс, сплетаются с ними, приводя к утрате разделимости классов задач. Для того чтобы выполнять различные стадии основного цикла на разных аппаратных узлах, необходимо реплицировать инфраструктуру между ними. Такая репликация вызывает основные трудности при разработке и эксплуатации крупномасштабных систем мониторинга.

3. Алгебраический подход к анализу эффективности вычислительных аспектов. Эффективность систем мониторинга в значительной степени определяется способностью выполнять анализ данных в темпе их поступления. Хотя длительность одного цикла мониторинга, как правило, достаточно велика (от нескольких минут до нескольких суток), большое количество источников данных о состоянии крупномасштабного объекта (10^3−10^5 ед.) приводит к достаточно жестким требованиям по производительности средств обработки данных и объему ресурсов для их хранения. Поэтому аспекты анализа данных реализуются в среде распределенных вычислений, в том числе с динамическим развертыванием на базе технологий Grid. Однако изначально алгоритмы анализа описываются в терминах абстрактных типов данных (числа, списки и т. д.), образующих бесконечные множества и не предполагающих распределенного доступа. Для обеспечения эффективности необходима специальная процедура их отображения на вычислительную среду, включающая редукцию на конечные множества доступных ресурсов, построение конфигураций сборки вычислительных компонентов в распределенные системы, оценку и минимизацию сложности (количества обращений к доступным вычислительным операциям). Для формализации этих процедур следует применять алгебраический аппарат. Здесь основным понятием является 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. Поскольку процесс вычисления сводится к суперпозиции примитивов, операциями, возможность выполнения которых предоставляет модель **A**, являются в точности все термальные операции алгебры **A**, т. е. интерпретации термов ее сигнатуры. Совокупность всех термальных операций алгебры называется ее клоном. Алгебра **A** называется полной (primal), если любая функция на $|\mathbf{A}|$ содержится в ее клоне. Функционально (слабо) полной называется алгебра **A**, обогащение сигнатуры которой константами, представляющими все элементы множества $|\mathbf{A}|$, порождает полную алгебру. При моделировании вычислительных систем общего назначения только функционально полные модели вычислений представляют практический интерес.

Одним из способов формального синтеза моделей вычислений является метод частичной интерпретации теории первого порядка T сигнатуры σ, предложенный в [9]. Он состоит в явном отборе из T формул, построение которых не выводит за рамки набора констант, описывающего доступные ресурсы хранения данных. Формально пусть σ_0 – конечная подсигнатура сигнатуры σ; σ_0-проекцией называется такое бесквантворное предложение сигнатуры σ_0, выводимое в T, что для любого терма t, входящего в него, существует такой константный символ $c \in \sigma_0$, что формула $t = c$ выводима в T. Частичной интерпретацией теории T в сигнатуре σ_0 называется пара $\langle \mathbf{A}, \sigma_0 \rangle$, где **A** – конечная алгебраическая система сигнатуры $\sigma(\mathbf{A}) \supseteq \sigma_0$, в которой истинны все σ_0-проекции. Можно доказать, что любая непротиворечивая теория имеет частичную интерпретацию в любой своей конечной подсигнатуре. Обычно в $|\mathbf{A}|$ помимо констант из σ_0 включают флаги – специальные значения, возвращаемые операциями при нарушении ресурсных ограничений. При моделировании вычислений в качестве T берутся арифметики на различных классах чисел, обогащенные достаточным количеством числовых констант. В число их частичных интерпретаций входят широко используемая машинная арифметика по модулю 2^q (чаще всего $q = 32$ или $q = 64$), арифметика с переполнением, арифметика вещественных чисел, определенная стандартом IEEE-754, и т. д.

Однако полноценные модели вычислений содержат не только арифметические действия, но и средства управления потоком их выполнения. Базовым средством такого рода является функция

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

отвечающая условному оператору `if x = a then y else z`. Как показано в [7], в число термальных операций алгебры **A** входят функции $\mathtt{If}^a$ для всех $a \in |\mathbf{A}|$ тогда и только тогда, когда ее клон состоит из всех функций, сохраняющих все ее подалгебры; такие алгебры называются полупримальными (semi-primal). Напомним, что условие сохранения множества X k-местной функцией f заключается в выполнении соотношения $f(X^k) \subseteq X$. Отметим, что любая полупримальная алгебра функционально полна.

Для анализа процессов сборки вычислительных систем мы привлекаем формализм дисциплин проектирования. Рассмотрим класс формальных моделей вычислений, состоящий из всех полупримальных алгебр. Для моделирования действий по их интеграции традиционное понятие гомоморфизма является слишком узким, поскольку требуется рассматривать в едином контексте алгебры, обладающие различными сигнатурами и вычислительными возможностями. Вместе с тем любое действие по интеграции должно быть отображением основных множеств, в достаточной степени согласованным с операциями. Можно показать, что оптимальным обобщением служит понятие субинъективного отображения, определяемое следующим образом. Будем обозначать через $\mathbf{A}[X]$ подалгебру алгебры **A**, порожденную множеством $X \subseteq |\mathbf{A}|$. Пусть **A** и **B** – конечные алгебры. Отображение $\varphi : |\mathbf{A}| \to |\mathbf{B}|$ называется субинъективным, если для любого подмножества $S \subseteq |\mathbf{A}|$, такого что $\varphi x = \varphi y$ влечет $x = y$ для всех $x, y \in S$, выполняется условие $\varphi(\mathbf{A}[S]) \subseteq \mathbf{B}[\varphi S]$. Несмотря на то что в результате субинъективного отображения моделей вычислений некоторые числа, различные в исходной модели, могут стать неразличимыми, для любого множества попарно различимых чисел никакие операции над ними не могут вывести за рамки подалгебры, порожденной его образом. Любое субинъективное отображение слабо субнепрерывно в том смысле, что прообраз любой подалгебры является объединением подалгебр. В свою очередь, любое субнепрерывное отображение (т. е. такое, что прообраз любой подалгебры относительно него является подалгеброй; в частности, любой гомоморфизм) субинъек-

тивно. Для инъективных отображений основных множеств алгебр свойство субнепрерывности эквивалентно субинъективности.

Все полупримальные алгебры и все субинъективные отображения их основных множеств образуют категорию, в рамках которой мы описываем процедуры сборки вычислительных систем, в том числе распределенных. 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 допустимой конфигурации вычислительной системы. Это облегчает реализацию динамического развертывания: топология размещения вычислительных компонентов диктуется ресурсными, а не структурными ограничениями.

В качестве интерфейса модели вычислений естественным образом выступает ее основное множество, получаемое путем "забывания" операций. Трансформации моделей вычислений сводятся к расширению множества термальных операций без изменения множества поддерживаемых чисел. Таким образом, получается формальная дисциплина проектирования $\langle \mathrm{CS}, \Delta\text{-CS}, |\text{–}|, \text{bij-CS} \rangle$, где CS – категория всех полупримальных алгебр и всех их субинъективных отображений; Δ-CS – класс всех конечных CS-диаграмм; $|\text{–}| : \mathrm{CS} \to \mathbf{FinSet}$ – канонический функтор в категорию **FinSet** всех конечных множеств и всех их отображений, забывающий операции; bij-CS – категория всех полупримальных алгебр и всех биективных субинъективных отображений. Эта конструкция демонстрирует, что вычислительные аспекты являются финальными единицами аспектной декомпозиции в том смысле, что для их проектирования используются традиционные модульные подходы. Действительно, в отличие от дисциплины моделирования сценариев, трансформации моделей вычислений являются действиями по интеграции, а не дуальными к ним. Вследствие этого вычислительные системы плохо поддаются аспектной ориентации: трассируемыми трансформациями являются только изоморфизмы, так что категория AO, порождаемая дисциплиной проектирования вычислений, эквивалентна CS. В связи с этим АОП не применяется в практике решения вычислительных задач.

Сложность вычислений оценивается как минимальное количество обращений к операциям некоторого базиса алгебры – минимального набора термальных операций, суперпозиции которых образуют ее клон [5]. Практический интерес представляют базисы, основу которых составляют вычислительные примитивы. Рассмотрим в качестве примера наиболее часто применяемую в современных компьютерах модель целочисленных вычислений по $\mathrm{mod}\, n$ $(n > 1)$, 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. В качестве флага используется значение n, называемое переносом (carry) и неотличимое от нуля при сложении и умножении. Предусматривается операция ветвления потока вычислений по флагу (предикация). Получается алгебраическая система следующего вида [9]:

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

Функции данной системы (не считая констант) порождают алгебру, любая подалгебра которой имеет вид $\{kd \mid k = 0, 1, \ldots, n/d\}$ для некоторого d, являющегося делителем n (это можно проверить с помощью алгоритма Евклида). Полупримальная алгебра, обладающая таким множеством подалгебр, является алгебраическим представлением $(n + 1)$-значной логики Лукасевича $Ł_{n+1}$ [21]. В этом представлении логическим значениям отвечают элементы множества $\{0, 1, \ldots, n\}$, а пропозициональным связкам – термы. В своей классической форме язык логики Лукасевича состоит из двух связок: отрицания $\sim x = n - x$ (совпадающей с унарным отрицанием в системе MA_{n+1}) и импликации $x \to y = \min(n, n - x + y)$. Тавтологиями этой логики в точности являются все термы, принимающие значение n при всех значениях своих аргументов. Поэтому для верификации реализации алгоритмов на предмет от-

сутствия переполнения можно привлекать технику доказательства логики Лукасевича (в том числе автоматизированную [22]).

Однако функции системы MA_{n+1} порождают $Ł_{n+1}$ не при всех n. Например, можно показать, что базисом в $Ł_{n+1}$ служит множество функций

$$\text{ММ}Ł_{n+1} = \begin{cases} \{(+), (-), (\text{Carry})\},\ n = 2, \\ \{(+), (-), (\times), (\text{Carry})\},\ n \in \{2^{\alpha}\textstyle\prod_i p_i \mid, \\ \quad \alpha \in \{0, 1\},\ p_i - \text{попарно различные простые числа}\} \setminus \{2\}, \\ \{(+), (-), (\max), (\text{Carry})\} \text{ в остальных случаях}, \end{cases}$$

где через (max) обозначена бинарная функция вычисления максимума чисел (упорядоченных как константы системы MA_{n+1}). Потребность в дополнительных функциях свидетельствует об относительно низкой эффективности модели вычислений по модулю, в том числе при $n = 2^{32}$ или $n = 2^{64}$. Тем не менее, такой потребности не возникает при решении ряда практически значимых вычислительных задач, например при программной реализации цифрового шифра RSA, основанного на вычислениях по mod pq, где p и q – различные простые числа.

4. Инженерия предметной области крупномасштабных систем мониторинга. При разработке крупномасштабных систем мониторинга не удается непосредственно использовать не только традиционные методы проектирования, но и традиционные виды технологических процессов. Для того чтобы сформулировать задачи, способные служить входными для широкоиспользуемых процессов (таких как Rational Unified Process или eXtreme Programming), требуется предварительно провести полномасштабный технологический цикл исследования и моделирования предметной области "мониторинг крупномасштабных объектов". Результатом этого цикла является набор формальных моделей, описывающих архитектуру ядра типовой системы мониторинга, интерфейсы взаимозаменяемых проблемно-ориентированных компонентов, точки изменчивости и расширения системы.

Первым этапом цикла является идентификация предметной области – установление границ, определение круга представляющих ее лиц и их целей, выделение смежных областей. В настоящей работе рассматривается мониторинг объектов, допускающих представление в виде сетей распространения потоков ресурсов (материальных объектов, энергии, информации и т. д.). Мониторингу подлежат параметры состояния узлов сети, значения расхода и характеристик ресурсов в определенных точках. По результатам анализа этих данных вырабатываются рекомендации по оптимизации распределения потоков, ремонту оборудования узлов, настройке противоаварийной автоматики и т. д. В качестве смежных предметных областей выступают различные инфраструктуры управления, от бухгалтерии до контролирующих инстанций.

В качестве примера приведем систему управления энергообеспечением крупного промышленного объединения, состоящего из множества территориально распределенных производственных единиц. К таким объединениям относятся федеральные транспортные инфраструктуры (железные дороги, магистральные нефте- и газопроводы), металлургические холдинги полного цикла и т. д. Система мониторинга энергообеспечения должна обеспечивать консолидацию и координацию процессов учета и планирования потребления энергоресурсов, оперативно-диспетчерского управления, технического обслуживания и ремонта энергетического оборудования, ведения нормативно-справочной информации об энергохозяйстве. Пример такой системы рассматривается в п. **5** настоящей работы.

Потребность в распределенной системе возникает также при мониторинге окружающей среды, когда непрерывно выполняется измерение характеристик потоков химических веществ, распространяющихся в атмосфере и гидросфере. Точки измерения представляют собой стационарные станции наблюдения либо передвижные лаборатории, оснащенные метеорологическим и гидрологическим оборудованием. Результаты измерений используются для экологического контроля работы предприятий, предсказания погоды, оценки состояния биосферы. Примером такой системы регионального масштаба служит электронный атлас "Атмосферные аэрозоли Сибири" [23].

В качестве менее традиционного примера системы мониторинга рассмотрим информационный портал – единую интегрированную точку доступа к распределенной совокупности разнородных информационных ресурсов. Интеграцию ресурсов можно свести к их связыванию посредством каналов распространения информации, обра-

зующих виртуальную сеть с динамической топологией [11]. Мониторинг потока блоков информации, проходящей по каналам, заключается в оперативном автоматизированном извлечении и анализе их метаданных с целью принятия решения об анонсировании (публикации) на портале. Единый формат анонсов повышает удобство восприятия и поиска тематической информации. Несмотря на то что к порталам не предъявляется критических требований реального времени и безопасности, с точки зрения архитектуры их можно рассматривать как системы мониторинга сети распространения информации. Примером применения такого подхода служит портал математических ресурсов MathTree [24].

После того как предметная область идентифицирована, необходимо произвести ее моделирование – сбор, систематизацию и формализацию предметной информации, поступающей из различных источников. В качестве базовой модели используется онтология – формализованный свод основных понятий, отношений между ними и правил логического вывода предметных знаний. Важным источником понятий онтологии задач мониторинга являются стандарты, например ГОСТ Р 8.596-2002 "Метрологическое обеспечение измерительных систем. Основные положения", определяющий концепцию измерительного канала – контейнера метаданных результатов измерения параметров состояния объекта. Ряд стандартов требует специального анализа с целью выделения онтологической основы включая разделение статических и динамических отношений, построение классификационной иерархии свойств и т. д. [25]. Привлекаются формализованные методы построения онтологий, например ODE (Ontology Domain Engineering [26]). Для записи онтологии используется язык OWL (Ontology Web Language), стандартизированный консорциумом W3C в рамках проекта Semantic Web. Применяются средства автоматизации процесса разработки онтологии, такие как система поддержки распределенной работы экспертных групп ONTOGRID [27], в которой в качестве модуля пользовательского интерфейса задействован графический редактор Protégé [28]. Онтология используется в качестве источника терминов для разметки сценариев мониторинга, проектирования алгоритмов анализа данных, формулирования требований качества к системе.

Модели предметной области служат источником информации для проектирования средств автоматизации предметных задач. Типовые проектные решения, применимые для широкого класса сетевых объектов мониторинга, можно выработать, используя аспектно-ориентированный подход. В первую очередь выделяются аспекты взаимодействия со средствами реального времени (АСУ ТП), с пользователями, с корпоративными автоматизированными системами (АСУП). Они обеспечивают сбор и хранение результатов измерения параметров объекта, оказание управляющих воздействий на объект, визуализацию информации и ее представление в виде различных отчетов, сопутствующий документооборот. Система в целом приобретает вид интеграционной платформы, связывающей АСУ ТП и АСУП в целостный комплекс.

Унификация обмена данными обеспечивается путем формирования единой информационной модели объектов и средств мониторинга. Она содержит реестры организационной структуры объекта, состава и характеристик узлов сети, схем и алгоритмов мониторинга, нормативно-справочной информации. Она является источником метаданных, полностью описывающих условия выполнения процессов мониторинга, от состава измерительных каналов до правил разграничения доступа к информации. Она строится путем трансформации онтологии предметной области в диаграмму "сущность – связь", реализуемую в виде реляционной базы данных [10]. Для выборки информации из модели предусмотрен универсальный интерфейс – профиль выборки данных. Он представляет собой реляционную структуру, описывающую совокупности извлекаемых объектов различных видов путем явного перечисления либо по ограничительным критериям (например "все силовые трансформаторы, установленные на заданной подстанции"). Каждый профиль имеет наименование и целевую функцию (класс задач, для которых требуется выборка). В частности, посредством профилей задаются входные данные для алгоритмов анализа, правила наполнения генерируемых отчетов, предпочтения пользователей, контрольные списки доступа субъектов к информации. Можно составлять сложные профили, для этого реализованы теоретико-множественные операции над ними. Аспект поддержки профилей представляет собой, по сути, средство решения задач в ограничениях (constraint solver), реализованное средствами реляционной СУБД.

Аспекты взаимодействия служат поставщиками и получателями информации для аспектов расчета и анализа данных. Большинство расчетных задач мониторинга состоит в распространении значений показателей, полученных в определенные моменты времени в определенных точках, вдоль оси времени (ретроспективный анализ, про-

гнозирование) и по сети объекта (замещение отсутствующих значений, вычисление интегральных показателей). Решение таких задач сводится к суперпозиции базовых алгоритмов, определяющих один шаг распространения (прогноз изменения состояния элемента сети за один период, объем потери ресурса в элементе сети и т. п.). Поэтому расчетный аспект представляет собой проблемно-ориентированную модель вычислений, в которой совокупность примитивов и вид целевых функций полностью определяются информационной моделью объекта, для анализа эффективности и корректности которой применяется подход, изложенный в п. **3**. Ее программная реализация генерируется по наполнению информационной модели в виде динамически подгружаемых модулей с фиксированным интерфейсом.

Координация функционирования аспектов в реальном времени осуществляется посредством автоматического реагирования на регистрируемые события. Базовый сценарий цикла реагирования приведен в п. **2**. Его реализация включает аспекты ведения единого журнала событий, отражающих изменения состояния системы, и оперативного оповещения аспектов обработки событий о фактах их возникновения. Отметим, что предметом мониторинга являются не только узлы сети объекта, но и компоненты самой системы мониторинга, в отношении которых проверяется работоспособность, доступность, загруженность и другие показатели качества. Унификация функций мониторинга позволяет придать системе высокий уровень автономности – способности автоматически оптимизировать свою конфигурацию при изменениях во внешнем окружении [12].

5. Интеграционная платформа учета и управления энергообеспечением. Управление энергообеспечением крупных территориально распределенных производственных объединений относится к числу ключевых инфраструктурных процессов, требующих координации на всех уровнях управления. Это особенно важно в условиях реформы энергетики, предъявляющей жесткие требования к крупным участникам рынка энергоносителей: они должны активно играть в конкурентном секторе, тщательно соблюдать заявленные режимы отпуска и потребления энергии, оперативно предоставлять контрагентам и регуляторам достоверную учетную информацию, строго соответствовать нормам безопасности и энергосбережения. Удовлетворение этих требований невозможно без надежного и гибкого решения по сквозной автоматизации энергоучета и управления энергообеспечением.

В рамках этого решения необходимо обеспечить адекватное разграничение задач между средствами АСУ ТП и АСУП. Средства АСУ ТП отвечают за получение и регулирование "натуральных" показателей объема энергообеспечения, измеряемых в физических единицах (кВт·ч, Гкал и др.). Они выполняют сбор и обработку данных, опираясь на физические параметры процессов энергообеспечения: состав и характеристики линий передачи энергоносителей, коммутационных аппаратов, счетчиков и датчиков расхода энергоносителей и т. д. Основные требования, предъявляемые к ним, задают значения показателей реального времени, надежности, погрешности.

В то же время средства АСУП обеспечивают в первую очередь функцию биллинга – превращение натурального показателя энергообеспечения в финансовый путем умножения на тариф. От них обычно требуются поддержка гибких многовариантных тарифных планов, начисление и исполнение штрафных санкций, обмен электронными документами с контрагентами и расчетно-кассовыми центрами. Корпоративное управление также включает бухгалтерский учет основных средств энергохозяйства, организацию их технического обслуживания и ремонта и т. д. Поэтому здесь автоматизация должна быть ориентирована на поддержку делопроизводства, а не на измерение и регулирование физических величин.

Такие глубокие функциональные различия между средствами АСУ ТП и АСУП требуют создания промежуточного слоя – интеграционной платформы, обеспечивающей прозрачный информационный обмен между ними. Содержание обмена составляет следующая информация:

− нормативно-справочная информация (НСИ), описывающая модель объекта управления;

− результаты измерений расхода энергоносителей и иных параметров;

− события объектов и средств измерения и управления;

− команды оперативно-диспетчерского управления.

Интеграционная платформа переводит эту информацию в открытые форматы, установленные международными и ведомственными стандартами, такими как IEC 61970 (CIM), IEC 60870-5, XML-макет 80020 АТС. Подключение к ней конкретных программных средств технического и управленческого уровня сводится к созданию мо-

дулей-адаптеров, обеспечивающих полноценную двустороннюю трансляцию всех перечисленных видов информации из форматов, поддерживаемых этими средствами, в открытые форматы и обратно. Основной сценарий информационного обмена предполагает, что НСИ ведется на уровне АСУП, а результаты измерений и события объектов управления поступают от уровня АСУ ТП. Однако на практике встречаются различные другие циклы информационной поддержки управления энергообеспечением, и все они должны поддерживаться интеграционной платформой.

Фактически основным назначением платформы является формирование и поддержка единого информационного пространства управления энергообеспечением объединенной структуры. Она становится основным инструментом автоматизации крупномасштабных циклов технологического управления энергохозяйством. Ввиду этого она должна поддерживать решение следующих классов задач.

1) Сбор и хранение значений отпуска и потребления, параметров качества энергоносителей, полученных различными средствами (измерение аттестованными приборами и АИИС, ручной ввод с актов учета расхода энергии).

2) Анализ и планирование энергообеспечения (замещение отсутствующих и недостоверных данных согласно методике выполнения измерений, вычисление учетных показателей энергообеспечения объектов и объемов обмена энергоносителями со смежными субъектами энергосети, расчет режимов энергообъектов, статистическое прогнозирование, сверка плановых значений с фактическими).

3) Анализ состояния энергооборудования (расчет балансов и потерь энергоносителей в оборудовании, показателей износа и надежности, погрешностей измерительных приборов), сравнение фактических значений показателей с нормативными, формирование рекомендаций по настройке, техническому обслуживанию и ремонту (ТОиР) оборудования.

4) Дистанционное управление технологическими процессами и компонентами системы (удаленная загрузка и настройка контроллеров энергообъектов, удаленное параметрирование систем автоматического управления и средств измерений, синхронизация аппаратных часов, телеуправление коммутационными аппаратами).

5) Предоставление единой точки доступа персоналу ко всей необходимой информации, имеющейся в системе, в режиме online (данные расхода энергоносителей, параметры состояния оборудования, массивы НСИ, документация и отчеты).

6) Технологический документооборот между распределенными уровнями организационной структуры (консолидация показателей энергообеспечения объектов на центральном уровне управления, предоставление результатов комплексного анализа и планирования на уровень отдельных объектов, оперативное управление ТОиР) и с внешним окружением (смежные субъекты, регуляторы энергорынка).

7) Постоянный мониторинг состояния энергосистемы (ведение журналов событий оборудования и телесигналов, сигнализация о системных сбоях, обнаружение значительных расхождений план/факт энергообеспечения, регистрация выполнения регламентных процедур, протоколирование доступа к защищенным элементам) с автоматическим оперативным оповещением заинтересованного персонала.

8) Консолидация разнородных массивов НСИ (организационная структура, состав и характеристики энергооборудования, реестры точек измерения и поставки энергоносителей, библиотеки схем энергосети, алгоритмы расчета и анализа) в единую масштабируемую модель объекта управления, с сохранением истории ее изменений.

9) Защита информации от несанкционированного доступа (парольная защита доступа пользователей, разграничение доступа по ролям и периметрам, наложение электронной цифровой подписи на документы с коммерческой информацией).

Особо отметим функции ведения массива НСИ. Энергетическая сеть представляет собой сложное машиностроительное изделие, изготовленное в единственном экземпляре. Опыт автоматизации машиностроения показывает, что для обеспечения высокого качества сложного изделия необходимо в течение всего его жизненного цикла вести всеобъемлющий информационный массив, фиксирующий его технические характеристики, проектные решения, факты технического обслуживания. Главной проблемой здесь является гарантирование актуальности информации, недопущение незарегистрированных воздействий на изделие. Интеграционная платформа позволяет решить эту проблему путем автоматической сверки фактического состава и характеристик энергооборудования,

считываемых с уровня АСУ ТП, с наполнением массива НСИ. Поэтому целесообразно развертывать платформу на объекте автоматизации как можно раньше, используя ее как средство управления строительно-монтажными и пусконаладочными работами. Это позволит значительно повысить их качество и обеспечить строгое соответствие фактически реализуемых технических решений проектным. В таком режиме платформа выполняет функции системы класса CALS (Continuous Acquisition and Lifecycle Support [29]).

Интеграционный уровень в настоящее время не представлен на рынке в виде самостоятельного программного решения, способного выполнять все перечисленные функции и готового к применению в промышленном режиме. Многие системы энергоучета и управления энергообеспечением, особенно зарубежного производства, включают те или иные средства интеграции. Однако последние не выходят за рамки расширений функциональных возможностей уровня АСУ ТП либо АСУП, не обладающих концептуальной целостностью и полнотой. Этим обусловлена практическая значимость интеграционной платформы, созданной автором настоящей работы и его коллегами. Ввиду спутанности требований и большого масштаба объектов ее разработка организована согласно технологическому циклу инженерии предметной области, описанному в п. **4** настоящей работы. Аспектно-ориентированный подход к проектированию платформы позволил обеспечить корректное разделение задач, установленных нормативами современного рынка энергоносителей, между единицами организационной структуры предприятий федерального масштаба с соблюдением ограничений производительности, масштабируемости и безопасности. В частности, классы задач (1)-(9) реализованы в рамках аспектов различных видов следующим образом:

− интерфейсные аспекты реализуют задачи (1), (4)-(6);

− аспекты информационной модели реализуют задачи (8)-(9);

− вычислительные аспекты реализуют задачи (2)-(3);

− аспекты координации реализуют задачу (7).

Первоначально подход был апробирован в рамках АИИС КУЭ для ОАО "АК Транснефть". В дальнейшем был создан программный продукт «Интеграционная платформа учета и управления энергообеспечением "Энергиус"» (свидетельство о государственной регистрации программы для ЭВМ №2009613359). Разработанная на ее основе система автоматизации оперативно-диспетчерского документооборота "Энергиус-Диспетчер" (свидетельство о государственной регистрации программы для ЭВМ № 2009611387) внедряется на объектах ООО "Газпром энерго" – оператора энергохозяйства группы "Газпром". Ведутся работы по созданию информационно-аналитической системы сбора и управления энергетическими данными (ССУЭД) для ОАО ФСК ЕЭС и "МРСК Холдинг".

Заключение. В работе представлены новые методы анализа и проектирования систем мониторинга, основанные на АОП: аспектно-ориентированное моделирование сценариев мониторинга, анализ эффективности вычислительных аспектов обработки событий, инженерия предметной области "мониторинг крупномасштабных объектов". Теоретическая база этих методов в общих чертах сформирована, в перспективе предполагается выполнить ее глубокий анализ и расширить сферу применения. На практике методы апробированы при разработке систем управления энергообеспечением крупных распределенных предприятий. Планируется развивать эти системы, а также разрабатывать системы мониторинга для других технологических секторов, характеризующихся потребностью в вертикальной интеграции (например, сегмент телекоммуникаций и связи).

Список литературы

1. Aspect-Oriented Software Development. Addison Wesley, Reading. 2004.

2. JAGADEESAN R., PITCHER C., RIELY J. Open Bisimulation for Aspects // Proc. AOSD'07, 2007. P. 107−120.

3. WHITTLE J., JAYARAMAN P. MATA: a tool for aspect-oriented modeling based on graph transformation // Lecture Notes in Computer Sci. 2008. V. 5002. P. 16–27.

4. SASSONE V., NIELSEN M., WINSKELL G. Deterministic behavioural models for concurrency // Lecture Notes in Computer Sci. 1993. V. 711. P. 682–692.

5. Дискретная математика и математические вопросы кибернетики. Т. 1. М.: Наука, 1974.

6. ЧАРНЕЦКИ К. Порождающее программирование: методы, инструменты, применение / К. Чарнецки, У. Айзенекер. СПб.: Питер, 2005.

7. Ковалёв С. П. Алгебраический подход к проектированию распределенных вычислительных систем // Сиб. журн. индустр. математики. 2007. Т. 10, № 2. С. 70–84.

8. Ковалёв С. П. Архитектура времени в распределенных информационных системах // Вычисл. технологии. 2002. Т. 7, № 6. С. 38–53.

9. Ковалёв С. П. Математические основания компьютерной арифметики // Математические труды. 2005. Т. 8, № 1. С. 3–42.

10. Ковалёв С. П. Применение онтологий при разработке распределенных автоматизированных информационно-измерительных систем // Автометрия. 2008. Т. 44, № 2. С. 41–49.

11. Ковалёв С. П., Яковченко К. Н. Организация информационных порталов на основе канальной интеграции // Тр. Междунар. конф. по вычислительной математике "МКВМ-2004". Рабочие совещания. Новосибирск: Ин-т вычисл. математики и мат. геофизики СО РАН, 2004. С. 66–72.

12. Кузнецов А. А., Ковалёв С. П. Тестирование и мониторинг в распределенных автоматизированных системах технологического управления // Вычисл. технологии. 2009. Т. 14, № 4. С. 57–69.

13. Маклейн С. Категории для работающего математика. М.: Физматлит, 2004.

14. Fiadeiro J. L., Lopes A., Wermelinger M. A mathematical semantics for architectural connectors // Lecture Notes in Computer Sci. 2003. V. 2793. P. 190–234.

15. Kiczales G., et al. Aspect-Oriented Programming // Lecture Notes in Computer Sci. 1997. V. 1241. P. 220–242.

16. Colyer A., Clement A., Harley G., Webster M. Eclipse AspectJ. Addison-Wesley, Reading, 2004.

17. Pinto M., Fuentes L., Troya J. M. DAOP-ADL: an architecture description language for dynamic component and aspect-based development // Lecture Notes in Computer Sci. 2003. V. 2830. P. 118–137.

18. Pratt V. R. Modeling concurrency with partial orders // Intern. J. Parallel Programming. 1986. V. 15, N 1. P. 33–71.

19. Glabbeek R. J. van, Goltz U. Refinement of actions and equivalence notions for concurrent system // Acta Informatica. 2000. V. 37, iss. 4/5. P. 229–327.

20. Sutcliffe A. Scenario-Based Requirement Analysis // Requirements Engng. 1998. N 3. P. 48–65.

21. Карпенко А. С. Логики Лукасевича и простые числа. М.: Наука, 2000.

22. Beavers G. Automated theorem proving for Łukasiewicz logics // Studia Logica. 1993. V. 52, N 2. P. 183–195.

23. Kovalyov S. P. Architecture of distributed information-computing system for exploring atmospheric aerosol // Proc. SPIE. 2005. V. 6160, N 1. P. 21–26.

24. Барахнин В. Б., Клименко О. А., Ковалёв С. П. Сбор и систематизация информации для портала математических ресурсов MATHTREE // Тр. Междунар. конф. "Вычислительные и информационные технологии в науке, технике и образовании". Т. 2. Павлодар: ТОО НПФ "ЭКО", 2006. С. 381–389.

25. Андрюшкевич С. К., Ковалёв С. П. Опыт адаптации стандартных информационных моделей для распределенных объектов технологического управления // Тр. 7-й Междунар. науч.-практ. конф. "Исследование, разработка и применение высоких технологий в промышленности". СПб.: Изд-во политехн. ун-та, 2009. С. 56–57.

26. Falbo R. A., Guizzardi G., Duarte K. C. An ontological approach to domain engineering // Proc. of the 14th Intern. conf. software engineering and knowledge engineering (SEKE-2002), Ischia (Italy), 2002.

27. Загоруйко Н. Г., Гусев В. Д., Завертайлов А. В., Ковалёв С. П. и др. Система ONTOGRID для автоматизации процессов построения онтологий предметных областей // Автометрия. 2005. Т. 41, № 5. С. 13–25.

28. KSL Protege Project. Stanford University. [Electron. resource]. http://protege.stanford.edu.

29. Норенков И. П., Кузьмик П. К. Информационная поддержка наукоемких изделий (CALS-технологии) / И. П. Норенков, П. К. Кузьмик. М.: Изд-во МГТУ, 2002.

Ковалёв Сергей Протасович – канд. физ.-мат. наук,
зав. лабораторией Конструкторско-технологического ин-та
вычисл. техники СО РАН; тел. (383)333-37-94; kovalyov@nsc.ru