Ломако Е.И.

Математические и понятийные средства системантики

Москва

2008

Содержание

Введение

Начиная с двадцатых годов XX века, усилиями многих выдающихся ученых стало развиваться новое научное направление известное как общая теория систем или системология [7, 8, 12, 38, 5, 59, 56, 19, 55 и др.]. За прошедший период были развиты многочисленные методы исследования систем, основанные на использовании дифференциальных и разностных уравнений, конечных автоматов, теории групп и математических структур [10, 21, 24, 1, 17, 37].

Простые технические и кибернетические системы могут быть представлены некоторой абстрактной моделью классической математики. Например, для большинства из них достаточным является описание в виде дифференциальных или разностных уравнений, особенностью которых является возможность предсказания поведения системы, если известно ее начальное состояние. Но для сложной системы мы не можем предсказать ее поведение на основе только локальной информации.

Широкое внедрение электронных вычислительных машин, создание теории программирования дало новый импульс методам анализа и моделирования сложных систем. В результате этих исследований был предложен объектно-ориентированный подход к описанию сложных систем [11, 44]. Сложилось целое направление научных исследований, получившее название семантического или концептуального моделирования в базах данных [42, 43, 47, 48,49, 63].

Неудовлетворительное состояние теории систем, опирающейся в основном на методы классической математики, привело к необходимости явного введения в системологию *семантики* как самого существенного фактора, который должен учитываться при изучении, понимании и описании реальных систем. Данный подход к анализу сложных систем был четко

сформулирован Ю. И. Шемакиным и выразился в создании нового научного направления – *системантики* [50].

Предметом изучения системантики являются сложные естественные и искусственные системы. Сложную систему необходимо отличать от большой. В большой системе информационный обмен между подсистемами осуществляется сигналами, которые имеют однозначную интерпретацию, а сам сигнал лишен семантики и общение между подсистемами происходит на чисто синтаксическом, функциональном уровне [55].

Напротив, в сложных биологических, общественных, социально-экономических и интеллектуальных системах передаваемая между подсистемами информация обладает определенным смыслом, который подсистемы-адресаты способны интерпретировать и выдавать на него разнообразные реакции, зависящие как от состояния самой подсистемы, так и от внешнего окружения. Таким образом, сложная система характеризуется семиотической (языковой) природой информационных связей между подсистемами, в противовес большим системам, где используется лишь чисто функциональное взаимодействие [55].

В сложных системах обнаруживаются не только ситуации детерминированного целевого поведения, но и ситуации, когда поведение системы не определяется четкой целью. Для сложных систем характерна возможность поведения, основанного не на заданной структуре целей, а на системе общих ценностей, позволяющих осуществить иерархическую структуру управления. В этом случае говорят о стратегически ориентированном управлении сложной системой [23].

Перечисленные особенности сложных систем требуют для своего описания адекватного математического и понятийного аппарата. В данной работе предлагается совокупность математических и понятийных средств, которые могут быть положены в основу системантики как науки, изучающей сложные системы.

Понимание особенностей исследования и описания сложных систем, трудность математической экспликации реальных систем привела, в итоге, к формированию новых взглядов на возможность применения классических математических методов для описания сложных систем и переходу к использованию языка теории категорий, универсальных алгебр и математической логики [9, 15, 26, 28, 35, 37, 40, 51].

Отличие современных логико-алгебраических методов от классической математики состоит в том, что в них явно присутствуют два уровня рассмотрения проблемы представления знаний. На внешнем, синтаксическом уровне с использованием математической логики строится абстрактное описание системы, которое можно определить как теорию системы. На внутреннем, более конкретном уровне рассматривается некоторая алгебраическая система или категория, которая является экспликацией реализации системы и одновременно моделью теории системы. И, кроме того, определяется некоторое отображение между алгебраической системой, как реализацией реальной системы, и абстрактной синтаксической теорией системы, то есть задается *семантическая интерпретация* для абстрактной теории.

Это, в частности, означает перенесение акцента с изучения лишь состояний сложных систем на изучение переходов между состояниями и систематическое включение в поле зрения исследователей одновременно статических и динамических аспектов систем, то есть переход от изучения статики к изучению процессов.

Использование теоретико-категорного подхода к описанию сложных систем также позволяет на основе экстремального принципа из совокупности возможных состояний системы в стационарном режиме выделить реально осуществляющиеся состояния. При этом, в отличие от традиционного применения вариационных принципов, функционал для нахождения экстремума не постулируется, а определяется естественным образом как

обобщение информации содержащейся в структурированном состоянии, исходя из категорной модели сложной системы [35].

Разбиение структурированного состояния системы на классы эквивалентности в данной работе трактуется как возникновение понятийной структуры сложной системы. Требуемые для анализа понятийной структуры концепции были достаточно детально изучены в семиотике и информатике. Одновременно, основываясь на объектно-ориентированном подходе в программировании, были созданы универсальные языки моделирования систем типа UML (Unified Modrelling Language) [11] и BPML [62], в рамках которых был представлен мультимодельный подход к описанию сложных систем.

Поэтому в работе значительное внимание уделяется методам концептуального анализа абстракций понятийных средств системантики, методам и средствам объектно-ориентированного описания статических и динамических аспектов сложных систем.

В начале работы дается краткий обзор методов классической математики, использовавшихся в ранних исследованиях систем. Показывается их ограниченная применимость для описания сложных систем, в силу отсутствия необходимых понятий для отражения структурных и динамических характеристик систем.

Дело в том, что классические математические модели системного анализа могут быть полностью представлены с помощью нескольких простых уравнений. А это означает, что они обладают низким информационным содержанием. С другой стороны, очевидно, что технические, социально-экономические и естественные сложные системы обладают высоким информационным содержанием.

Одним из средств, обеспечивающим возможность формулирования и передачи большого информационного содержания является естественный язык. Поэтому для сложных систем должны использоваться такие математические средства описания, которые опираются на формальные

конструкции языкового типа. К ним в первую очередь следует отнести методы современной алгебры (алгебраические системы и теорию категорий [9, 15, 55, 37, 26]), математической логики (языки исчисления предикатов [35, 51]), понятийные средства спецификации программ [3, 4, 20] и моделирования семантики в базах данных [47, 48, 40, 6].

Использование языка теории категорий и алгебраических систем, как более адекватного кругу возникающих проблем, позволяет рассматривать сложную систему как категорию в виде универсума структурированных объектов определенного вида и совокупности определенного вида преобразований (морфизмов) между такими объектами, допускаемых их макроструктурой. Такой взгляд на сложную систему обеспечивает возможность введения понятий структурного инварианта и структурной информации системы, как меры оценки сложности ее состояний. А это, в свою очередь, позволяет сравнивать мощности структурированных состояний, а затем на основе экстремального принципа вывести соотношения, определяющие оптимальное стационарное состояние системы, в котором она будет находится, в связи с исчерпанием запасов ресурсов. Материал данной части в основном опирается на результаты по теоретической биологии [28, 30, 31].

Во второй части работы развивается объектно-ориентированный подход для описания и анализа сложных систем, широко используемый в современном программировании. Как и в теории категорий в объектно-ориентированном моделировании (ООМ) сложных систем основным понятием является объект, который характеризуется определенным поведением. Но в отличие от теории категорий, которая не представляет средств для описания взаимодействий между объектами, в ООМ необходимые понятия вводятся явно в виде передаваемых между объектами сообщений, что делает ООМ более практичным и удобным инструментом для описания сложных систем системантики.

В ООМ принят мультимодельный подход к средствам описания статических и динамических свойств сложных систем. Для этой цели в ООМ используются модели верхнего уровня: модели прецедентов, позволяющие отобразить взаимодействие систем с внешним окружением (средой) и модели для представления сценариев внутренних функциональных процессов систем. Необходимые для этой цели концепции были разработаны в двух независимых программах, ставших в настоящее время стандартом де-факто: унифицированном языке моделирования (UML) и нотации моделей сценариев процессов (BPMN). Эти две модели, описывающие динамическое поведение сложных систем, в UML могут быть дополнены моделями, представляющими логическое и физическое описание сложной системы.

Отметим, что в ООМ также как и в логико-алгебраических моделях сложных систем принципиальным является представление систем реального мира в виде неразрывной совокупности двух уровней описания – концептуального (абстрактного) и физического (материального). В итоге совокупность всех этих взаимосвязанных моделей обеспечивает полную спецификацию сложной системы.

1 Классические методы описания простых и больших систем

1.1 Простые, большие и сложные системы

Под *системой* будем понимать целостную совокупность взаимосвязанных компонент (объектов), которая характеризуется организованностью, общим поведением и устойчивостью. *Организованность* системы проявляется в связях и взаимодействиях составляющих ее подсистем и объектов. *Целостность* позволяет рассматривать систему как некую единую сущность, как единый объект при анализе ее взаимодействий со средой. *Устойчивость* обеспечивает идентификацию системы в процессе ее эволюции. Таким образом, система – это устойчивая во времени и ограниченная в пространстве совокупность объектов, интересующая нас как единое целое [19, 56].

Выделение системы и фиксация ее объектов делит реальный мир на две части — систему и среду. *Среда* – это множество всех других потенциальных объектов, интересующих нас только с точки зрения их влияния на состояние и поведение системы и обратного влияния системы на состояния и поведение объектов среды. Таким образом, среда включает в себя множество потенциальных объектов, способных взаимодействовать с системой и влиять на ее состояние и поведение. При этом связи элементов внутри системы имеют значительно большую силу, чем их связи с элементами среды.

Системы можно подразделить на простые, большие и сложные. Простые системы – это системы с незначительным числом компонент и небольшим количеством взаимодействий между ними. Большие системы характеризуются большим числом элементов, функциональной организацией связей между подсистемами и однозначной интерпретацией поступающих сигналов. В таких системах сигнал лишен семантики и общение между

подсистемами и с внешней средой происходит на чисто синтаксическом уровне.

Сложная система, напротив, обладает языковой природой взаимодействий между подсистемами и внешней средой. Так, живые системы (биологические и социальные) поддерживают общение на языке, тексты которого обладают инвариантным смыслом и, кроме того, допускают различные интерпретации этого смысла, реализуемые различными получателями.

При описании сложных естественнонаучных и социальных систем сформировался скорее общий философский, чем математический подход. Словесное (вербальное) описание системы, как правило, представляет собой нагромождение нечетких высказываний, которые лишь затуманивают существо дела. Избавиться от таких нечетких и не до конца продуманных соображений помогает компактная математическая символика. При этом математическое описание часто приобретает чисто теоретико-множественный характер, и изменяется лишь взгляд на природу самого объекта [55].

1.2 Статика и динамика систем

Система определяется *структурой* и *поведением*. Эти понятия для системологии являются такими же фундаментальными, изначальными, неопределяемыми понятиями как для физики пространство и время. Под *структурой* понимается инвариантная во времени фиксация связей между элементами системы (статическое описание системы).

Под *поведением* системы понимается ее функционирование во времени (динамическое описание системы). Поведенческие аспекты системы выражаются с помощью понятий *функция* и *цель*. В самом общем плане *цель* – это состояние (совокупность свойств системы, определяющих характер её отношений с внешней средой), к которому направлена тенденция движения

системы. Заметим, что с усложнением иерархического строения системы возрастает и сложность ее поведения. Это связано с возрастанием сложности неабстрактной структуры систем [56].

Система в каждый момент времени может быть представлена в виде совокупности таких сущностей как объекты, классы и ситуации. Выделенная совокупность объектов, классов и ситуаций системы называется ее *состоянием.* С течением времени одни сущности системы исчезают, другие появляются, меняются свойства и взаимосвязи. Новые возникающие состояния считаются состояниями одной и той же системы. Следовательно, систему можно рассматривать в виде множества последовательно переживаемых состояний, и можно говорить о жизненном цикле системы.

С математической точки зрения система Σ – это некоторая часть мира, которую в любое данное время можно описать, приписав конкретные значения некоторому множеству переменных (фазовых координат), характеризующих ее состояние. Системная точка зрения требует введения *пространства состояний* $St\Sigma$ системы Σ как совокупности всех возможных конечных множеств ситуаций, в которых могут находиться элементы системы.

В пространстве состояний $St\Sigma$ системы можно определить ее траектории как последовательности состояний, в которых система находится в определенные моменты времени. Члены этой последовательности состояний не могут быть совершенно произвольными, поскольку каждое текущее состояние как-то связано с предшествующими состояниями. Поэтому систему можно определить как класс всех действительно возможных траекторий состояний. А совокупность всех общих ограничений на свойства траекторий системы и ее состояния – это семантика системы [47]. Данное определение семантики системы указывает на то, что точное и одновременно удобное описание системы может быть получено только в результате тщательного анализа различных представлений в виде *статических и динамических* моделей данной системы.

Статическое или структурное описание – это совокупность утверждений, которые связывают значения фазовых координат системы друг с другом при определенном выборе фазового пространства системы $St\Sigma$.

Динамическое описание, напротив, указывает, как изменение значений фазовых координат зависит от значений или изменений значений других координат. Таким образом, динамическое описание – это совокупность утверждений, из которых можно математически вывести поведение системы: описать, как она переходит из одного состояния в другое.

Динамическое описание рассматривает систему Σ как структуру, на которую в определенные моменты $t \in T$ оказывается входное воздействие $x(t)$ (например, поступает вещество, энергия или информация) и которая в какие-то моменты времени порождает некоторую выходную величину $y(t)$. Предполагается, что значения входного воздействия $x(t) \in X$ выбираются из некоторого фиксированного множества X (то есть в любой момент времени $x(t)$ должно принадлежать множеству X), а любое мгновенное значение выходной величины $y(t)$ – принадлежать некоторому фиксированному множеству Y.

Причем, в общем случае, значение выходной величины системы Σ зависит как от текущего входного воздействия, так и от предыстории этого воздействия, которое сказывается на внутреннем состоянии $q(t) \in Q$. Другими словами, внутреннее состояние системы $q(t) \in Q$, определяет текущее значение выходной величины системы $y(t) \in Y$ и оказывает влияние на ее будущее. Таким образом, при таком подходе внутреннее состояние системы $q(t)$ рассматривается как память, или накопитель прецедентов системы Σ, а математическое описание динамической системы является описанием потока причинно-следственных связей из прошлого в будущее [21].

Система тем сложнее, чем больше фазовых координат необходимо для описания ее состояния. Система тем более организована, чем больше у нее

возможностей противостоять возмущениям, препятствующим достижению цели. Математическая теория должна давать описание структуры, поведения и эволюции системы.

Математическое описание оказывается информативнее любого словесного описания, его использование позволяет каждому аспекту изучаемого явления поставить в соответствие математический символ, в результате чего становится более наглядной взаимосвязь, существующая между различными компонентами системы. Кроме того, математическое описание дает основу для численного анализа, с помощью которого могут быть получены данные не только описательного, но и прогнозного характера.

Теоретическое описание любой системы должно включать в себя, по крайней мере, два важных аспекта. Во-первых, построение конкретной математической модели допустимых состояний системы и ее переходов между этими состояниями. Во-вторых, установление правил отбора, позволяющих находить среди множества теоретически допустимых состояний системы те, которые осуществляются в реальности при данных внешних условиях.

Существует не так уж много вариантов формального описания простых и сложных систем в теоретическом естествознании. Ниже мы кратко рассмотрим несколько распространенных математических описаний, используемых при изучении простых и больших систем [24].

1.3 Внутреннее описание

Одно из широко распространенных описаний динамических процессов систем основывается на использовании языка дифференциальных уравнений, которые, по существу, представляют собой утверждения о связи некоторых величин и скоростей их изменения. При этом предполагается, что систему можно описать с помощью пространства состояний конечной размерности. В общем виде такое описание может быть представлено как

$$dq/dt = f(x(t), q(t), t), \quad q(0) = q_0,$$

$$y(t) = h(x(t), q(t), t),$$

где $q(t)$ – n-мерный вектор, компоненты которого описывают состояния системы в момент времени t, $x(t)$ – m-мерный вектор входов системы, $y(t)$ – p-мерный вектор наблюдаемых выходов системы, q_0 – начальное состояние системы, f: $Q \times X \rightarrow Q$ – переходная функция состояния системы Σ, а h: $Q \times X \rightarrow Y$ – функция выхода.

В дискретном времени динамика системы может быть описана с помощью разностных соотношений

$$q(k+1) = F(x(k), q(k), k), \quad q(0) = q_0,$$

$$y(k+1) = H(x(k), q(k), k).$$

Важным свойством такого описания является то, что оно дает представление о поведении системы в некоторой окрестности текущего состояния. То есть внутреннее состояние системы и состояние окружающей среды однозначно определяют последующее состояние системы. При этом предполагается, что собранная локальная информация позволит понять глобальное поведение системы во времени и в пространстве.

1.4 Внешнее описание

Внешнее описание основано на непосредственном представлении связи между входом и выходом системы. Иными словами, при данном подходе система $\sum$ является информационным процессором в некотором обобщенном смысле. Данное описание диаметрально противоположно локальному внутреннему описанию, поскольку оно не содержит деталей внутреннего представления системы и единственным источником информации является закономерность, связывающая выход системы с ее входом. При этом внутренний механизм преобразования входа в выход остается неизвестным.

Например, если мы располагаем лишь таблицей чисел, характеризующих *реакцию (выход)* системы на различные *внешние воздействия* (входы), то внешнее описание эквивалентно отображению

$f: X \rightarrow Y$,

где X – это множество возможных входов, Y – множество наблюдаемых выходов, а f – функция связи между множествами X и Y.

Описывать выходные значения системы как математические функции от входных значений для большинства систем оказывается наивным: вместо этого приходится определять ряд промежуточных значений, но не соответствующих никаким выходным значениям. Такой подход достаточен для анализа многих физических и технических систем, но его использование для описания менее изученных объектов типа кибернетических, биологических или социально-экономических систем не столь эффективно, а зачастую и невозможно.

1.5 Алгебраические методы

Описание системы с помощью дифференциальных уравнений слишком ограничено для систем, состояние которых изменяется дискретно. В отличие от традиционных математических моделей, которые опираются на непрерывные структуры и соответствующие методы анализа, в новых моделях основную роль играют дискретные структуры современной алгебры, а предположение о конечной размерности пространства может быть заменено на предположение о конечности числа его элементов и анализ системы может быть проведен с помощью алгебраических методов [1].

Алгебраическое описание системы Σ с конечным числом состояний включает

- множество допустимых входов X,
- множество наблюдаемых выходов Y,
- множество состояний системы Q,

- функцию перехода μ: $Q \times X \rightarrow Q$,
- функцию выхода λ: $Q \times X \rightarrow Q$,

При этом предполагается, что множества X, Y и Q конечны. Это позволяет представить описание системы Σ в виде пятерки

$$\Sigma = <X, Y, Q, \mu, \lambda>.$$

Изучение и понимание алгебраической структуры подобного описания основывается на теории конечных полугрупп [1].

1.6 Экстремальные методы

Уравнения, описывающие изменение состояния системы, - это лишь один из распространенных способов описания систем. Другой распространенный вариант состоит в применении целевой функции и формулировании экстремального принципа. В этом случае угадывание закона движения заменяется на постулирование функции или функционала, поиск экстремума которого методами вариационного исчисления приводит к описанию движения исследуемой системы. Теории, построенные с помощью уравнений движения или максимизации функционалов, как правило, эквивалентны друг другу, так как вариационные методы позволяют для любого заданного функционала выписать уравнения движения в форме уравнений Эйлера-Лагранжа, а для любого уравнения движения можно подобрать функционал, для которого оно является уравнением Эйлера-Лагранжа.

Использование в теории систем экстремальных методов основываются на изучении сложных систем с информационно-теоретической точки зрения, когда в качестве целевого функционала используется обобщенная энтропия. В этом случае анализ систем осуществляется на языке энтропии и целевых (потенциальных) функций, а реакция системы на внешнее воздействие рассматривается как динамическое изменение состояния системы, в процессе которого она стремиться максимизировать некоторую целевую функцию.

Описание динамического процесса на языке целевых (потенциальных) функций включает следующие составляющие [24]:

- пространство состояний (фазовое пространство системы) Q;
- множество входов системы X;
- целевую (потенциальную) функцию $H: Q \times X \rightarrow R$,

где R есть пространство действительных чисел.

При этом предполагается, что при фиксированном входном векторе $x = (x_1,...x_n)$ $x \in X$ наблюдаемое состояние системы соответствует локальному максимуму (минимуму) целевой функции H.

Использование потенциальных функций для описания хорошо изученных физических систем оказалось удачной альтернативой внутреннему описанию систем. С описанием системы на языке целевых функций тесно связана идея описания поведения систем с помощью обобщенной энтропии H, когда предполагается, что система изменяется таким образом, что в стационарном состоянии минимизирует изменение энтропии

$$H(q_1,...q_n) \rightarrow \text{extr},$$

где $q_1,...q_n$ - состояние фазовых координат системы, соответствующих экстремальному значению обобщенной энтропии.

Экстремальные принципы – наиболее общая форма теоретического объяснения во многих областях естествознания: оптике, классической теоретической механике, квантовой механике, электродинамике, теории управления, экономике и т. д.

Основная проблема в применении этого принципа состоит в отсутствии явных процедур для сопоставления каждой из исследуемых систем адекватного ее природе функционала. Но к концу XX столетия в математике сформировался подход, основанный на теории категорий и функторов, позволяющий оперировать полными совокупностями одинаково структурированных множеств и устанавливать соответствия между классами

объектов с различной аксиоматикой. Это значит, что уже на уровне описания систем возникает конструктивная возможность рассчитывать количественные характеристики состояний и с их помощью выявлять экстремальные состояния системы [29, 31, 32, 38].

2 Математические методы представления сложных систем

2.1 Системантика - наука о сложных системах

Важное место в теории систем занимает вопрос, что такое сложная система и чем она отличается от системы с большим числом элементов? Примерами больших систем могут служить многие технические и кибернетический системы, созданные человеком, а в качестве сложных систем обычно выступают биологический, общественные и социально-экономические естественные системы. Состав и количество компонентов, связей, функций не формируют сложность и не являются характеристикой сложной системы, ее отличительным признаком.

Техническая или кибернетическая система могут иметь любое число компонентов, любое разнообразие функций, но от этого они не становятся сложными системами. Структура системы в процессе ее жизненного цикла может изменяться, претерпевать эволюцию. Это означает, что статическое описание системы не может служить характеристикой ее сложности. Относительность понятия структуры системы заставляет отказаться от него при определении сложности системы и использовать для этого понятие поведения системы.

Сложность формируют число степеней свободы и размерность фазового пространства системы, нелинейность и способы информационного взаимодействия подсистем между собой. Для сложных систем определяющими являются не вещественно-энергетическая сложность, а структурно-поведенческие реакции, основанные на акте принятия решения и характеризующиеся целенаправленным поведением,.

Сложной является такая система, для которой сколь угодно подробное и точное знание устройства (морфологии) не раскрывает ее поведения (функцию), а сколь угодно длительное наблюдение за поведением сложной системы не позволяет предсказать ее поведения в будущем. Это означает,

что сложные системы слабо предсказуемы: их морфология не определяет функцию (и наоборот), а предыстория не определяет перспективу [27]. В частности, одной из мер сложности может служить интервал, на котором возможно предсказать поведение системы, если известно ее функционирование в прошлом. Чем меньше этот интервал, тем выше сложность.

Первое отличие сложной системы от большой состоит в том, что в большой системе между ее подсистемами происходит информационный обмен сигналами, которые имеют однозначную интерпретацию. Каждая подсистема воспринимает поступающие сигналы однозначно, а ее реакция заранее предусмотрена конструктором системы. Сообщение, передаваемое между подсистемами большой системы, не допускает никакого механизма интерпретации, никакого извлечения смысла. Фактически это означает, что связи между подсистемами построены по функциональному принципу, сигнал лишен семантики и общение между подсистемами происходит на чисто синтаксическом уровне. [55].

С другой стороны в биологических, общественных, социально-экономических и интеллектуальных системах циркулирующая между подсистемами информация обладает определенным смыслом, инвариантным относительно способов кодирования и используемого канала передачи, а подсистемы-адресаты способны *интерпретировать* получаемое сообщение и выдавать на него разнообразные реакции. Таким образом, сложная система характеризуется *семиотической* (языковой) природой информационных связей между подсистемами, в противовес большим системам, где используется функциональная сигнализация [55].

Второе отличие сложной системы от большой заключается в том, что в больших (например, технических и кибернетических) системах управление их поведением основывается на *целевых критериях*, когда достижимое конечное состояние определяется заранее поставленной целью или деревом

целей. Детерминированное целеполагание и иерархичность целевого управления специфичны именно для технических и кибернетических систем.

Напротив, в сложных системах (биологических и социальных) обнаруживаются не только ситуации детерминированного целевого поведения, но и ситуации, когда поведение системы не определяется четкой целью. Для сложных систем характерна возможность поведения, основанного не на заданной структуре целей, а на системе общих *ценностей*, позволяющих осуществить иерархическую структуру управления. В этом случае говорят о стратегически ориентированном управлении сложной системой [23].

Целевое управление возможно и при синтаксическом уровне обмена информацией в системе. Для этого достаточно, чтобы каждая подсистема получала информацию о том, насколько ее поведение соответствует принятой системе целей. Для этого не требуется, чтобы получаемое сообщение обладало инвариантной семантикой, общей для всех подсистем. Когда же согласованное действие всех компонент системы определяется общими ценностями, необходимо, чтобы эти общие ценности были выражены. Поэтому становится существенным, чтобы коммуникация и координация между подсистемами носила семантический характер, что, в свою очередь, требует наличия механизма интерпретации, механизма извлечения смысла получаемых сообщений.

Отмеченные различия между большими и сложными системами позволили Ю. И. Шемакину назвать науку, изучающую сложные системы *системантикой* [50].

2.2 Логико-алгебраические методы представление знаний о сложной системе

Технические и кибернетические системы могут быть представлены некоторой синтаксической абстрактной моделью. Например, для

большинства из них достаточным является описание в виде дифференциальных или разностных уравнений, особенностью которых является возможность предсказания поведения системы, если известно ее начальное состояние. Но для сложной системы мы не можем предсказать ее поведение на основе только локальной информации.

Кроме того, представление в виде синтаксической абстрактной модели инвариантно к системе, которую оно представляет. Эта инвариантность обеспечивает классической математике широкую применимость предлагаемых ею методов. Но в процессе перехода от содержательного описания сложной системы к ее синтаксической математической модели игнорируется семантика системы, что лишает такую модель конкретности. Вместе с тем, опыт показывает, что каждая сложная система уникальна. Абстрагирование от этой уникальности невозможно, так как построенная синтаксическая математическая модель не будет отражать индивидуального поведения сложной системы.

Сложную систему почти невозможно описать полностью и подробно. Поэтому проблема состоит в нахождении компромисса между простотой описания и необходимостью учета поведенческих характеристик сложной системы. Для разрешения этой дилеммы обычно используется иерархическое описание, когда система задается семейством представлений (моделей), каждое из которых описывает систему с точки зрения некоторого уровня абстрагирования. Модель системы может быть структурной, подчеркивающей организацию системы, или поведенческой, то есть отражающей ее динамику. При проектировании сложной системы такие модели также необходимы для визуализации и управления архитектурой системы. Модели помогают добиться лучшего понимания создаваемой системы и, тем самым, минимизировать возможные риски.

Для каждого уровня абстрагирования выделяется ряд характерных особенностей, законов, принципов, понятий и переменных, с помощью которых и описывается модель системы. Чтобы такое описание было

эффективным, необходима как можно большая независимость используемых моделей для различных уровней описания [39].

В зависимости от природы системы некоторые ее модели могут оказаться важнее других. Так, при создании систем для обработки больших объемов данных более важны модели, ориентированные на статические аспекты проектируемой системы. В интерактивных системах на первый план выходят представления статических и динамических прецедентов. В системах реального времени наиболее существенными будут представления с точки зрения динамических процессов. Наконец, в распределенных системах основное внимание необходимо уделять физической реализации и развертыванию компонент системы.

Уникальность поведения сложной системы требует для описания ее структурных и поведенческих свойств специальных семантических и прагматических средств. Для построения семиотической теории сложной системы необходимы специальные средства, которые применимы к любым системам, достаточно просто интерпретируются людьми и, кроме того, одновременно являются точными, структурированными и обозримыми.

К сожалению, классическая математика не позволяет сделать этого. Дело в том, что классические математические модели системного анализа, как это показано в первой главе, могут быть полностью представлены с помощью нескольких простых уравнений. Это, в свою очередь, означает, что они обладают низким информационным содержанием. С другой стороны, очевидно, что технические, социально-экономические и естественные сложные системы обладают высоким информационным содержанием. Вероятность возникновения высоко информативных конфигураций в моделях с низким информационным содержанием чрезвычайно мала. Одним из средств, обеспечивающим возможность формулирования и передачи большого информационного содержания является естественный язык. Поэтому для сложных систем должны использоваться такие математические средства описания, которые опираются на формальные конструкции

языкового типа. К ним в первую очередь следует отнести методы современной алгебры (алгебраические системы и теорию категорий [9, 15, 55, 37, 26]), математической логики (языки исчисления предикатов [35, 51]), понятийные средства спецификации программ [3, 4, 20] и моделирования семантики в базах данных [47, 48, 40, 6].

Основное отличие современных логико-алгебраических методов от классической математики состоит в том, что в них присутствует два уровня рассмотрения проблемы представления информации об описываемой системы. На внешнем, синтаксическом уровне с использованием математической логики строится абстрактное описание системы, которое можно определить как *теорию* системы. На внутреннем, более конкретном уровне рассматривается некоторая алгебраическая система или категория, которая является экспликацией реализации системы и одновременно *моделью* теории системы. И, кроме того, определяется некоторое отображение между алгебраической системой, как ее реализацией, и абстрактной синтаксической теорией системы, то есть задается *семантическая интерпретация* для абстрактной теории.

2.3 Многоуровневое представление сложных систем

Данная задача может быть решена путем использования таких средств описания системы, которые представляют необходимые базовые понятия, инвариантные по отношению к любым системам, и правила, позволяющие строить более сложные синтаксические конструкции на основе базовых. Средства представления информации о системах реального мира различные исследователи называют по-разному. Так, в работах по базам данных [22, 48, 49] их определяют как *концептуальные* (понятийные), или *информационно-логические,* а в исследованиях по искусственному интеллекту и экспертным системам [42] — как *представление знаний.*

Б. Брукс рассматривает знание как совокупность понятий, организованную в некоторую структуру или систему. В последнее время идея организации понятий все больше связывается с построением онтологий понятий, для которых разработаны специализированные инструментальные средства. Онтология – это набор определений фрагмента декларативных знаний на формальном языке, которые ориентированы на совместное многократное использование различными пользователями.

Универсализм этих средств должен обеспечиваться высокой общностью, абстрактностью системы базовых метапонятий и правил порождения новых понятий, которые допускают семантическую интерпретацию применительно к любым системам. Такие средства в силу их абстрактности в теории спецификации программ и моделировании баз данных стали называться *концептуальными* [4, 47].

При таком подходе все инвариантные свойства сложной системы могут быть выражены на концептуальном уровне (в теории сложной системы), а индивидуальные свойства системы отображены с помощью *внутренней модели*, описывающей ее конкретное поведение. Более того, различные внутренние модели, удовлетворяющие абстрактной теории системы, могут отражать разные взгляды исследователей на сложную систему (моделировать разные представления системы), что, в принципе, позволяет учитывать *прагматику* использования системы.

Концептуальная модель создается в результате формализации описания сложной системы с помощью соответствующих средств представления знаний в виде понятий, а реализация системы описывает конкретные сущности (значения), которые охватываются понятиями, введенными в рамках концептуальной модели. Это значит, что реализация системы моделирует состояния системы, а в рамках ее концептуального представления задаются статические и динамические правила (семантика сложной системы), определяющие ограничения на состояния системы,

которые могут рассматриваться как некоторые аксиоматические высказывания.

Таким образом, при данном подходе производится четкое разграничение концептуальной модели сложной системы и ее реализации. Выделение этапа концептуального моделирования сложных систем обеспечивает большую гибкость и независимость всего проекта описания сложной системы от ее конкретной реализации. При этом способ реализации отделяется от используемой концептуальной модели сложной системы. Конкретные данные о реализации системы отделены от содержащейся в них семантической информации (знаний), а индивидуальное поведение системы — от представления знаний.

Такой подход целесообразен по той причине, что количество элементов системы намного превосходит число понятий, необходимых для описания семантики системы, в связи с чем должны использоваться специализированные средства накопления, хранения и ведения большого числа однородно форматированных данных, обеспечиваемых современными системами управления базами данных.

Концептуальное моделирование обеспечивает высокоуровневые средства спецификации синтаксиса и семантики сложной системы, а базы данных хранят конкретные экземпляры элементов, составляющих сложную систему. Для решения данной задачи необходима взаимосвязанная совокупность понятий, механизмов и языковых средств, обеспечивающих формализованное описание системы. Эту задачу позволяет выполнить унифицированный язык моделирования UML (Unified Modeling Language), являющийся графическим языком визуализации, специфицирования, конструирования и документирования систем. Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.

Язык UML основан на некотором числе базовых понятий, которые могут быть изучены и применены большинством исследователей и разработчиков, знакомых с методами *объектно-ориентированного* анализа и проектирования. При этом базовые понятия могут комбинироваться и расширяться таким образом, что специалисты получают возможность самостоятельно разрабатывать модели больших и сложных систем в самых различных областях приложений [34].

Конструктивное использование языка UML для концептуального моделирования основывается на понимании общих принципов представления знаний о сложных системах и особенностей процесса их *объектно-ориентированного анализа* и проектирования. При этом одним из основных принципов описания сложных систем является принцип абстрагирования, который предписывает включать в описание только те аспекты проектируемой системы, которые имеют непосредственное отношение к выполнению системой своих функций или своего целевого предназначения. При этом все второстепенные детали опускаются, чтобы чрезмерно не усложнять процесс анализа и исследования полученной модели.

При объектно-ориентированном подходе система может быть представлена как совокупность некоторых объектов и отношений, которые существуют между ними. В зависимости от того, являются ли отправной точкой для представления знаний о системе объекты, отношения, трансформации состояний или истинные высказывания о состояниях, мы имеем, соответственно, описание, ориентированное на структурное, динамическое или логико-лингвистическое представление сложной системы.

Концептуальные спецификации системы на языке UML предоставляют участвующим в разработке ее модели специалистам (системотехникам, архитекторам, системным аналитикам и программистам) единую основу для создания проекта сложной системы.

2.4 Теория категорий - математический аппарат системантики

Знания о конкретной системе можно адекватным образом выразить, если они могут быть структурированы в виде системы понятий. Для этого должен использоваться набор базовых понятий и возможность конструирования более сложных понятий с помощью исходных. Выделенная понятийная структура сложной системы, определяет содержательное представление исследуемой системы.

Понимание этих особенностей исследования и описания сложных систем, трудность математической экспликации реальных систем привела, в итоге, к формированию новых взглядов на возможность применения классических математических методов для описания сложных систем и переходу к использованию языка теории категорий и универсальных алгебр [28, 29, 55].

Процесс математической формализации естественных наук показал, что каждое научное направление в своих основаниях обычно связано с использованием специфичного математического аппарата. Так, например, формулирование законов механики тесно связано с созданным Ньютоном дифференциальным и интегральным исчислением, теория электромагнитного поля Максвелла – с дифференциальными уравнениями в частных производных, статистическая физика – с теорией вероятностей, теория относительности Эйнштейна – с тензорным исчислением, а квантовая механика – с теорией гильбертовых пространств.

До недавнего времени наиболее общим математическим аппаратом, использовавшимся при изучении сложных систем, была теория множеств, когда явно или неявно предполагается, что любой объект системы принадлежит к некоторому множеству.

В отличие от этих более ранних работ, использующих теорию множеств и посвященных изучению внутреннего устройства различных систем, работы, основанные на использовании теории категорий, можно

охарактеризовать как *внешние*. В этих работах внимание сосредоточено в основном на таких взаимосвязях между состояниями системы, которые можно установить, не определяя явно внутренние операции или элементы структур состояний.

В этом случае для многих практически интересных задач состояние сложной динамической системы удается отождествить с некоторым несущим множеством, наделенным содержательной математической структурой (предпорядка, порядка, эквивалентности, толерантности, алгебраической, топологической и т.д.), отражающей с помощью определяющих соотношений семантику конкретной системы. Заданная на несущем множестве математическая структура, моделирует лишь те свойства *(макросостояния)* реальной системы, которые сохраняются неизменными при ее переходах из одного допустимого состояния в любое другое.

Макросостояния, определяемые выбранной математической структурой, можно отождествить с некоторыми классами состояний (понятий), которые сохраняются неизменными при переходах системы из одного состояния в другое. Именно этот фактор обеспечивает сложным системам необходимую устойчивость, несмотря на то, что элементы классов состояний могут претерпевать постоянные изменения.

Классам состояния системы, представляющим некоторые понятия, необходимые для описания системы, практически всегда удается сопоставить либо отношение эквивалентности, толерантности, либо отношение порядка на несущем множестве состояния системы *A*. При этом особую роль на начальном этапе играет правильное выделение классов эквивалентности, так как именно они образуют тот фундамент, на котором потом выстраивается вся иерархия понятий при объектно-ориентированном моделировании систем.

Множество всех классов эквивалентности образует фактормножество *A/E* несущего множества состояния системы *A* по отношению эквивалентности *E*. Это значит, что фактормножество *A/E* определяет

разбиение множества *A* на попарно непересекающиеся подмножества – *классы-понятия* исследуемой системы. Обратное также верно, если определено некоторое разбиение множества *A* на классы-понятия, то этому разбиению можно сопоставить отношение эквивалентности на этом множестве.

Например, множествами с разбиениями на классы эквивалентности можно описывать распределение по энергиям частиц газа, выделение из текста наборов сходных словоформ или сообщества биологических видов. При этом таксономическая иерархия выделенных классов может быть описана с помощью операций включения множеств, или некоторым отношением порядка [28, 55].

Если в системантике мы хотим рассмотреть сразу все возможные структурированные состояния системы (разделенные на некоторые классы и во взаимосвязях, существующих между ними), то, очевидно, нецелесообразно рассматривать отдельно элементы каждого состояния, принадлежащего *StΣ*. Гораздо удобнее изучать структурированные состояния интересующей нас системы как *единые и неделимые объекты*. Но в этом случае необходимо совместное рассмотрение структурированных состояния сложной динамической системы и отображений (переходов) между состояниями. Причем переходам между состояниями отводится особая роль. Именно в теории категорий состояния как объекты рассматриваются в связях с другими им подобными объектами, то есть совместно с допустимыми преобразованиями.

В теории категорий рассматриваются не отдельные множества, с какой либо структурой, а в поле зрения одновременно включаются все одинаково структурированные состояния, которые можно разделить на отдельные классы состояний, отождествив их с содержательными понятиями исследуемой системы. Это значит, что совокупность всех одинаково структурированных множеств (другими словами, множеств вместе с заданной на них *аксиоматикой*) составляет класс объектов категории, класс

состояний системы. Аксиоматика математической структуры (отношения, топологии, законы композиции и т.д.), используемой для моделирования состояний системы и определяющая категорию, задает семантику системы и выделяет рассматриваемую систему среди других систем.

Соответственно, при категорном подходе отдельно рассматривается устойчивая крупномасштабная структура состояний системы (статика системы), которая характеризует макросостояния данной системы, и множество микросостояний, изменяющихся в результате совместимых с макросостояниями переходов (динамика системы). Как станет ясно из дальнейшего изложения макросостояния системы можно отождествить с понятиями концептуального описания системы, а микросостояния с экстенсионалом этих понятий.

При категорном описании сложных систем должны выполняться следующие условия.

1) Каждой упорядоченной паре состояний системы $(A,B)\in St\Sigma\times St\Sigma$ отвечает множество $Mor_\Sigma(A,B)\subset Mor\Sigma$ переходов (морфизмов) структуры системы из состояния A в состояние B. При этом состояния системы (объекты категории) A и B представляют собой структурированные множества элементов $a_i\in A,\ i=\overline{1,n}$ и $b_j\in B,\ j=\overline{1,m}$. Каждый переход $\alpha\in Mor\Sigma$ это подмножество пар отношений

$$\alpha=\{(a_i,b_j)/a_i\in A,\ b_j\in B\}\subset A\times B.$$

Переход α принадлежит одному и только одному множеству переходов $Mor_\Sigma(A,B)$ из состояния A в состояние B, где $A,B\in St\Sigma$. Не исключается случай, когда множество переходов из состояния A в состояние B пусто, тогда $Mor_\Sigma(A,B)=\varnothing$ для некоторых A и B.

2) Для морфизмов, как для отношений, определена операция композиции или “умножения”, совпадающая с последовательным осуществлением переходов (функционированием) системы Σ из одного состояния в другое. Для каждой тройки состояний системы $A,B,C \in St\Sigma$ и переходов $\alpha: A \to B \in Mor_{\Sigma}(A,B)$, $\beta: B \to C \in Mor_{\Sigma}(B,C)$ естественным образом вводится операция $\gamma = \alpha * \beta$ последовательного перехода системы (операция композиции, или умножения отношений) из состояния A в состояние C $\alpha * \beta = \gamma: A \to C \in Mor_{\Sigma}(A,C)$. Здесь операция композиции переходов $\alpha * \beta \in Mor_{\Sigma}(A,C)$ – это отношение

$$\gamma = \alpha * \beta = \{(a,c) / a \in A, c \in C\} \subset A \times C$$

тогда и только тогда, когда существует элемент $b \in B$ такой, что одновременно

$$\alpha = \{(a,b) / a \in A, b \in B\} \subset A \times B \text{ и}$$

$$\beta = \{(b,c) / b \in B, c \in C\} \subset B \times C.$$

Композицию переходов системы Σ можно интерпретировать так. Если существуют переходы от структурированного состояния A к структурированному состоянию B и от B к C, то тем самым однозначно определяется переход от структурированного состояния A к структурированному состоянию C. Это означает коммутативность диаграммы, иллюстрирующей композицию переходов структурированных состояний системы (рис. 2.1), которую также можно интерпретировать как способы сравнения состояний системы: если состояние A сравнивается с B, а B с C, то автоматически возможно сравнение состояния A с состоянием C.

Рис. 2.1 Диаграмма композиции переходов структурированных состояний системы

Следовательно, композиция двух переходов α и β определена только в том случае, если образ перехода α совпадает с прообразом перехода β. Это значит, что композиция переходов системы возможна только при выполнении равенства im(α)=coim(β), то есть операция композиции условна и накладывает на систему *Σ*, рассматриваемую как целое, довольно серьезное ограничение. Это дополнительное ограничение теории категорий о возможности композиции морфизмов обеспечивает целостный взгляд на внутренние процессы, которые могут протекать в системе.

3). Композиция переходов системы *Σ* ассоциативна, т.е. $(\alpha*\beta)*\gamma=\alpha*(\beta*\gamma)$ для любых $\alpha \in Mor_{\Sigma}(A,B)$, $\beta \in Mor_{\Sigma}(B,C)$, $\gamma \in Mor_{\Sigma}(C,D)$. Ассоциативность композиции переходов означает, что переходы по разным путям должны приводить к одному и тому же состоянию системы (давать одинаковый результат).

4). Для каждого состояния $A \in St\Sigma$ существует тождественный переход $id(A,A) \in Mor_{\Sigma}(A,A)$, называемый единичным, такой, что $id(A,A) * \alpha = \alpha$, $\alpha \in Mor_{\Sigma}(A,B)$ и $\beta * id(A,A) = \beta$, $\beta \in Mor_{\Sigma}(C,A)$ и $A,B,C \in St\Sigma$. Морфизм $id(A,A)$ — это тождественное отношение $id(A,A)=\{(a,a)/a \in A\} \subset A \times A$, сохраняющее состояние системы.

Для указания элементов состояний системы *Σ* в теории категорий вводится особое финальное состояние, которое обозначается через *I*, обладающее следующим свойством: для каждого состояния *A* системы *Σ*

существует единственный морфизм из A в I, который обозначается через $I(A)$: $A \rightarrow I$. Кроме того, полезным является пустое состояние системы $\varnothing$, которое задается следующим свойством: для любого состояния A системы Σ существует единственный морфизм из $\varnothing(A):\varnothing \rightarrow I$ [15].

В системантике очень важно иметь также способ сравнения структурированных состояний системы, позволяющий находить в них сходные части. Именно эту роль в теории категорий выполняют *морфизмы*.

Морфизмы – это *отношения* между состояниями, сохраняющие математическую структуру состояний системы. Частный случай отношений представляют собой обычные функции, а также отображения. При этом каждая система характеризуется вполне определенным, присущим только ей, классом морфизмов. Например, морфизмами структуры множеств с разбиениями на классы эквивалентности являются отношения, переводящие каждый класс разбиения одного множества целиком в некоторый класс разбиения другого множества так, что разные классы эквивалентности преобразуются в разные.

Таким образом, сложная система представляется некоторой категорией, объединяющей класс объектов (класс состояний) и класс морфизмов (переходов между состояниями). Объекты категории эксплицируют состояния системы, а переходы между допустимыми состояниями представляемой системы отождествляются с морфизмами категории [29, 30, 32].

Предлагаемая математическая модель описания систем в системантике допускает простую теоретико-категорную интерпретацию, согласно которой допустимые состояния системы представляют собой объекты категории Σ, а разрешенные переходы между этими состояниями — морфизмы категории Σ. Представления об изменчивости системы Σ на языке теории категорий формализованы как изменения базового структурированного множества при сохранении его *крупномасштабной* структуры концептуального уровня. Так как аппарат категорий инвариантен к внутренней структуре используемых

понятий, то категорная модель системы принципиально целостна, в виду того, что внутренняя структура ее состояний не рассматривается.

Использование категорного подхода переносит акцент с "застывших", "мертвых" макросостояний системы, что характерно для теоретико-множественного подхода, на различные формы их движений и преобразований. Предметом исследования становятся не столько состояния систем, сколько совокупности способов их преобразований [32]. Но именно такая черта, как постоянное обновление, смена, преобразование материального субстрата, есть одно из отличий большинства естественных и биологических систем.

Три особенности теоретико-категорного описания систем позволяют думать, что язык теории категорий более адекватен для системантики и описания реальности, нежели язык теории множеств.

Первая особенность – возможность оперировать сразу всей совокупностью одинаково структурированных множеств (структурированных понятий системы), инвариантных относительно способа описания их внутренней структуры, что позволяет отождествить эту совокупность с пространством всех возможных состояний системы.

Вторая особенность – та, что в категорию наряду со структурированными понятиями равноправно и обязательно входят все допустимые их структурой способы изменения понятий, т.е. преобразования допускаемые макросостояниями системы, что позволяет рассматривать свойства состояний системы в сравнении с другими состояниями. Следовательно, использование морфизмов обеспечивает замену теоретико-множественного идеализированного представления систем в виде "застывших" понятий на более адекватное их представление *процессами*.

Наконец, третья особенность – это то, что теоретико-категорное описание систем не требует обязательного представления системы математической структурой: объектами категории могут быть не только математические конструкции, но и сущности реального мира. Возможно

“качественное” категорное описание систем, т.е. непосредственное перечисление и описание состояний системы и всех переходов между состояниями (морфизмов) не на математическом, а на специализированном, содержательном языке. Морфизмами при этом являются какие-нибудь реальные отношения между объектами категории (например, отношения толерантности), если они удовлетворяют аксиомам теории категорий.

Категория для описываемой системы предоставляет целый набор структур состояний – *фазовое пространство состояний*, хотя в реальности для данной системы реализуется одно из возможных состояний. Определение существующего состояния может быть сделано с помощью экстремального принципа, согласно которому осуществляются состояния системы с экстремальным количеством сохраняющих макроструктуру системы преобразований (переходов) состояний.

Основной чертой использования теории категорий в системантике является явное включение не только структурированных состояний сложной системы, но и всех операций, (преобразований, морфизмов), допустимых структурой состояний, то есть таких отношений из одних структурированных множеств в другие структурированные множества, которые не нарушают их структуру. При этом оказывается, что свойства объекта категории, эксплицирующего структурированные состояния данной системы, которые обычно формулируются через его внутреннюю структуру, адекватно выражаются через свойства отношений этого объекта с однотипными с ним объектами. Именно эта возможность – переводить изучение внутренней структуры в изучение внешних связей объясняет значение теории категорий в изучении сложных систем.

2.5 Алгебраические системы и реляционные алгебры как модели состояний сложной системы

Таким образом, в категорном подходе к представлению знаний об исследуемой системе, предполагается, что состояние сложной системы может быть структурировано в виде системы понятий (макросостояний) и переходов между состояниями. С каждым понятием связывается имя понятия, его определение и семантика в виде системы аксиом или определяющих соотношений, то есть теория категорий предоставляет математические средства для отражения семантики и логики сложной системы.

Одной из наиболее важных категорий для системантики является категория, в которой объектами являются многосортные *алгебраические системы*, то есть в качестве структурированных состояний сложной системы выступают алгебраические системы. Первоначально понятие алгебраической системы в математике было введено А. И. Мальцевым. Позднее оно стало эффективно использоваться в программировании и информатике при исследованиях абстрактных типов данных.

Напомним, что согласно [37] многосортной алгебраической системой *AC* называется тройка

$$AC=<s, \omega_f, \omega_r>, \text{ где}$$

$s =\{s_1, s_2 \cdots s_n\}$ – несущие множества (сорта) алгебраической системы;

$\omega_f =\{f_1, f_2 \cdots f_k\}$ – множество операций, определенных на несущих множествах;

$\omega_r = \{r_1, r_2 \cdots r_l\}$ – множество отношений, заданных на несущих множествах.

Для рассмотрения проблем системантики исследователю не всегда может быть известна конкретная реализация состояний сложной системы, но обычно ясны основные имена ее составных частей, имена используемых

операций и имена взаимосвязей между частями системы. Этого знания обычно достаточно, для понимания концептуальной архитектуры системы.

Для того чтобы выделить некоторый класс подобных алгебраических систем, введем понятие сигнатуры и множества определяющих соотношений. Для этого сопоставим конкретным сортам s_i несущих множеств алгебраической системы имена этих сортов S_i, операциям f_j – имена операций F_j и отношениям r_l – имена отношений R_l.

Теперь можно определить сигнатуру Ω многосортной алгебраической системы как тройку

$$\Omega = <S, \Omega_F, \Omega_R>, \text{ где}$$

$S = \{S_1, S_2 \cdots S_n\}$ – множество имен несущих множеств алгебраической системы;

$\Omega_F = \{F_1, F_2 \cdots F_k\}$ – множество имен операций, с указанием типа операции и ее местности α_i вида

$$F_i: S_{i1} \times S_{i2} \times \cdots \times S_{i\alpha_i} \to S_{t(i)};$$

$\Omega_R = \{R_1, R_2 \cdots R_l\}$ – множество отношений местности β_j, заданных на именах сортов несущих множеств

$$R_j \subseteq: S_{j1} \times S_{j2} \times \cdots \times S_{j\beta_j}.$$

Сигнатура несет важную информацию о самой алгебраической системе: по ней легко восстановить, сколько операций и отношений в данной алгебраической системе и узнать какова их местность. Кроме того, знание сигнатуры позволяет говорить об одноименных операциях и отношениях в алгебраических системах одинаковой сигнатуры. Но сигнатура не отражает всех свойств алгебраической системы. Так, например, сигнатура не позволяет указать, между какими элементами несущих множеств выполняются те или иные отношения.

Любая сигнатура определяет синтаксическую основу языка, на котором можно описывать свойства алгебраических систем. Выделение сигнатуры важно для того, чтобы знать какие логические формулы можно писать с

операциями и отношениями алгебраической системы. Например, если известно, что отношение R_j имеет местность β_j, то можно написать формулу $R_j(x_{j1}, x_{j2}, \cdots x_{j\beta_j})$, которая принимает истинное значение на подмножествах алгебраической системы, принадлежащих данному отношению, то есть формулу $R_j(x_{j1}, x_{j2}, \cdots x_{j\beta_j})$ можно рассматривать как некоторое высказывание о произвольных элементах $x_{j1}, x_{j2}, \cdots x_{j\beta_j}$ несущих множеств системы. С помощью логических операций можно из таких атомарных формул строить более сложные. Более точная формулировка будет приведена в разделе 4.1.

Каждая алгебраическая система, построенная путем отображения символов сигнатуры **Ω** в реальные несущие множества, операции и отношения на этих множествах, а также удовлетворяющая некоторым определяющим соотношениям (аксиомам), придает символам сигнатуры определенное *содержание*, или задает *семантику* сигнатурных символов: сорта превращаются во множества элементов, символы операций – в конкретные отображения, символы отношений – в конкретные подмножества. Это означает, что сигнатура определяет структуру системы как абстрактную категорию, с точки зрения которой неважно, из каких элементов построена данная система и какова реальная природа отношений между этими элементами.

Для того чтобы конкретизировать алгебраическую систему, представляющую данную сложную систему необходимо задать соответствующую совокупность определяющих соотношений. Определяющие соотношения могут быть заданы несколькими способами. Самый распространенный способ заключается в задании некоторой системы равенств типа

$$E=\{E_i\}, i=\overline{1,m}.$$

Для выделения алгебраических систем, представляющих реальные сложные системы, равенств часто недостаточно. Поэтому применяют

подход, основанный на использовании понятия *формула*. Формула – это равенство или одна из следующих форм:

$$\neg\Phi,\ \Phi_1 \wedge \Phi_2,\ \Phi_1 \vee \Phi_2,\ \Phi_1 \rightarrow \Phi_2,\ (\forall x)\Phi,\ (\exists x)\Phi,$$

где x – переменная;

$\Phi,\ \Phi_1,\ \Phi_2$ – формулы;

$\neg,\ \wedge,\ \vee,\ \rightarrow$ – булевские операции *не, и, или* и *импликация*.

Формулы такого вида, входящие в определение алгебраической системы, называются также аксиомами.

Определяющие соотношения сами по себе не истинны и не ложны. Но при их сопоставлении с конкретной алгебраической системой, они становятся высказываниями о свойствах данной системы. Задание системы аксиом или определяющих соотношений позволяет более точно специфицировать алгебраическую систему представляющую состояния реальной сложной системы. В такой алгебраической системе можно выделить отношение (классы) эквивалентности элементов, для которых выполняются заданные аксиомы. Отношение эквивалентности данного типа можно определить как отношение *семантической эквивалентности*, элементы каждого класса которой составляют экстенсионал данного класса. Классам эквивалентности алгебраической системы можно присвоить конкретные имена, связав с указанными классами определенные понятия сложной системы. Классы эквивалентности позволяют рассматривать алгебраическую систему как фактор-множество, соответствующее определяющим соотношениям данной системы.

Одной из наиболее интересных алгебраических систем, получившей широкое распространение для описания баз данных и позволяющей абстрактно представлять сложные системы является *реляционная алгебра RelAl*, а подход, основанный на использовании реляционной алгебры для представления состояний реальных систем можно назвать *реляционным подходом*.

Суть реляционного подхода в том, что состояния реальных систем можно представить в виде совокупности поименованных отношений (таблиц), а функционирование системы – в виде выполнения набора операций над этими отношениями. Операции над отношениями понимаются как алгебраические операции, а реляционная алгебра – это множество элементов, замкнутое относительно операций над отношениями.

Так как операции реляционной алгебры определяются на множествах различных сортов, то соответствующая алгебраическая система является многоосновной. Соотношения между операциями, которые выполняются на любых отношениях, называются аксиомами реляционной алгебры или ограничениями целостности. С помощью ограничений целостности задается семантика реляционной алгебры, представляющей данную конкретную сложную систему [6].

Следовательно, основными понятиями реляционного подхода являются понятия *отношение, схема отношения, операции над отношениями и состояния схемы отношения*.

Схема отношения $R(A_1, A_2 \cdots A_k)$ – это пара, состоящая из имени отношения R и множества его атрибутов $(A_1, A_2 \cdots A_k)$. Каждый атрибут A_i, задается парой $(A_i, DomA_i)$, состоящей из имени атрибута A_i и множества возможных его значений – домена атрибута $DomA_i$.

Отношением r в реляционной алгебре называется подмножество в декартовом произведении доменов атрибутов данного отношения

$$r \subseteq: DomA_1 \times DomA_2 \times \cdots \times DomA_k.$$

Отношения реляционной алгебры можно представить в виде таблицы, у которой строками являются элементы отношений, а атрибутам соответствуют столбцы таблицы. Каждый столбец состоит из имени и соответствующего этому столбцу домена значений. Строки таблицы представляют собой множество, поэтому порядок строк не имеет значения, а так как каждый столбец таблицы имеет имя, то порядок столбцов также не важен.

Состоянием схемы отношения $R(A_1, A_2 \cdots A_k)$ является отношение r с тем же набором атрибутов. В различные моменты времени состояние схемы отношения может быть различным, что позволяет моделировать статику системы.

Для моделирования динамики сложных систем в реляционной алгебре используются операции над отношениями. Эти операции, в принципе, могут быть различными для разных систем, но обычно они замкнуты относительно композиции операций. Наиболее часто используются традиционные операции над отношениями, предложенные Е. Ф. Коддом: проекции, соединения отношений, переименования атрибутов отношений и теоретико-множественные операции объединения, пересечения и разности отношений над отношениями с одинаковым набором атрибутов.

Результатом применения теоретико-множественных операций $\cup$, $\cap$ или $/$ к таблицам с одинаковыми наборами атрибутов является таблица с тем же набором атрибутов, множество строк которой состоит соответственно из объединения, пересечения и разности строк исходных таблиц

$\cup, \cap, /: R(A_1, A_2 \cdots A_k) \times R(A_1, A_2 \cdots A_k) \rightarrow R(A_1, A_2 \cdots A_k);$

Для набора атрибутов $X=\{A_1, A_2 \cdots A_n\}$ и $Y=\{A_1, A_2 \cdots A_m\}$, таких, что $Y \subset X$ определена операция проекции X на Y

$$pr_Y^X : R(X) \rightarrow R(Y).$$

Для набора атрибутов X, Y и Z, таких, что $Z = X \cup Y$ определена операция соединения отношений

$$* : R(X) \times R(Y) \rightarrow R(Z).$$

Для любого согласования наборов атрибутов μ: $Y \rightarrow X$ определена операция

$$\mu^* : R(X) \rightarrow R(Y).$$

В реляционной алгебре роль аналогичную сигнатуре абстрактной алгебраической системы играет схема реляционной алгебры. Схема реляционной алгебры – это пара

$$SchRel = <R, E>,$$

состоящая из набора базисных схем отношений $R = \{R_1, R_2 \cdots R_k\}$, состояние которых описывает исследуемую систему и набора зависимостей (определяющих соотношений), которые в реляционной алгебре называются ограничениями целостности E, вида:

$$F_1(R_1, R_2, \ldots R_n) = G_1(R_1, R_2, \ldots R_n)$$

$$\ldots \qquad\qquad \ldots$$

$$F_k(R_1, R_2, \ldots R_n) = G_k(R_1, R_2, \ldots R_n),$$

где $F_i(R_{i1}, R_{i2}, \ldots R_{in})$ и $G_i(R_{i1}, R_{i2}, \ldots R_{in})$ правильно построенные выражения, содержащие имена операций, имена отношений и константы.

В виде таких выражений можно записать условия функциональной зависимости атрибутов отношений, вхождение данного отношения в другое как подмножество и т. д. Ограничения целостности выражают смысл отношений, используемых для представления реальных систем, то есть выражают семантику моделируемой системы.

Состоянием S схемы реляционной алгебры называется такое сопоставление схемам отношений с именами $R_1, R_2, \ldots R_n$ соответствующих отношений $r_i = S(R_i)$ с тем же набором атрибутов, при котором выполняются все ограничения целостности, то есть если R_i сопоставляется отношение r_i, $i = \overline{1,n}$, то

$$f_i(r_{i1}, r_{i2}, \ldots r_{in}) = g_i(r_{i1}, r_{i2}, \ldots r_{in}), \; i = \overline{1,k}$$

Множество всех состояний схем отношений, которые получаются из базисных состояний посредством применения к ним операций над отношениями, замкнуто относительно операций, то есть является реляционной алгеброй.

Таким образом, в системантике класс структурированных состояний системы — это совокупность многосортных абстрактных алгебраических систем, определенных на одних и тех же несущих множествах $Sort=\{S_1, S_2 \cdots S_n\}$ и обладающих одной и той же сигнатурой Ω (одной и той же схемой), а класс $Mor\Sigma$ — фиксированный класс переходов системы из одной абстрактной алгебраической системы в другую алгебраическую систему (класс морфизмов системы).

2.6 Представление поведения систем в теории категорий

Как уже указывалось в основе категорного подхода лежит некоторый набор операций (морфизмов), который позволяет выразить поведение понятий. Следовательно, в отличие от теории множеств, в теории категорий преобразования (динамика) системы входят в аксиоматическое определение категории наравне с самими объектами (объекты – аналоги множеств).

Таким образом, при категорном представлении системы акцент со статического описания системы переносится на различные формы их преобразований, то есть на динамику происходящих в системе процессов. Предметом изучения становятся не столько состояния системы, сколько совокупности способов их динамических преобразований, что есть одно из отличий большинства естественных и искусственных систем системантики.

Так, например, для экологического сообщества морфизмами, сохраняющими разбиение на виды, будут преобразования, состоящие в рождении или смерти особей. При этих преобразованиях биологический вид переходит в себя. Для произвольных множеств с разбиениями по определению морфизмами служат соответствия, не перемешивающие классы эквивалентных элементов.

Основным тезисом категорного подхода к представлению знаний является предположение, что понятие может быть представлено в виде

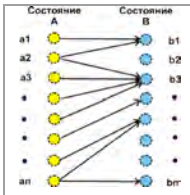
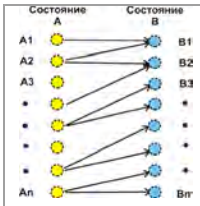
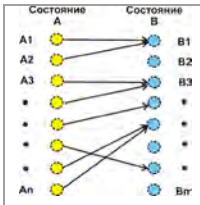
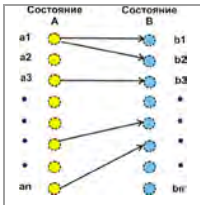
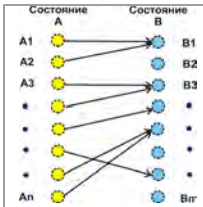
наборов имен типов (сортов) элементов, имен преобразований (операций) и аксиом (соотношений), которым должны удовлетворять операции. При этом не все элементы типов могут быть известны внутри данного понятия: они могут уточняться в других понятиях [6].

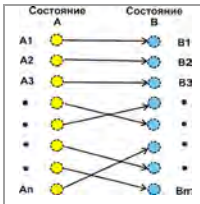
Операции перехода $\alpha(A,B)$ от состояния A к состоянию B системы Σ в теории категорий определяются в виде множества морфизмов $\alpha(A,B) \in Mor_{\Sigma}(A,B)$. Кроме того, могут использоваться определенные выше операции *композиция морфизмов* (α,β) при условии, что $coim\beta = im\alpha$, *тождественного отображения* $id(A,A)$, *морфизм* $I(A)$ перехода произвольного состояния A в финальное состояние системы I и *морфизм* $\varnothing(A)$ перехода из инициального состояния системы $\varnothing$ в состояние A, а также операции вида $im(\circ)$, $coim(\circ)$, задающие соответственно образ морфизма и его прообраз.

Для системантики наиболее важными классами морфизмов между состояниями $A=\{a_1,a_2,...a_n\}$ и $B=\{b_1,b_2,...b_m\}$ сложной динамической системы являются (табл. 2.1):

- Всюду определенное отношение $p(a_i) \equiv \{b|\ (a,b) \in p\} \subset A \times B$, для которого $p(a) \neq \varnothing$ при любом элементе a, принадлежащем состоянию A.
- Сюръективное отношение $s(a) \subset A \times B$, для которого $s^{-1}(b) \neq \varnothing$ для любого элемента b, принадлежащего состоянию B.
- Функциональное отношение $f(a) \subset A \times B$, для которого $f(a)$ либо пусто, либо состоит из одного элемента при любом a, принадлежащем состоянию A.
- Инъективное отношение $i(a) \subset A \times B$, для которого $i^{-1}(b)$ либо пусто, либо состоит из одного элемента при любом b, принадлежащем состоянию B.

Табл. 2.1. Наиболее важные классы морфизмов системантики.

Тип	**Обозначение**	**Семантические ограничения**	**Граф морфизма**
Всюду определенное отношение	$p(a_i) \equiv$ $\{b_j \mid (a_i, b_j) \in p\}$ $\subset A \times B$	$p(a_i) \neq \varnothing,\ i = \overline{1,n}$ Для всюду определенного отношения стрелки выходят из всех элементов исходного состояния A	
Сюръективное отношение	$s(a_i) \equiv$ $\{b_j \mid (a_i, b_j) \in s\}$ $\subset A \times B$	$s^{-1}(b_j) \neq \varnothing,\ j = \overline{1,m}$ Для сюръекции стрелки приходят вовсе элементы конечного состояния B	
Функциональное отношение	$f(a_i) \equiv$ $\{b_j \mid (a_i, b_j) \in f\} \subset$ $A \times B$	Либо $f(a_i) = \varnothing$, либо состоит из одного элемента. Для функции из исходного состояния A выходит не более одной стрелки	
Инъективное отношение	$i(a_i) \equiv$ $\{b_j \mid (a_i, b_j) \in i\} \subset$ $A \times B$	Либо $i^{-1}(b_j) = \varnothing$, либо состоит из одного элемента, $j = \overline{1,m}$ Для инъекции в каждый из элементов конечного состояния B приходит не более одной стрелки	
Отображение	$o(a_i) \equiv$ $\{b_j \mid (a_i, b_j) \in o\}$ $\subset A \times B$	$o(a_i) \neq \varnothing$ и состоит из одного элемента, $i = \overline{1,n}$ Для отображения из всех элементов исходного состояния A выходит ровно одна стрелка, то есть отображение – это одновременно всюду	

		определенное и функциональное отношение множеств	
Биективное отношение	$b(a_i) \equiv$ $\{b_j \mid (a_i, b_j) \in b\}$ $\subset A \times B$	$b(a_i) \neq \varnothing$, а также $b(a_i)$ и $b^{-1}(b_j)$ состоят из одного элемента, $i, j = \overline{1,n}$, $n=m$ Для биекции из всех элементов исходного состояния *A* выходит ровно одна стрелка и во все элементы конечного состояния *B* приходит ровно одна стрелка, то есть биекция – это одновременно функциональное, сюръективное и инъективное отношение множеств.	

Однако для представления сложных состояний систем приведенных операций недостаточно. Нужны также операции типа декартова произведения состояний, объединения состояний, пересечения состояний, разбиения состояний по классам эквивалентности и т. д.

При этом необходимо помнить, что при категорном подходе понятия (состояния) не определяются своими элементами, что позволяет отказаться от использования теоретико-множественных определений, требующих описания соответствующих множеств как законченных объектов, что, чаще всего, не реально.

Дело в том, что множество существует постольку, поскольку априорно существуют его элементы. Но для системы сама возможность выделения элементов и их идентификация, особенно на начальном этапе исследования, проблематична, так как первоначальный состав этих элементов размыт, не обладает достаточной индивидуальностью, а класс этих элементов открыт и не определен до конца.

Напротив, для множеств характерна именно замкнутость его элементов, которые должны быть полностью выделены, индивидуализированы и отождествлены. Поэтому при описании систем необходимо, в первую очередь, в неустойчивой и аморфной ситуации неполного знания выделить четко обнаруживаемые и различаемые эквивалентные элементы состояний (понятия, подсистемы и т. д.), адекватно описывающие поведение системы. Это значит, что четкие множества, необходимые для представления системы, появляются лишь в результате самого процесса изучения и описания реальных систем.

Поэтому в системантике вместо актуально бесконечных множеств, используемых в математике, достаточно рассматривать развивающиеся во времени конечные множества (потенциально бесконечные множества), то есть множества, которые в каждый момент времени представляются своим конечным набором элементов [6, 55].

При описании систем исследователю может быть неизвестна, а часто и неважна, конкретная реализация системы, но необходимо знание имен основных понятий, их отображение на составные части системы и знание некоторых свойств используемых операций и отношений. Такое знание достаточно для того, чтобы четко описать систему с помощью выделенных понятий и определить способ их обработки. Описание системы на основе использования теории категорий позволяет определить базовые понятия в виде макросостояний, их интерпретацию и указать их семантику путем задания соответствующей системы аксиом или определяющих соотношений, которые обычно формулируются в виде совокупности равенств [4, 6].

3 Структурная информация и сложные системы

3.1 Сравнение структурированных состояний сложной системы и их инварианты

Задавая структуру состояний системы некоторой аксиоматикой и определяя наборы допустимых переходов как морфизмы, мы фиксируем лишь «предструктуру» системы. Необходимо иметь возможность сравнивать структуры состояний и способы перехода и отличать подобные и неодинаковые структуры, то есть необходимо иметь отношение порядка и отношение эквивалентности на представляющей систему категории, иметь меру сложности состояний . Необходимый подход для решения данной задачи был предложен в работах [28, 29].

Поиск реально осуществляющихся состояний систем среди всех потенциально возможных может быть осуществлен при условии умения каким-либо образом упорядочить различные состояния системы между собой и определения в полученном упорядочении экстремального из этих состояний. На языке математических структур – это означает найти способ сравнения структурированные множеств, описывающие систему, между собой, а затем, используя некоторый механизм отбора, выбирать из возможных состояний наиболее "сильную" или наиболее "слабую" структуру в качестве той, которой должно соответствовать реально реализующееся состояние из всех допустимых [32].

Для сравнения состояний системы могут использоваться три способа оценки меры сложности системы:

- кардинальные числа несущих множеств состояний (для конечных множеств – количество элементов), описывающих состояния системы;
- установление соответствий между структурированными состояниями системы;

- представление состояний одной системы с помощью состояний другой системы.

Кардинальные числа множеств (мощности множеств) позволяют сравнивать состояния по количеству элементов в них и обычно используются для сравнения бесструктурных множеств. В этом случае мерой сложности состояний служит количество элементов в нем. Для структурированных состояний характеристика по количеству элементов неинформативна, поскольку никак не связана со структурой состояния.

Сравнение структурированных состояний с помощью соответствий (преобразований), сохраняющих имеющуюся структуру, порождает *структурные числа* структурированных состояний, обобщающие понятия кардинального числа, или количества элементов, используемые для сравнения множеств без структуры. В частности, структурированные состояния можно сравнивать с помощью инъективных соответствий. В этом случае структурные числа превращаются в обычные количества элементов для частного случая неструктурированных состояний.

Однако в отличие от бесструктурных состояний, которые всегда сравнимы с помощью числа элементов в них, сравнение структурированных состояний с помощью соответствий в некоторых случаях может оказаться невозможным, поскольку необходимые для сравнения соответствия могут существовать не для любой пары структурированных состояний. Это делает невозможным с помощью соответствий сравнить различные состояния системы между собой и, как следствие, невозможным использовать структурные числа как меру того, насколько далеко находится система от стационарного состояния или состояния равновесия [30, 31, 32].

В теории категорий эту проблему можно обойти, путем использования метода *представлений*. Такие представления из одной категории в другую называются *функторами*. Функтор — это отображение одной категории в другую, сохраняющее категорную структуру системы, когда состояния системы переходят в другие состояния, а морфизмы — в морфизмы и, кроме

того, сохраняются единичные морфизмы и их композиция. Фактически функтор теории категорий – это широко используемый в различных научных дисциплинах *метод аналогий*.

Применительно к системам функтор $F:\Sigma_1 \rightarrow \Sigma_2$ устанавливает соответствие между состояниями и переходами (морфизмами) одной системы Σ_1 и состояниями и переходами другой системы Σ_2. Делается это так, чтобы задаваемые структурой системы Σ_1 связи между состояниями и переходами не были нарушены. Функтор позволяет сопоставлять состояния разных систем. Это сопоставление отражает существенные свойства состояний изучаемой системы, и поэтому с помощью функтора удается свести многие задачи о состояниях одной системы к задачам о состояниях другой более изученной системы.

Так об упорядочении структурированных состояний A и B, обладающих сложной и непривычной структурой, можно судить по упорядочению их образов $F(A)$ и $F(B)$ с более простой или хорошо изученной структурой. Функторный метод представлений предоставляет широкие возможности для сравнения систем, состояния которых порождаются совершенно различной аксиоматикой.

Требуемый для решения многих задач стандартный функтор фактически указан в самом определении категории, так как каждой паре состояний системы A и B сопоставляется *множество* морфизмов $Mor_{\Sigma}(A,B)$, которое, в свою очередь, можно рассматривать как объект категории множеств *Set,* для которой объектами являются множества, а морфизмами – все функциональные отношения между множествами.

Поэтому мощность множества переходов системы из одного состояния в другое (морфизмов) можно сделать универсальным инструментом сравнения структур состояний системы, так как мощность множества всех морфизмов $Mor_{\Sigma}(A,B)$ может быть определена для любой пары структурированных состояний. Этот стандартный функтор из произвольной категории структурированных состояний в категорию

неструктурированных множеств, сопоставляющий состоянию B множество всех морфизмов $Mor_{\Sigma}(A,B)$ из фиксированного состояния A, является монотонным при упорядочении структурированных множеств инъекциями.

Основной результат функторного представления из категории системы в структуру множества ее морфизмов состоит в том, что в системантике естественно возникает внутренне присущая теории категорий числовая функция, характеризующая структурированные состояния системы количеством допустимых этим состоянием переходов. Причем, если структурные числа состояний сравнимы, количества этих переходов упорядочены так же, как структурные числа множеств.

Следовательно, теория категорий позволяет естественным образом ввести количественное описание для систем с самыми различными видами структур, так как представление в кардинальную структуру множеств существует для любых систем, порою весьма далеких от структур числовых множеств.

Как показано в [28] стандартный функтор $F_X{:}\Sigma \rightarrow FinSet$ из произвольной категории системы Σ в категорию неструктурированных конечных множеств $FinSet$, сопоставляющий каждому состоянию A множество всех переходов $Mor_{\Sigma}(X,A)$ из фиксированного состояния X в состояние A, оказывается монотонным при упорядочении структурированных состояний инъекциями. Здесь $FinSet$ – это категория множеств, в которой объектами являются произвольные множества, а морфизмами все функциональные отношения между конечными множествами.

При этом если структура состояния A системы Σ сильнее структуры состояния B, то количество сохраняющих структуру переходов произвольного состояния X в состояние A больше, чем количество таких же переходов из состояния X в состояние B, то есть

$$CardMor_{\Sigma}(X,A) \geq CardMor_{\Sigma}(X,B).$$

Если структуры состояний A и B системы Σ одинаковы, то количества возможных переходов из состояния X в состояние A равно количеству возможных переходов из состояния X в состояние B, то есть

$$CardMor_{\Sigma}(X,A) = CardMor_{\Sigma}(X,B).$$

Количество допустимых морфизмов структурированных состояний системы можно назвать *структурными инвариантами*, или структурными числами состояний системы.

Структурные инварианты представляют собой естественной обобщение понятия количества элементов для структурированных состояний систем [29, 30]. Согласно этому обобщению, сравнение "силы" структурированных состояний системы состоит не в сравнении мощностей базовых для структуры состояний множеств, а в сравнении количеств их преобразований, не нарушающих заданную на множествах структуру.

Таким образом, для сравнения структур состояний A и B системы Σ необходимо уметь подсчитывать число возможных переходов из состояния X в состояния A и B.

3.2 Структурная информация состояний сложной системы

Инвариант $CardMor_{\Sigma}(X,A)$, равный числу потенциальных переходов $Mor_{\Sigma}(X,A)$, можно определить, как структурную информацию состояния A относительно состояния X. Чем больше возможных путей ведут из фиксированного состояния X в состояние A, тем более "богато" содержанием (информацией) состояние системы A относительно состояния X, или, другими словами, тем выше вероятность перехода системы из состояния X в состояние A (реализации состояния А).

Обозначим структурный инвариант состояния A относительно состояния X

$$J^{X}(A) \equiv CardMor_{\Sigma}(X,A).$$

Структурный инвариант $J^X(A)$ есть количество таких переходов структурированного состояния X в структурированное состояние A, при которых они сохраняют задаваемую системой Σ структуру. Переходы из состояния X в состояние A можно рассматривать, как допустимые структурой системы способы получить состояние A из состояния X, а $J^X(A)$ в таком случае есть число таких способов. Таким образом, понятие инварианта состояния системы можно интерпретировать как присущее этому состоянию число *микросостояний*.

Поскольку число состояний, в которые переходит данное состояние при допустимых преобразованиях, в точности равно числу этих переходов, то структурный инвариант оказывается количеством микросостояний, соответствующих макросостоянию системы. Следовательно, содержащаяся в структуре состояний системы информация выражается через микросостояния системы.

Как было показано выше для большинства реальных систем задание семантики системы с помощью определяющих соотношений или ограничений целостности определяет разбиение состояний системы на классы эквивалентности (понятия), что позволяет описывать состояния таких систем с помощью множеств с разбиениями. Совокупность всех возможных разбиений, допускаемых определяющими соотношениями, структурирует фазовое пространство системы Σ, а допустимые переходы сохраняют исходные классы разбиений состояния системы на классы эквивалентности.

Это означает, что переходы из структурированного состояния X в состояние A переводят i класс-понятие K^X_i разбиения структурированного состояния X целиком в определенный единственный $t(i)$ класс-понятие $K^A_{t(i)}$ разбиения структурированного состояния A (рис. 3.1).

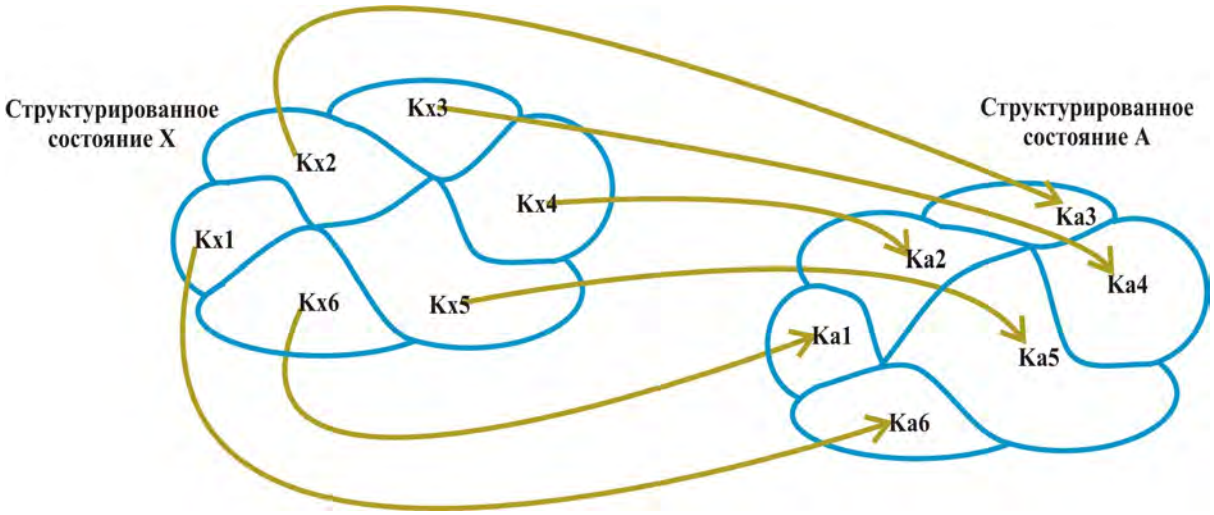


Рис. 3.1 Переходы структурированных состояний системы

Понятийная макроструктура системы в обоих состояниях X и A одна и та же, но допускаются переходы между различными классами-понятиями состояний X и A. При этом структура разбиений состояний данной системы (ее макросостояния) при всех переходах сохраняется, но элементы внутри классов-понятий могут переходить друг в друга. Если число классов разбиения равно w, то структурный инвариант $J^X(A)$ перехода системы из состояния X в состояние A равен произведению числа возможных переходов из класса-понятия K^X_i в класс-понятие $K^A_{t(i)}$

$$J^X(A) = \prod_{i=1}^{w} J^{K_i^X}(K^A_{t(i)}) \ .$$

Если переходы сложной системы это отображения и $CardK_i = n_i$, а $CardK^A_{t(i)} = n_{t(i)}$, то

$$J^X(A) = \prod_{i=1}^{w} n_{t(i)}^{n_i} .$$

Если допустимыми являются не отображения из класса в класс, а какие-либо произвольные отношения между классами, то формула для структурного инварианта сохраняет вид произведения по классам с измененными сомножителями.

В таблице 3.1 приведены формулы для сомножителей структурных инвариантов при переходе системы из состояния A в состояние B при всех возможных комбинациях мультииндекса $\theta \subset \{p,s,f,i\}$ для различных типов отношений [30]. При этом предполагается, что $CardK^A = m$, а $CardK^B = n$.

Табл. 3.1. Структурные инварианты для различных типов отношений состояний системы

№	Значение мультииндекса θ	Содержательное описание	Структурный инвариант
1	***p*** – всюду определенное отношение между классами	Прообраз отношения не имеет пустых вхождений элементов исходного состояния A	$J_p^A(B) = (2^n - 1)^m$
2	***s*** – сюрьективное отношение между классами	Образ отношения не имеет пустых вхождений элементов конечного состояния B	$J_s^A(B) = (2^m - 1)^n$
3	***f*** – функциональное отношение	Прообраз отношения включает не более одного вхождения каждого элемента исходного состояния A	$J_f^A(B) = (n+1)^m$
4	***i*** – инъективное отношение	Образ отношения включает не более одного вхождения каждого элемента конечного состояния B	$J_i^A(B) = (m+1)^n$
5	***p,s*** – всюду определенное сюрьективное отношение	Прообраз отношения не имеет пустых вхождений элементов исходного состояния A. Образ отношения не имеет пустых вхождений элементов конечного состояния B	$J_{p,s}^A(B) = \sum_{P_B \in T_B} J_{p,s,f}^A(P_B)$, здесь T_B — система всех покрытий множества B
6	***p,f*** – всюду определенное функциональное отношение (отображение элементов)	Прообраз отношения включает ровно по одному вхождению каждого элемента исходного состояния A.	$J_{p,f}^A(B) = \mathrm{n}^{\mathrm{m}}$
7	***p,i*** – всюду определенное инъективное отношение	Прообраз отношения не имеет пустых вхождений элементов исходного состояния A.	$J_{p,i}^A(B) = \sum_{k=0}^{n} C_n^k J_{p,s,f}^A(M_k)$, где M_k — произвольное k-элементное

		Образ отношения включает не более одного вхождения каждого элемента конечного состояния *B*	множество, C_n^k - сочетания из n элементов по k
8	***s,f*** – сюръективное и функциональное отношение	Прообраз отношения включает не более одного вхождения каждого элемента исходного состояния *A* Образ отношения не имеет пустых вхождений элементов конечного состояния *B*	$J_{s,f}^{A}(B)=\sum_{k=0}^{m}C_m^k J_{p,s,f}^{B}(M_k)$
9	***s,i*** – сюръективное и инъективное отношение	Образ отношения включает ровно по одному вхождению каждого элемента конечного состояния *B*	$J_{s,i}^{A}(B)=m^n$
10	***f,i*** – функциональное и инъективное отношение	Прообраз отношения включает не более одного вхождения каждого элемента исходного состояния *A* Образ отношения включает не более одного вхождения каждого элемента конечного состояния *B*	$J_{f,i}^{A}(B)=\sum_{k=0}^{n}C_n^k C_m^k k!$
11	***p,s,f*** – всюду определенное сюръективное и функциональное отношение	Прообраз отношения включает ровно одно вхождение каждого элемента исходного состояния *A*. Образ отношения не имеет пустых вхождений элементов конечного состояния *B*	$J_{p,s,f}^{A}(B)=\sum_{k=0}^{n}C_n^k(-1)^k(n-k)^m$
12	***p,s,i*** – всюду определенное сюръективное и инъективное отношение	Прообраз отношения не имеет пустых вхождений элементов исходного состояния *A*. Образ отношения включает ровно одно вхождение каждого элемента конечного	$J_{p,s,i}^{A}(B)=\sum_{k=0}^{m}C_m^k(-1)^k(m-k)^n$

		состояния *B*.	
13	***p,f,i–*** всюду определенное функциональное и инъективное отношение	Прообраз отношения включает ровно по одному вхождению каждого элемента исходного состояния *A*. Образ отношения включает не более одного вхождения каждого элемента конечного состояния *B*	$J^{A}_{p,f,i}(B) = \frac{n!}{(n-m)!}$
14	***s,f,i*** – сюръективное, функциональное и инъективное отношение	Прообраз отношения включает не более одного вхождения каждого элемента исходного состояния *A*. Образ отношения включает ровно по одному вхождению каждого элемента конечного состояния *B*	$J^{A}_{s,f,i}(B) = \frac{m!}{(m-n)!}$
15	***p,s,f,i*** – всюду определенное, сюръективное, функциональное и инъективное отношение	Прообраз отношения включает ровно одно вхождение каждого элемента исходного состояния *A*. Образ отношения включает ровно одно вхождение каждого элемента конечного состояния *B*	$J^{A}_{p,s,f,i}(B) = m!$
16	Произвольное отношение	Отсутствие определенности как для прообраза, так и для образа. Число подмножеств множества $A \times B$	$J^{A}(B) = 2^{nm}$

Структурный инвариант $J^X_\Sigma(A)$ системы Σ зависит как от количества элементов в состояниях X и A, так и от заданной на элементах состояния структуры. Для того чтобы целевая функция системы отражала именно структурные свойства состояния, необходимо вместо общего количества допустимых переходов использовать их удельное количество

$$J^X_\Sigma(A)/J^X_S(A),$$

где $J^X_S(A)$ – это структурный инвариант тех же состояний X и A, но для системы S без структур. Это значит, что понятийная структура системы Σ стерта, разбиения состояний в системе S отсутствуют и они представляются обычными множествами.

Применительно к системе Σ величина удельного структурного инварианта означает, что состояния системы с большими значениями удельного структурного инварианта сильнее структурированы нежели состояния с меньшими значениями инварианта.

Логарифм от величины, обратной числу допустимых переходов в заданное состояние системы (обратной удельному структурному инварианту), называется *структурной информацией* этого состояния системы Σ

$$H^X_\Sigma(A) = log(J^X_S(A)/J^X_\Sigma(A)) = – log(J^X_\Sigma(A)/J^X_S(A)).$$

Так как структурные инварианты состояний, описываемых множествами с разбиениями мультипликативны относительно инвариантов каждого из классов-понятий разбиения, то логарифмы инвариантов, входящие в структурную информацию, аддитивны и имеют характерный для энтропии вид сумм по классам-понятиям разбиения состояний системы.

Структурная информация представляет собой фундаментальную физическую величину, обладающей высокой степенью универсальности и описывающей фундаментальные процессы в динамических системах. Наличие в системе структурной информации есть внутреннее свойство этой системы, которое не зависит от внешних факторов и которое можно определить как информационный ресурс системы [27, 52]. Структурная

информация, определенная как физическая величина, может быть положена в основу создания количественной теории, позволяющей описывать структурные свойства сложных систем, а также использоваться при проектировании систем.

Например, если несущее множество состояний A динамической системы, состоящее из n элементов, разбито на w непересекающихся классов с количеством элементов в i классе n_i

$$n = \sum_{i=1}^{w} n_i \ ,$$

а допустимые переходы – это всюду определенные функциональные отношения ($\theta = \{p,f\}$), переводящие классы-понятия разбиений состояния в себя, то удельный структурный инвариант имеет вид

$$J(n_1, n_2, ...n_w) = \frac{n^n}{\prod_{i=1}^{w} n_i^{n_i}} .$$

Тогда структурная информация состояния системы может быть представлена в виде

$$H(n_1, n_2, ...n_w) = \ln J(n_1, n_2, ...n_w) = -n \sum_{i=1}^{w} \frac{n_i}{n} \ln \frac{n_i}{n}$$

Структурная информация состояния системы является мерой удаленности этого состояния от его бесструктурного аналога, структурная информация которого равна нулю. Представленная конструкция структурной информации не использует каких-либо статистических или вероятностных предпосылок и может быть применена к широкому классу реальных систем. Она может быть строго рассчитана для состояний любых систем, описываемых математическими структурами с большим или малым числом элементов [29].

3.3 Принцип максимума структурной информации для сложных систем

Для применения в теории систем экстремальных принципов оказывается очень важным, что структурные инварианты структурной информации состояний системы в отличие от структурных чисел сравнимы для любых структурированных состояний системы, т.е. экстремальный принцип, сформулированный на языке структурной информации, применим для сравнения любых состояний сложной системы.

Поэтому можно принять следующий постулат, который можно назвать экстремальным принципом динамической системы Σ [29, 31]: из заданного состояния $X \in St\Sigma$ системы Σ в реальности осуществляется переход в то состояние $A \in St\Sigma$, для которого структурная информация $H^X_\Sigma(A)$ максимальна в пределах, допускаемых внешними условиями (например, имеющимися материальными, энергетическими и другими ресурсами).

Согласно экстремальному принципу предпочтительными являются такие состояния динамической системы, которые сильнее других удалены от своих полностью бесструктурных аналогов, т.е. максимально "структурированы".

Инвариант $J^X_S(A)$ при фиксированном X зависит лишь от количества элементов в множестве A. Поэтому, если $CardA$ фиксировано, то энтропия $H^X_\Sigma(A)$ максимальна тогда, когда инвариант $J^X_\Sigma(A)$ минимален. Но малое число переходов из структуры X в структуру A можно трактовать как высокую устойчивость состояния А. Следовательно, экстремальный принцип выделяет такие состояния системы, в которых максимально число элементов в множестве-носителе математической структуры и которые обладают максимальной устойчивостью по отношению к переходам системы.

Экстремальное состояние, достижимое системой, лимитировано возможным наличием трех видов ресурсов: материального, энергетического, информационного. Материальный ресурс является основным в системе. Энергетический ресурс определяет интенсивность процессов в системе. Информационный ресурс выступает в двух ролях. Одна его часть определяет структуру системы и обеспечивает стабильность системы, то есть устойчивость системы к внутренним и внешним возмущениям. Другая – выполняет функции управления, то есть определяет характер процессов, протекающих в системе, а значит, может служить мерой для сопоставления различных состояний системы, служить мерой их изменчивости.

Возможный подход к рассмотрению данной проблемы излагается ниже.

3.4 Сложные динамические системы и ресурсы

Используемые динамической системой ресурсы не только порождают последовательность смены состояний, но и ограничивают степень их изменчивости. Поэтому в экстремальном принципе, порождающем закон изменчивости, экстремум обязательно должен быть условным, и может быть сформулирован следующим образом: из заданного состояния система переходит в такое состояние, для которого структурная информация максимальна в пределах, задаваемых доступными динамической системе ресурсами. Использование условного экстремального принципа позволяет определить конечное стационарное состояние динамической системы, в котором она будет находиться в связи с исчерпанием запасов ресурсов, но не обеспечивает описание нестационарного процесса смены состояний динамической системы.

В качестве примера можно указать на динамические системы, в которых переходы это преобразования множеств состояний X со структурой разбиений на классы эквивалентности $X_1, X_2, \ldots X_w$, с численностями $n_1, n_2, \ldots n_w$. Структура состояния X сохранится, если каждый элемент $x \in X$ при переходе системы Σ в другое состояние перейдет в элемент того же класса эквивалентности. Это означает, что динамическое поведение данной системы Σ описывается перестановками элементов внутри одного класса, или, другими словами, всюду определенными инъективными, сюръективными и функциональными соответствиями.

К данному типу многокомпонентных открытых динамических систем, потребляющих многочисленные ресурсы можно отнести [32]:

- *Технические системы*, состоящие из наборов агрегатов различных классов. Например, система электроснабжения, включающая электродвигатели и другие силовые установки, потребляющие электрическую энергию, эксплуатационные и ремонтные материалы, средства на обновление и ремонт, людские ресурсы и т. д.
- *Экономические системы*, состоящие из классов предприятий, объединенных по некоторому признаку. Например, предприятия отрасли производства, объединенные в группы идентичных по мощности, размеру, потребностям в ресурсах. Предприятия потребляют сырье, материалы, воду, энергию, инвестиции, кредиты, рабочую силу и т. д.
- *Производственные системы* с многопрофильной продукцией, объединяющие производство нескольких типов товаров и потребляющие общие для этих товаров ресурсы.

Для динамических систем, допускающих иньективные, всюду определенные сюръективные и функциональные соответствия ($\theta=\{p,f,s,i\}$), удельный структурный инвариант имеет вид

$$J(n_1,n_2,....n_w)=\frac{n\ !}{\prod_{i=1}^{w} n_i\ !}.$$

Если для больших n воспользоваться асимптотической формулой Стирлинга $\ln n! \approx n\ln n - n$, то структурную информацию можно представить в виде

$$H(n_1,n_2,....n_w)=\ln J(n_1,n_2,....n_w)=n\ln n-\sum_{i=1}^{w} n_i \ln n_i$$

Условный экстремум структурной информации для динамических систем, ограниченных по ресурсам, влечет неоднородные распределения. Причем степень их неоднородности может быть сколь угодно велика в зависимости от различий компонентов исследуемой системы по "потребностям" в ресурсах, ограничивающих развитие [32].

Применительно к динамической системе рассмотренной выше соответствующая вариационная задача на условный экстремум выглядит следующим образом [14, 57]:

$$\begin{cases} H(n_1,....n_w)=(\sum_{i=1}^{w} n_i)\ln(\sum_{i=1}^{w} n_i)-\sum_{i=1}^{w} n_i \ln n_i \to \max; \\ \sum_{i=1}^{w} r_i^k n_i \le R^k, k=\overline{1,m}; \\ n_i \ge 0, i=\overline{1,w}; \end{cases}$$

где

$n=\sum_{i=1}^{w} n_i$, n_i – искомые элементы каждого из классов макросостояния системы в стационарном состоянии, w – число классов в макросостоянии;

r_i^k – количество k-го ресурса, потребляемое i классом из среды;

m– общее количество различных ресурсов, потребляемых динамической системой;

R^k – доступное системе количество ресурса k в среде ($R^k \geq 0$).

Данная задача является математической формализацией постулируемого экстремального принципа, согласно которому динамическая система из заданного состояния переходит в состояние с экстремальной (в пределах, допустимых имеющимися ресурсами) структурой. Искомыми функциями являются численности классов n_i и их сумма n.

Экстремальность функционала $H(\vec{n})$, порождающего динамику системы, можно интерпретировать и как требование максимального разнообразия системы, ограниченного доступными ресурсами среды.

Для любого вектора ресурсов $\vec{R} = (R_1,...R_m)$ решение сформулированной задачи на условный экстремум существует, и задается формулой

$$n_i = n\exp(-\sum_{k=1}^{m}\lambda^k r_i^k),$$

а величина n и множители Лагранжа λ^k являются функциями ресурсов $\vec{R} = (R_1,...R_m)$ и отыскиваются из системы

$$\begin{cases} \sum_{i=1}^{w} \exp(-\sum_{k=1}^{m}\lambda^k r_i^k) = 1; \\ \lambda^k (n\sum_{i=1}^{w} r_i^k \exp(-\sum_{j=1}^{m}\lambda^j r_i^j) - R^k) = 0 \ , k = \overline{1,m}; \\ \lambda^k \geq 0 \end{cases}$$

Из заданного состояния система переходит в такое состояние, для которого потребление ограничивающих рост ресурсов минимально в пределах, задаваемых необходимой степенью структурированности системы. Видно, что количество элементов в каждом из классов макросостояний

экспоненциально зависят от наличных ресурсов, что сильно ограничивает число классов макросостояний *w*.

Несмотря на продемонстрированные аналитические возможности анализа поведения сложных систем с помощью теории категорий, вне ее поля зрения остается очень важный вопрос, связанный с механизмом обеспечения взаимодействий между компонентами системы. Практически теория категорий абстрагируется от данного вопроса, не предоставляя для рассмотрения данной проблемы адекватных средств. В объектно-ориентированном моделировании такие понятия с самого начала вводятся явно в виде передаваемых между компонентами системы сообщений.

4 Понятийные средства системантики

4.1 Семиотическая модель системантики

Системантику и семиотику объединяет то обстоятельство, что в них речь идет об описании сложных систем посредством определения адекватных понятий и их комбинирования в соответствии с определенными правилами. Способ организации используемых понятийных средств в виде языка с фиксированным синтаксисом и семантикой получил распространение в значительной степени в математической логике и программировании.

Для сложных систем, когда априорная информация об их структуре и функционировании обладает значительной неполнотой, необходимо строить модели семиотического типа. В этом случае основой представления знаний о системе является язык подходящей абстрактной системы. Этот язык по своим изобразительным возможностям должен быть достаточно богат, чтобы в семиотической модели адекватно отображать сведения о структуре системы, среде ее функционирования и протекающих в них процессах. Такой язык должен быть достаточно точным и формальным.

Так, например, в исчислении предикатов и языках программирования для этого широко используются конструктивные формальные теории, основанные на математической логике и абстрактной грамматике. Аналогично, в системантике для описания сложных систем может быть использована некоторая *абстрактная система* (*формальная система* [51]*) формальная логико-лингвистическая модель* [41] *или теория* [55. 17], которая может быть определена без ссылки на какую-либо конкретную реализацию своих частей и представлена в виде тройки

$$AS = \langle N, Синт, Сем \rangle,$$

где

- N — конечное множество имен терминальных объектов (словарь имен), описывающих сложную систему;

- *Синт* — синтаксические правила (синтаксис абстрактной системы), определяющие грамматику формирования сложных выражений абстрактной системы из простых;
- *Сем* — совокупность семантических правил абстрактной системы.

Множество N определяет словарь понятий абстрактной системы, которые используются для ее описания и построения более сложных (производных) конструкций теории. Имена составляются из символов некоторого алфавита A, в качестве которого могут выступать буквы или цифры. Элементами множества N могут, например, быть слова и словосочетания естественного языка, являющиеся именами понятий. Использование в абстрактной системе имен позволяет в описании эффективно выражать совместное использование частей разными компонентами системы, не требуя их непосредственного дублирования. При преобразованиях такие части можно рассматривать как независимые сущности.

Над элементами множества N можно выполнять определенные операции или функции. Каждое имя понятия соотносится с некоторым классом, указывающим в операциях какого типа это имя можно использовать.

Синтаксические правила Синт используются для того, чтобы из базовых понятий строить такие их совокупности, которые в рамках данной абстрактной системы считаются синтаксически правильными. Чтобы точно описать синтаксис абстрактной системы, надо задать синтаксис ее выражений, которые в математической логике принято называть термами или формулами. Синтаксис может быть задан абстрактной грамматикой, определяющей более абстрактные синтаксические объекты абстрактной системы - правильно построенные формулы (*ППФ)*, которые определяются следующим образом.

В множестве имен N выделяется конечное множество *имен констант* $Const = \{C_1, C_2 \cdots C_m\}$, переменных $Var = \{X_1, X_2 \cdots X_k\}$, символов операций $Op = \{F_1, F_2 \cdots F_k\}$ и символов отношений $Rel = \{R_1, R_2 \cdots R_l\}$, а также набор имен множеств $Sort=\{T_1, T_2 \cdots T_n\}$, которые называются *сортами или типами.*

Для каждого имени операции F_i указывается набор имен типов $T_{i1}, \cdots T_{i\alpha_i}$ из которого операция действует, и имя типа $T_{t(i)}$ результата операции

$$F_i: T_{i1} \times T_{i2} \times \cdots \times T_{i\alpha_i} \to T_{t(i)}.$$

Тип операции определяется типом ее результата.

Для каждого имени отношения R_j указывается набор имен типов T_{j1}, $\cdots T_{j\beta_j}$, на котором это отношение определено

$$R_j \subseteq: T_{j1} \times T_{j2} \times \cdots \times T_{j\beta_j}.$$

В результате каждому символу операции и отношения приписывается некоторое натуральное число – арность этого имени.

На множестве переменных Var определяется отображение $Type: Var \to Sort,$ которое каждой переменной $x \in X_i$ приписывает некоторый тип T_i, если $Type(x) = T_i$.

В рамках синтаксических объектов абстрактной системы сначала определяется множество слов, которые называются термами. Термы играют роль существительных и местоимений и обычно интерпретируются как имена понятий абстрактной системы. Такое использование получают, прежде всего, переменные, которые поэтому и относятся к термам. Более точно, множество термов $Term$ определяется следующим образом:

1. Константы и переменные типа T_i являются термами типа $Term_i$. Это значит, что каждый терм имеет определенный тип.
2. Если F_i – имя операции арности α_i и $t_1, t_2 \cdots t_{\alpha_i}$ термы типа $T_{i1}, \cdots T_{i\alpha_i}$ соответственно, то $F_i(t_1, t_2 \cdots t_{\alpha_i})$ – терм типа $T_{t(i)}$.
3. Других термов типа $Sort$ в AS нет.

Следовательно, множество термов абстрактной семиотической системы определяется равенством

$$Term = \bigcup_{i=1}^{n} Term_i .$$

Теперь можно определить множество *ППФ* и множество свободных и связанных переменных. В отличие от термов правильно построенные формулы – это выражения, которые интерпретируются как некоторые утверждения о свойствах системы. Множество правильно построенных формул определяется следующими правилами:

1. Каждый терм является правильно построенной формулой, то есть $Term \subset ППФ$.
2. Если имя отношения $R_j \in Rel$ имеет арность β_j и $t_{1,}\, t_2 \cdots t_{\beta_j}$ термы типа $T_{j1}, \cdots T_{j\beta_j}$, то $R_j(t_{1,}\, t_2 \cdots t_{\beta_j}) \in ППФ$, в которой свободны все переменные, которые свободны в термах $t_{1,}\, t_2 \cdots t_{\beta_j}$.
3. Если Q и R – ППФ, то $Q \vee R$, $Q \wedge R$, $Q \Rightarrow R$, $Q \Leftrightarrow R$ и $\neg R$ являются ППФ. В этих формулах переменная x свободна, если она свободна в одной из формул Q или R.
4. Если Q формула со свободной переменной x, то $(\exists x)Q$ и $(\forall x)Q$ – формулы, в которых переменная x связана.
5. Других формул в *ППФ* нет.

Среди множества формул мы выделяем такие, которые не содержат свободных вхождений ни одной переменной. Такие формулы называются *предложениями*. Роль предложений состоит в том, что вполне определенные утверждения выражаются не произвольными формулами, а лишь такими, которые являются предложениями. Но так как сложное предложение о свойствах системы может быть построено из компонент, которые сами по себе не являются предложениями, а всего лишь формулами, то непосредственное определение понятия предложения без промежуточного понятия формулы представляется неестественным [35].

Таким образом, исходные элементы, из которых строятся правильно построенные формулы, – это простые, атомарные выражения или термы. К ним, например, относятся выражения вида $x = y$, где x и y – переменные, некоторый набор функций $F_i(t_1, t_2 \cdots t_{\alpha_i})$ и отношений $R_i(t_1, t_2 \cdots t_{\beta_i})$ и выражения, получаемые из них подстановкой вместо одних термов других. Произвольные правильно построенные формулы строятся из этих атомарных с помощью логических связок $\vee, \wedge, \Rightarrow, \Leftrightarrow, \neg$ и кванторов $\exists$, $\forall$.

Введенные выше понятия определяют некоторый язык L_{as} абстрактной системы, состоящий из термов *Term* и правильно построенных формул *ППФ*, который может быть использован для абстрактного представления системы Σ в терминах имен множеств термов *Term*, имен операций F_i, имен отношений R_j и соотношений *Сем* между операциями, выражающими семантику абстрактной системы.

Правильно построенные формулы могут иметь различное, или одно и то же содержание, одну и ту же семантику, один и тот же смысл. *Семантика* или то значение, которое мы приписываем правильно построенным формулам, определяется конкретной их *интерпретацией (Int)* или присваиванием значений в виде выбранных состояний системы. Таким образом, придание *ППФ* определенных значений с помощью состояний системы, зависит от точки зрения исследователя.

Очевидно, что несколько *ППФ* могут иметь одну и ту же семантику (смысл), интерпретироваться одинаково (синонимия), и, наоборот, одна формула может иметь несколько смыслов, несколько интерпретаций (омонимия). Но не всем *ППФ*, построенным в соответствии с указанными выше правилами, можно придать конкретный смысл или дать соответствующую интерпретацию.

Формулы, имеющие одно и то же содержание, одну и ту же интерпретацию, эквивалентны. Определение эквивалентных формул путем их интерпретации с помощью состояний вызывает определенную трудность,

так как требует введения состояния системы Σ и необходимость предвидеть все потенциальное множество состояний.

Для того чтобы ограничить множество *ППФ* только теми *ППФ*, которые выражают свойства и закономерности некоторой реальной системы, на множество *ППФ* накладывается набор ограничений, выражаемых с помощью семантических правил *Сем*.

Решить данную проблему можно двумя способами: либо семантически, либо синтаксически [35]. В первом случае описание абстрактной системы *AS* состоит из всех формул языка L_{as}, истинных в любой интерпретации из фиксированного набора интерпретаций. Во втором случае описание абстрактной системы *AS* состоит из всех формул, которые могут быть выведены в соответствии с некоторым фиксированным отношением синтаксического следования из некоторого определенного множества формул, называемых аксиомами.

В математической логике для ограничения множества *ППФ* только теми *ППФ,* которые соответствуют реальным системам, чаще используется аксиоматический подход, когда аксиомы, по существу, выражают правила определения эквивалентности правильно построенных формул, не требующих обращения к состояниям. Он основан на использовании подходящего исчисления.

Для этого, некоторые формулы объявляются *аксиомами* (*Ax*), и указываются *правила вывода (Inf)*. Две формулы $u \in$ *ППФ* и $v \in$ *ППФ* объявляются эквивалентными, если формула $(u \rightarrow v)\&(v \rightarrow u)$ выводима из аксиом *Ax*. Причем доказывается, что формулы u и v тогда и только тогда эквивалентны в синтаксическом смысле, когда они семантически эквивалентны, то есть когда их интерпретации совпадают.

Таким образом, совокупность семантических правил *Сем* включает множество *Ax* аксиоматических соотношений абстрактной системы и множество правил вывода *Inf*, позволяющих на основе множества исходных аксиом *Ax* строить (выводить) другие правильно построенные формулы.

Аксиомы *Ax,* выражающие семантические ограничения системы (статические и динамические), можно разделить на две части

$$Ax = <Ax_{st}, Ax_{dyn}>.$$

Здесь Ax_{st} – определяют ограничения на статические состояния системы, а Ax_{dyn} – определяют ограничения на траектории системы (динамические ограничения), то есть накладывают ограничения на взаимосвязь состояний системы в различные моменты времени.

Правила вывода *Inf* расширяют множество аксиом, добавляя к ним новые правильно построенные формулы. Правильность таких формул означает выводимость заключений из исходных предположений, или из того подмножества исходных предположений, которое необходимо для составления цепи рассуждения.

В результате применения синтаксических и семантических правил в рамках абстрактной системы порождаются некоторые тексты на языке L_{as}, которые представляют *абстрактное* (идеальное, логико-лингвистическое, информационное) описание сложной системы.

4.2 Семантика абстрактной семиотической модели

Правильно построенные формулы являются чисто синтаксическими образованиями и от них мало пользы до тех пор, пока не будет придан смысл употребляемым в них именам, то есть пока не будет задана модель реализации системы, определяющая семантику имен абстрактной семиотической модели. Именам понятий необходимо сопоставить конкретные множества сущностей (состояний системы), а знакам операций – конкретные функции системы.

Для решения данной задачи состояния системы $St\Sigma$ разбиваются на *n базовых типов состояний* $St\Sigma = \bigcup_{i=1}^{n} s_i$ и устанавливается взаимооднозначное отображение между типами T_i абстрактной системы и базовыми состояниями

s_i системы, а также между операциями абстрактной системы $F_i(t_1, t_2 \cdots t_{\alpha_i})$: $T_{i1} \times T_{i2} \times \cdots \times T_{i\alpha_i} \to T_{t(i)}$ и функциями $f_i(q_1, q_2 \cdots q_{\alpha_i})$: $s_{i1} \times s_{i2} \times \cdots \times s_{i\alpha_i} \to s_{t(i)}$ системы Σ.

Каждому отношению $R_j(t_1, t_2 \cdots t_{\beta_j}) \subseteq: T_{j1} \times T_{j2} \times \cdots \times T_{j\beta_j} \in Rel$ абстрактной системы ставится в соответствие предикат $r_j(q_1, q_2 \cdots q_{\beta_j})$ арности β_j, который принимает значение *true*, если $r_j(q_1, q_2 \cdots q_{\beta_j})$ принадлежит области $s_{j1} \times s_{j2} \times \cdots \times s_{j\beta_j}$, и *false* в противном случае.

В результате нами определена алгебраическая система

$$AC = <s, func, rel>,$$

где

$s = \{s_1, s_2 \quad s_n\}$ – множество сортов,

$func = \{f_1, f_2 \quad f_k\}$ – множество функций,

$rel = \{r_1, r_2 \quad r_l\}$ – множество отношений. Эту алгебраическую систему можно рассматривать как реализацию абстрактной системы, если она удовлетворяет множеству исходных аксиом *Ax*. Такая алгебраическая система называется моделью *Mod* абстрактной системы *AS*.

Чтобы вычислить результат применения абстрактных операций $F_i(t_1, t_2 \cdots t_{\alpha_i}) \in Op$ необходимо указать оценку (значения) переменных x в термах $t_1, t_2 \cdots t_{\alpha_i}$ в рамках принятой алгебраической модели. Для этого определим отображение

$$ev: X \to St\Sigma,$$

ставящее в соответствие каждой переменной $x \in X_i$ некоторое значение $q \in St\Sigma$

$$ev(x) = q \in St\Sigma.$$

Чтобы для заданной алгебраической модели системы Σ вычислить результат применения операций $F_i(t_1, t_2 \cdots t_{\alpha_i})$, надо в терме $F_i(t_1, t_2 \cdots t_{\alpha_i})$ заменить операцию F_i на функцию f_i, а термы t_i соответствующими значениями $ev(t_i)$ и выполнить действия, определяемые термом $f_i(ev(t_1), ev(t_2) \cdots ev(t_{\alpha_i}))$. Следовательно, терм $F_i(t_1, t_2 \cdots t_{\alpha_i})$ становится функцией из $s_{i1} \times s_{i2} \times$

$\cdots \times s_{i\alpha_i}$ в $s_{t(i)}$. Если в терме нет переменных, то его значение постоянно, и оно вычисляется как результат выполнения операций над подходящими константами.

Аналогично, если с каждым отношением $R_j(t_1, t_2 \cdots t_{\beta_j})$ связывается определенный предикат $r_j(t_1, t_2 \cdots t_{\beta_j})$ и заданы значения переменных, входящих в термы $t_1, t_2 \cdots t_{\beta_j}$, то производится вычисление значений термов, а выражение $r_j(\text{ev}(t_1), \text{ev}(t_2) \cdots \text{ev}(t_{\beta_j}))$ принимает значение *true* или *false* на $s_{j1} \times s_{j2} \times \cdots \times s_{j\beta_j}$.

Таким образом, как только задана интерпретация имен переменных и констант абстрактной семиотической модели, все термы и *ППФ* получают соответствующую оценку путем вычислений, превращаясь в операции и предикаты на множестве состояний реальной системы $St\Sigma = \bigcup_{i=1}^{n} s_i$.

Из сказанного становится ясно, что необходимо различать *абстрактную (глубинную) семантику* семиотической системы, складывающуюся из смыслов элементарных имен (понятий) и представленную ее аксиомами или соотношениями в виде правил, и *конкретную (формальную) семантику*, которая определяется интерпретацией текста, сформулированного на языке абстрактной системы, смысл которой выявляется путем конкретных вычислений (конкретной реализацией) на множестве состояний системы Σ. Если выбрана некоторая абстрактная система, то любой текст, построенный по законам синтаксиса этой системы, будет грамматически правильным, но не всякий такой текст обязан иметь смысл, допускать определенную интерпретацию.

С другой стороны, для данной абстрактной системы может существовать много моделей, позволяющих интерпретировать тексты (*семантический аспект*) или много моделей возможного понимая текста на языке L_{as} адресатом (*прагматический аспект*). Смысл текста всегда

рассматривается по отношению к некоторому классу моделей, то есть возможных интерпретаций. Текст, для которого можно построить модель, в которой он получает определенное значение, является семантически осмысленным.

Таким образом, аксиомы абстрактной системы *Ax* сами по себе не истины и не ложны, но они становятся некоторыми высказываниями при сопоставлении абстрактной системы с некоторой ее реализацией, которая задается *моделью (Мод)* абстрактной системы

$$Мод = <St\Sigma, ev, Int>,$$

где

$St\Sigma$ – это множество состояний системы Σ;

$ev: N \rightarrow St\Sigma$ – правила отображения (оценки), устанавливающие семантику имен исходных понятий в виде взаимосвязи между множествами имен N и состояний системы $St\Sigma$;

$Int: ППФ \rightarrow St\Sigma$ – правила интерпретации, позволяющие любой правильно построенной формуле приписать некоторое состояние или класс состояний системы.

Функция $Int: ППФ \rightarrow St\Sigma$, задающая для каждого текста на языке L_{as}, состоящего из правильно построенных формул, определяемых грамматикой *Синт* и семантикой *Сем* абстрактной системы *AS,* некоторый класс состояний системы Σ, если определены оценки для всех исходных понятий абстрактной системы, определяет математический объект, который является значением (смыслом, семантикой, денотатом и т. д.) данного текста.

Формулы абстрактной системы *AS* при их сопоставлении с моделью *Мод* становятся высказываниями об этой модели. Модель *Мод* является моделью абстрактной системы *AS*, если сигнатура модели совпадает с сигнатурой абстрактной системы, и если в результате интерпретации каждая аксиома абстрактной системы становится истинным высказыванием, то есть выполняется для данной модели.

Главная особенность абстрактной системы состоит в том, что в ней не определяются несущие множества. Абстрактная система описывает некоторые свойства объектов, но не указывает сами эти объекты. Напротив, в модели *Мод* есть все, что есть в абстрактной системе, и, кроме того, добавляется набор несущих множеств. В результате каждый символ сигнатуры абстрактной системы приобретает конкретное содержание, то есть превращается в реальную операцию или отношение на несущих множествах модели *Мод.* В абстрактной системе есть операции, отношения и их свойства, но нет еще множеств, где эти операции и отношения выполняются.

Таким образом, абстрактная система *AS* – это перечень названий операций, отношений и свойств этих операций и отношений, а модель модели *Мод* – множество, на котором заданы операции, отношения и выполнены требуемые свойства. Как уже указывалось ранее в качестве модели *Мод* целесообразно использовать представление сложной системы в виде алгебраической системы.

Модель абстрактной системы *Мод* задает внутреннее (материальное), описание сложной системы, описывает одну из возможных ее реализаций, а пара

$$CM = <AS, Мод>,$$

состоящая из абстрактной системы *AS* и модели *Мод,* представляет семиотическое описание сложной системы в целом, которую можно назвать семиотической моделью системантики, которая может быть представлена в виде семерки

$$CM = <N, Синт, Ax, Inf, St\Sigma, ev, Int>$$

Семиотическая модель системантики СМ описывает сложную систему с двух точек зрения: с точки зрения формального текста на языке L_{as}, который позволяет строить формальные выражения для выявления новых взаимосвязей системы, используя аксиомы и правила вывода, и с точки зрения конструктивной модели, в рамках которой непроцедурные выражения абстрактной системы *AS* рассматриваются как высказывания относительно

конкретных свойств системы. Интерпретация таких высказываний предполагает их перевод из непроцедурной формы на процедурный язык алгебраических операций модели *Мод* и извлечение конкретных знаний о конструкции системы, зафиксированных в модели сложной системы. Тем самым, сложную систему можно рассматривать как автомат, преобразующий непроцедурные (входные) выражения абстрактной системы $L_x \in L_{as}$ в конкретные (выходные) значения свойств и закономерностей реальной системы L_y.

$$<L_x, St\Sigma, L_y>$$

Это значит, что семиотическая модель сложной системы представляет знания о системе в виде ее виртуального представления, которое может фиксироваться в виде реализации некоторой информационной модели на ЭВМ. При этом абстрактная система *AS* является непроцедурным интерфейсом такого виртуального представления, а модель *Мод* – ее базой данных, состояния которой моделируют состояния объектов сложной системы.

Построить семиотическую модель сложной системы – это значит, прежде всего, подобрать точные и адекватные описываемой системе понятия. Работа с понятиями – главное в построении и использовании семиотической модели сложной системы. Ниже мы рассмотрим понятийные средства, которые могут быть использованы для построения семиотической модели, описывающей структуру и поведение сложной системы.

4.3 Понятийные средства представления знаний о системах

Основной целью познавательного процесса является обобщение экспериментальных данных путем наблюдения. В полной мере это относится и к анализу систем. При этом важнейшее значение имеют выделение и использование элементарных базовых понятий, с помощью которых мы производим описание системы. Любое адекватное представление системы

состоит в установлении определенных отношений между некоторой совокупностью таких базовых понятий. Например, биологическая классификация животных включает такие основные понятия, как черви, рыбы, птицы, млекопитающие и т. д., которые разбивают все классифицируемое множество животных на отдельные виды, а они в свою очередь могут быть разделены на более узкие понятия. Отношения между понятиями также могут рассматриваться как новые понятия.

Понятие выступает как основная единица любой интеллектуальной деятельности. Оно отражает глубинные стороны изучаемых явлений. Иначе, понятие есть базовая конструкция при представлении знаний о системе. Выделение понятий осуществляется путем абстрагирования от конкретных особенностей данного явления или процесса.

Понятия именуются с помощью слов или словосочетаний естественного языка, которые играют роль знаков или имен. Понятие может относиться к множеству однотипных элементов системы или к конкретному единичному элементу.

В процессе описания системы, кроме понятий, обычно выделяется некоторая совокупность связей или отношений между понятиями. Выделенные отношения связывают между собой отдельные факты, объекты, явления, события или процессы. При этом возможны два принципиально различных подхода.

Первый подход подчеркивает доминирующую роль связей между понятиями. В этом случае семантика связи раскрывается через концепцию роли участвующих в связи понятий [36]. Можно считать, что роль выступает в качестве имени соответствующего свойства данной связи, а значениями этого свойства являются экземпляры связываемых понятий. Например, для связи РУКОВОДИТЬ между двумя сущностями понятия ЛИЧНОСТЬ могут быть выделены две роли, одна из которых — НАЧАЛЬНИК, а другая — ПОДЧИНЕННЫЙ. Семантика связи РУКОВОДИТЬ, а следовательно, и семантика предметной области описываются с помощью ролей

НАЧАЛЬНИК и ПОДЧИНЕННЫЙ. Основную семантическую нагрузку в описании системы в этом случае несут выделенные связи и те роли, которые играют связываемые понятия.

Описание системы, используемое при *втором* подходе, не требует введения концепции роли. Основное внимание, наоборот, уделяется использованию понятий; связи играют хотя и важное, но вспомогательное значение. В этом случае первичными элементами описания системы являются понятия, а связи используются как *конструкторы*, позволяющие создавать сложные понятия из более простых. Так, в рассматриваемом ранее примере для связывания отдельных экземпляров сущностей понятия ЛИЧНОСТЬ достаточно предусмотреть введение дополнительного признака РУКОВОДИТЕЛЬ.

Если при первом подходе семантика структуры системы раскрывается с помощью связей и ролей, которые играют понятия в связях, то при втором — главное значение для представления семантики состояния системы имеют понятия, а связи фактически выступают в качестве дополнительных признаков (валентных свойств) используемых понятий.

4.4 Имена, денотаты и интерпретация понятий

Каждому понятию, используемому для построения формального описания сложной системы, приписывается некоторое уникальное *имя* или *знак*. Одна из функций имени состоит в указании обозначаемого понятия, то есть имя можно рассматривать как синтаксический двойник понятия. Имена дают прямой способ идентификации понятий, с помощью имени понятия можно различать даже в том случае, когда их признаки совпадают.

В качестве имен понятий обычно используются любые слова или словосочетания естественного языка. Следовательно, имена замещают понятия. При этом между замещаемым понятием и его именем возникает определенная связь, которую обычно называют обозначением, или ссылкой

на именуемое понятие. Пара, состоящая из имени и обозначаемого, в семиотике называется знаковой ситуацией. Объекты или сущности сложной системы, на которые можно ссылаться с помощью имени или знака, называются *денотатами*, то есть денотат обеспечивает возможность оценки $ev: N \rightarrow St\Sigma$ данного понятия.

Денотат понятия — это значение, которое может иметь понятие в рамках определенного контекста [53]. В качестве такого значения может выступать, например, конкретная сущность (предмет), или объект, на которые указывает данное имя. Другими словами, денотат понятия является способом интерпретации данного имени в рамках некоторой ситуации, рассматриваемой в системе, то есть определяет смысл или семантику понятия.

Можно указать на следующие основные свойства знаковой ситуации:

- имена понятий способны замещать денотаты. Например, имя понятия АВТОМОБИЛЬ может использоваться в качестве заместителя любого конкретного автомобиля;
- имя понятия нетождественно денотату, оно не может полностью заменить денотат. Так, при алгебраических преобразованиях мы можем использовать буквенные обозначения чисел. Однако если необходимо вычислить числовое значение выражения, то требуется подставить конкретные числа (денотаты) вместо букв;
- связь «имя — денотат» многозначна, т. е. некоторое имя может обозначать множество денотатов (может иметь несколько интерпретаций) — омонимия, и, наоборот, одному денотату (одной интерпретации) можно поставить в соответствие несколько имен — синонимия.

4.5 Признаки, объем и схема понятия

Признаки выражают устойчивые, внутренне присущие понятию черты. Их выделение, зачастую, происходит на интуитивном уровне. Простое понятие часто воспринимается как носитель характеризующих его признаков. Семантика простого понятия проявляется в этих признаках, и они неотделимы от него.

Признаковые отношения предписывают одним понятиям выполнять роль некоторых качественных свойств по отношению к другим понятиям. Признаки понятий могут быть отнесены к одному из следующих типов: *дифференциальные, характеристические* и *валентные* (рис 4.1).

Рис. 4.1. Структура понятия

Дифференциальные признаки используются в качестве характеристики содержания понятия. *Характеристические* — это признаки, которые позволяют отличить сущности, относящиеся к объему одного и того же понятия. *Валентные* — это признаки, обеспечивающие связь между различными понятиями. Без потери общности можно считать такие связи бинарными.

Признак характеризуется именем и значением определенного типа. Можно выделить несколько типов значений признаков: логические, числовые, символьные и др. Имя признака вместе с его значением, оценкой образует полное наименование соответствующего признака. Например, признак ВОЗРАСТ вместе со значением «30 лет» образует наименование признака «возраст 30 лет».

Имя признака позволяет указать ту семантическую роль, которую играет его значение в организации связи между сущностью и признаком, ее характеризующим. Так, в предыдущем примере имя признака ВОЗРАСТ

характеризует определенную роль понятия «30 лет» по отношению к некоторой сущности, в качестве которой выступает конкретный человек.

Важной характеристикой понятия является *объем понятия*. Объем понятия — это множество (класс) всех денотатов (сущностей), обладающих существенными признаками понятия. Другими словами, объем понятия включает все предметы или объекты сложной системы, которые могут быть описаны данным понятием. Сущности, составляющие объем понятия, различаются с помощью признаков.

Признаки позволяют разбить некоторое множество объектов на классы эквивалентности *E*, которые определяются так. Будем считать, что пара объектов *(a, b)* принадлежит *E* тогда и только тогда, когда *a* и *b,* принадлежащие объему понятия *P*, обладают одним и тем же признаком. При таком определении отношения *E* выполняются законы рефлексивности, симметричности и транзитивности, а множество объектов, составляющих объем понятия, разбивается на классы эквивалентности. Классы эквивалентности образуют фактормножество данного множества по отношению *E.*

Совокупность имен дифференциальных, характеристических и валентных признаков составляет схему понятия, обозначаемую как *shmP*. Таким образом, схему понятия *P* можно представить в виде тройки

shmP = <*D, H, V*>,

где

$D=\{D_i\}$, $i=\overline{1,n}$ — множество имен дифференциальных признаков;

$H=\{H_j\}$, $j=\overline{1,m}$ — множество имен характеристических признаков;

$V=\{V_k\}$, $k=\overline{1,l}$ — множество имен валентных признаков.

Тот факт, что признак A_i данного понятия принимает одно из возможных значений $a_j^i \in \mathrm{dom}A_i$, будем выражать в виде пары (A_i, a_j^i). Здесь $\mathrm{dom}A_i$ обозначает множество (домен) всех возможных значений признака A_i. Тогда каждая сущность e_q, принадлежащая объему понятия, может быть

представлена в виде множества пар дифференциальных, характеристических и валентных признаков

e_q=<{$(D_i, d_q^i \mid d_q^i \in \text{dom}D_i,\ i=\overline{1,n})$}, {$(H_j, h_q^j \mid h_q^j \in \text{dom}H_j,\ j=\overline{1,m})$}, {$(V_k, v_q^k \mid v_q^k \in \text{dom}V_k,\ k=\overline{1,l}$}>.

Каждое имя понятия, кроме того, что оно может замещать некоторую конкретную сущность, связывается с теми общими свойствами, или признаками, которые присущи всем сущностям, составляющим множество всех денотатов данного понятия.

Познать некоторый сложный объект означает установить закономерность его строения, то есть выделить составляющие его простые компоненты и сформулировать правила, по которым эти компоненты соединяются между собой. В результате появляется именно схема изучаемого объекта.

4.6 Концепт, экстенсионал и интенсионал понятий

Совокупность правил, позволяющих идентифицировать данное понятие, называется *концептом*, или *содержанием*, понятия [17], признаки образующих/. Концепт понятия указывает необходимую информацию, с помощью которой данная сущность или элемент системы могут быть отнесены к некоторому множеству однотипных сущностей, т. е. концепт содержит знания о потенциальном денотате и характеризует определенный класс возможных денотатов [53]. Обычно эти правила могут быть выражены в виде некоторых соотношений (равенств), накладываемых на дифференциальные признаки понятия, которым должны удовлетворять его денотаты.

В концепте понятия отражается вся информация, необходимая для принятия решения об отнесении некоторой сущности к денотату понятия. В более сложных случаях, концепт включает совокупность правил, необходимых и достаточных для принятия решения о принадлежности

данной сущности денотату понятия. Это означает, что концепт содержит семантическую информацию, является носителем семантики понятия. Другими словами, концепт — это то знание, которое выражается данным понятием при описании системы.

Концепт понятия включает в себя как все собственные признаки понятия, так и те признаки, которые позволяют установить связь данного понятия с другими понятиями (валентные признаки), т. е. концепт включает схему строения понятия. Если понятие является сложным, то его концепт содержит также информацию о концептах понятия, входящих в данное понятие.

В семиотике для характеристики понятия получили распространение категории *денотата* и *концепта*, а в логике для тех же целей служат термины *экстенсионал* и *интенсионал*.

Множество всех элементов, объектов, предметов или сущностей, являющихся денотатами понятия, составляет его объем, или экстенсионал понятия. Если экстенсионал понятия *P* обозначить через *extP*, то можно записать выражение

$$extP = \{e_1, e_2, \ldots e_n\},$$

где $e_1, e_2, \ldots, e_n$ — сущности, являющиеся денотатами понятия *P*.

Экстенсионал понятия — это совокупность всех его допустимых денотатов, соответствующих концепту этого понятия в рамках используемой интерпретации. Так, чтобы описать экстенсионал понятия АВТОМОБИЛЬ, следует рассмотреть класс всех автомобилей.

Синонимичные понятия могут иметь один и тот же экстенсионал. Хотя они имеют общий экстенсионал, однако выражают различную информацию (различное знание), которую мы связываем с их *интенсионалами*. Так, на множестве точек «биссектриса угла» истинны два утверждения: «равноудалены от сторон данного угла» и «делят данный угол на два равных». Следовательно, понятие не полностью характеризуется

экстенсионалом. Необходимо учитывать также интенсиональный аспект понятия, который выражается его концептом.

Интенсионал понятия — это то знание, которое мы вкладываем в данное понятие, т. е. интенсионал характеризует концепт данного понятия, его семантику или содержание. Интенсионал понятия *P* будем обозначать *intP*.

Связи между различными категориями, используемыми для описания понятий в логике, семиотике и информатике, представлены на рис. 4.2.

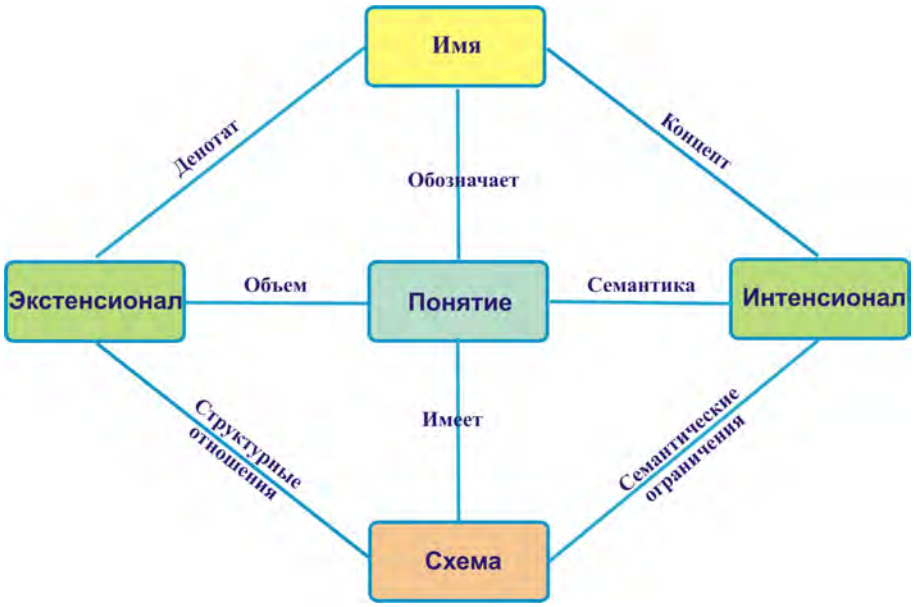


Рис. 4.2 Взаимосвязь категорий для описания понятия в логике, семиотике и информатике

Суммируя вышеизложенное, можно представить понятие в виде тройки

$$P = <intP, shmP, extP>.$$

В зависимости от характера описания системы внимание разработчиков концентрируется на различных составляющих понятия: системные аналитики и специалисты в прикладной области, создающие концептуальную модель системы, больше должны опираться на интенсионал понятия, а проектировщики базы данных — на ее схему. Тем самым понятие становится чрезвычайно удобным средством, которое позволяет, с одной стороны, путем использования интенсионала выразить семантические отношения для

некоторого фрагмента системы, а с другой стороны, с помощью схемы обеспечить возможность перехода к более детальному описанию и представлению этой информации в модели ее реализации.

4.7 Семантика понятия

Семантика понятия, представленная в интенсионале, может быть описана с использованием как статических, так и динамических аспектов понятия. На *статическом* уровне под интенсионалом понятия P понимают совокупность признаков, необходимых и достаточных для принятия решения о принадлежности некоторой сущности экстенсионалу данного понятия, то есть решения, что данная сущность может выступать в качестве значения (оценки) данного понятия. Так как широкий класс понятий может быть описан логическими формулами в виде хорновских дизъюнктов, то под интенсионалом понятия в этом случае обычно понимают

$$intP(X) = \&_{i=1}^{n} P_i(d_q^i, X).$$

Здесь $P_i(d_q^i, X)$ – это предикаты, которым можно дать следующую интерпретацию: дифференциальные признаки D_i сущности $X \in extP(X)$ имеют значения d_q^i, указанные в предикате P_i. Это структурный (статический) подход к определению семантики понятия. Он заключается в том, что интенсионал понятия P принимает истинное значение на всех сущностях $e_q \in extP$, которые обладают соответствующими значениями дифференциальных признаков.

Динамический подход к определению семантики понятий состоит в использовании поведенческих законов, которым подчиняются понятия. Общей формой, используемой для выражения этих законов, является их представление в виде равенств $L(x_1, x_2, \ldots x_n)=R(x_1, x_2, \ldots x_n)$, где L *и* R термы одинакового типа. Здесь L – это термы левой части равенства E, а R – правой.

Равенство E определяет основную информацию, содержащуюся в понятии P, – его интенсионал

$$intP \overset{def}{=} L(x_1, x_2, \ldots x_n)=R(x_1, x_2, \ldots x_n).$$

Термы равенства $L(x_1, x_2, \ldots x_n)=R(x_1, x_2, \ldots x_n)$ представляют собой выражения, построенные из обозначений функций (операций), допускаемых над объектами понятия, констант и переменных. Такое равенство означает, что каковы бы ни были значения (денотаты), задаваемые для переменных данного равенства моделью реализации системы, оценка (интерпретация или вычисление) выражения левой части интенсионала даст тот же результат, что и вычисление выражения правой части.

Для более широкого класса понятий вместо равенств мы имеем систему уравнений

$$L_1(x_1, x_2, \ldots x_n)=R_1(x_1, x_2, \ldots x_n)$$

$$\ldots \qquad\qquad \ldots$$

$$L_k(x_1, x_2, \ldots x_n)=R_k(x_1, x_2, \ldots x_n),$$

которая и определяет интенсионал данного понятия $intP \overset{def}{=} \{E_i\}$.

Имена функций $f_1, f_2, \ldots f_m$, используемые в этих равенствах образуют сигнатуру $\Omega \overset{def}{=} \{f_1, f_2, \ldots f_m\}$ понятия, которая вместе с равенствами $\{E_i\}$ образует аксиоматическое описание $<\Omega, \{E_i\}>$ понятия P.

Пара $<\Omega, intP>$ - определяет абстрактное (формальное) описание понятия P. Кроме формального описания для понятия Р необходимо также определить правила интерпретации, придания смысла символам сигнатуры Ω и равенствам $\{E_i\}$.

Эти правила могут быть разными и приводить к разным результатам. Но в итоге правило интерпретации связывает с аксиоматическим определением $<\Omega, \{E_i\}>$ понятия P некоторый класс моделей в виде алгебраических систем, который содержит не одну, а много алгебраических

систем, и аксиомы $\{E_i\}$ выражают свойства общие для всех этих алгебраических систем.

Ограничившись алгебраическими моделями, мы допускаем определенный произвол, как в выборе конкретной модели, так и в форме ее представления. В действительности, различие между используемыми моделями понятий — это, прежде всего, различие в форме представления реализации понятий системы Σ. Одна и та же модель может быть описана различными средствами, и разные описания одной и той же модели оказываются полезными при изучении свойств понятия.

Если сигнатуру Ω рассматривать как переменную, значениями которой являются алгебраические системы, а совокупность алгебраических систем, удовлетворяющих равенствам $\{E_i\}$, - как «решение системы уравнений» $\{E_i\}$ с одной переменной Ω, то описанная семантика близка к семантике обычных систем уравнений, понимаемой как совокупность всех возможных решений [4]. Только «решениями» здесь оказываются гораздо более абстрактные и сложные математические объекты в виде алгебраических систем, которые удовлетворяют определенным ограничениям, задаваемым системой равенств $\{E_i\}$.

В результате, аксиоматическое задание интенсионала понятия P с помощью множества равенств $\{E_i\}$ можно рассматривать как неявное (косвенное) представление знаний о понятии (внешнюю семантику), а построение алгебраической системы, удовлетворяющей системе равенств $\{E_i\}$, - как явное (конструктивное) представление понятия P (внутреннюю семантику).

5 Объектно-ориентированная методология системантики

5.1 Объекты и их взаимосвязи

Основным исходным понятием ООМ является *объект*. В самом общем смысле объект является описанием множества однотипных сущностей, обладающих *идентичностью* (имеет имя), *состоянием* (обычно для объекта можно указать определенные свойства или связать с ним некоторую схему), *поведением* (с ним можно выполнять определенные операции, или он сам может воздействовать на другие объекты) и *семантикой* (для объекта могут быть заданы некоторые правила, которым должны удовлетворять состояния и операции).

Каждый объект может рассматриваться как экземпляр некоторого класса. Объекты, которые не имеют полностью одинаковых свойств или не обладают одинаковым поведением, по определению, не могут быть отнесены к одному классу.

Объектно-ориентированный подход к описанию сложных систем состоит в том, что структурные элементы систем и их поведенческие свойства описываются в рамках единого понятия. Объекты не являются совокупностью пассивных данных. Это активные образования, которые могут, как производить необходимые операции над элементами своей структуры, так и вызывать требуемые изменении в состояниях других объектов. Объекты могут использоваться для представления как структурированных состояний, так и действий или событий. Следовательно, объекты могу изменяться в соответствии с действительностью, и одновременно могут сами влиять и воздействовать на другие объекты.

Таким образом, под объектом в ООМ фактически понимается понятие, дополняющее его логическую и теоретико-множественную интерпретации путем более последовательного учета поведенческих аспектов. Если при

теоретико-множественном и логическом подходе понятия рассматривалась как некоторая совокупность сущностей и множества отношений между ними, то объектно-ориентированный подход обобщает данную точку зрения, предполагая, что объекты включают совокупность свойств, множество операций, представленных в виде методов, и могут взаимодействовать друг с другом путем посылки сообщений при возникновении событий.

Объектный подход приводит к естественной структуризации сложных систем, обеспечивая между различными составляющими построенной модели простые и ясные интерфейсы в виде посылаемых сообщений. Он способствует практическому образу мышления при представлении знаний о системе с большим числом различных типов объектов, над которыми необходимо совершать разнообразные преобразования. Так как способ оперирования знаниями зависит от их представления и строения, то более целесообразно объединить операции со свойствами объектов, чем рассматривать их отдельно. При этом связи объектов осуществляются только через их взаимодействия, путем посылки сообщений.

Объектно-ориентированная методология дает естественное описание сложной системы в привычных для исследователя и проектировщика понятиях, которые требуются от средств представления знаний, обеспечивает более полное понимание сложных систем и их более адекватное информационное описание. Представляя более разумный компромисс между точностью, полнотой и эффективностью представления знаний о системе, объектно-ориентированные модели становятся одним из основных инструментальных средств при формализации информационного описания сложных систем, особенно если их методы совместно используются с логическим формализмом представления знаний.

Распространение методологии ООМ связано с процессом разработки программ. В настоящее время процедурно-ориентированная декомпозиция программ уступила место объектно-ориентированной, при которой в качестве отдельных структурных единиц программы рассматриваются не

процедуры и функции, а *классы* и *объекты* с соответствующими свойствами и методами. Как следствие, программы перестали быть последовательностью предопределенных на этапе кодирования действий, а преобразовались в событийно управляемые автоматы. Последнее обстоятельство доминирует и при описании поведения сложных систем. В этом случае каждый компонент описания системы представляет собой некоторый объект, ожидающий поступления некоторых событий. Инициаторами событий могут быть другие объекты или внешние системы, а при наступлении события соответствующий объект выходит из состояния ожидания и реагирует на него вполне адекватным образом.

Объектно-ориентированный подход к описанию сложных систем продемонстрировал свою полезность и эффективность при моделировании систем любого размера и сложности в самых различных областях. В частности, большинство современных языков программирования, инструментальных средств и операционных систем ЭВМ являются объектно-ориентированными, что обеспечивает серьезную технологическую поддержку сложному процессу описания любых систем.

Объектно-ориентированное описание сложной системы позволяет адекватным образом представить систему в ее текущем или целевом (желательном) состоянии, а также специфицировать структуру и поведение системы. Современные методы описания сложных систем практически полностью ориентированы на использование инструментальных средств ООМ в виде пакетов прикладных программ, функционирующих на ЭВМ.

5.2 Состояния и поведение объектов

Абстракции объектов в ООМ могут формулироваться в двух формах: *абстракции состояний* системы и *абстракции преобразований* системы. Первые – это абстрактные объекты, тогда как вторые – абстрактные операции. Абстрактные объекты не находятся в статическом состоянии, с

течением времени они изменяют свое состояние. Состояния объекта связывают с некоторым его экземпляром. Экземпляры одного объекта различаются значениями свойств, имена которых одинаковы. Таким образом, множество экземпляров составляет экстенсионал объекта.

В объектно-ориентированном подходе принципиально не делается различий между свойствами и методами (операциями) объекта: и свойства, и методы считаются неотъемлемыми атрибутами объекта. Свойства и методы объектов одного типа одинаковы. Состояние, в котором может находиться объект, изменяется с помощью методов.

Метод — это определенная внутри объекта процедура или функция, для которой значения свойств объекта доступны без явной передачи их в качестве параметров.

Спецификация методов объекта f_1, f_2, . . f_k может быть осуществлена одним из двух способов: функциональным или логическим. При функциональной спецификации метода он рассматривается как функциональное отображение

$$f_i: A_{i1} \times A_{i2} \times \ldots \times A_{iq_i} \rightarrow A_{t(i)},$$

наборов $\{A_1, A_2, \ldots A_n\}$ несущих множеств типов $t(i,1)$, $t(i,2)$, ... , $t(i,q_i)$, где q_i — арность метода f_i, $A_{i1} \times A_{i2} \times \ldots \times A_{i,q_i}$ — область определения, заданная как декартово произведение объектов определенного типа, $A_{t(i)}$ — область значений метода f_i.

Спецификацию функции можно рассматривать в виде пары

<имя функции > : <тип функции>,

где <тип функции> — это список множеств, образующих область определения, и область значения функции, разделенных знаком →.

Тело функции может быть задано интенсионально либо экстенсионально. При интенсиональном определении функции f_i обычно задают правила, формулу или аналитическое выражение, позволяющие по аргументам функции, принадлежащим области ее определения, вычислить

значение функции. При экстенсиональном определении функция задается в виде отображения, отражающего соотношение между аргументами и результатом функции.

При логической спецификации метода задается логическая формула, которая определяет вид запроса, который может быть обработан данным методом. В простейшем случае это может быть предикат, который определяет схему некоторого отношения между объектами. Если термы предиката соответствуют значениям атрибутов данного отношения, то предикат принимает значение истина. Отношение может быть задано экстенсионально в виде таблицы или интенсионально в виде логической программы, которая с помощью механизма логического вывода из известных отношений осуществит синтез требуемого отношения.

Более сложная структура метода может быть определена, если воспользоваться формализмом, используемым в языке логического программирования ПРОЛОГ. В этом случае метод может быть определен как один из запросов к логической программе, а сама программа — как тело для некоторой группы методов, определяющих интерфейс данного объекта или класса объектов.

5.3 Интерфейсы и инкапсуляция объектов

Описание объекта дается в виде спецификации, которая определяет все экземпляры объекта данного типа. Можно считать, что спецификация свойств объекта и множества его методов обеспечивает внешнее описание объекта — его интенсионал, а состояния объекта, напротив, определяют его экстенсионал. Тем самым обеспечивается отделение внешнего интерфейса объекта, известного другим объектам, от внутренней формы его представления в памяти ЭВМ. Если внутреннее представление метода изменяется, то это никак не сказывается на внешних связях и взаимодействиях данного объекта с другими объектами.

Данное свойство в абстрактных типах данных известно как *инкапсуляция,* под которой понимают защиту или сокрытие внутренней структуры представления объекта от внешних воздействий, когда доступ к элементам объекта возможен только к свойствам и методам, указанным в его внешней спецификации. Определение объекта как бы заключается в оболочку, или капсулу [3]. Изменение способа внутреннего представления операций и свойств объекта никак не сказывается на их абстрактном смысле, выраженном во внешней спецификации. Инкапсуляция гарантирует, что объекты действительно используются абстрактным образом, т. е. только через абстрактные методы и свойства, что исключает возможность ошибочного или незаконного использования внутреннего представления объекта.

При объектно-ориентированном подходе информация о системе представляется в виде независимых друг от друга объектов, способ реализации которых может выбираться независимо отдельными группами исследователей. При этом для понимания семантики процессов, происходящих в системе, достаточно понимания только способа использования внешней спецификации методов и свойств объектов и нет необходимости полностью понимать механизм их реализации. Таким образом, объектно-ориентированный подход обеспечивает *локализацию* определенного количества информации о системе в абстрактной форме, выраженной во внешней спецификации объекта или его интенсионале.

С другой стороны, использование внешнего представления объекта позволяет отделить спецификацию от тела объекта и тем самым упростить модификацию его внутреннего представления. Если реализация тела объекта изменяется, но его спецификация остается прежней, то такая модификация не повлияет на взаимодействия с другими объектами.

Объектно-ориентированная модель системы, включающая внешнее поведение объектов в виде явно определенных интерфейсов, позволяет обеспечить:

- уменьшение сложности описания системы;
- большую естественность, прозрачность и более адекватное моделирование реального мира и, как следствие, лучшую понимаемость системы;
- возможность модификации внутреннего представления объекта без разрушении его внешней спецификации, задаваемой интерфейсом.

5.4 Сообщения и события

При объектно-ориентированном подходе система представляется в виде некоторого множества объектов, которые взаимодействуют между собой только путем посылки *сообщений*, вызывающих в других объектах определенные *события*, связанные с изменением состояния объектов [20].

Взаимодействие объектов осуществляется через посылку сообщения, которое включает:

- имя объекта — получателя сообщения;
- сообщение, представляющее собой требование на выполнение определенного метода над объектом и задающее значения необходимых аргументов.

Сообщения позволяют активизировать некоторое действие над объектом, задаваемое с помощью метода. Объект, принимающий сообщение, должен содержать механизм, позволяющий опознать сообщение, выбрать соответствующий метод, активизировать его и передать требуемые для выполнения метода аргументы. Обычно метод вычисляется как некоторая функция, значение которой возвращается в качестве ответа на сообщение. Вычисление метода может изменить состояние объекта или быть причиной посылки новых сообщений другим объектам в зависимости от состояния данного объекта.

Таким образом, объекты могут быть представлены в виде пятерки, содержащей структурную и процедурную части

obj = < int (obj), ext(obj), shm(obj), met(obj), mes(obj)>,

где *met (obj)*—методы, используемые для изменения экземпляров объекта, а *mes(obj)* —сообщения, которые могут посылаться объектом.

Если с каждым фактом посылки сообщения связать некоторое событие, то функционирование объектно-ориентированной модели системы заключается в последовательности возникновения событий, каждое из которых служит запуском некоторого сообщения, реакцией на него некоторого объекта, посылкой сообщения другому объекту. Объекты принимают сообщения, интерпретируют их, активизируют соответствующие методы, изменяют свое состояние, посылают новые сообщения и т. д.

Событие является тем основным понятием, которое используется для моделирования динамических процессов, происходящих в системе. Для целей описания систем достаточно считать, что любое изменение ее состояния (событие) происходит мгновенно и не имеет протяженности во времени. Если изменение является протяженным, то его можно рассматривать в виде пары событий, одно из которых соответствует началу изменения состояния, а второе — его завершению [58].

Мгновенный характер свершения события позволяет полностью абстрагироваться от того, каким образом осуществлялись происшедшие в системе изменения, и не принимать во внимание все ее промежуточные состояния, так как для целей концептуального моделирования системы важны лишь ее начальное и конечное состояния в некоторый момент времени.

Событие может состоять из более простых событий. Событие, которое заключается в изменении состояния одной сущности или ситуации, назовем элементарным. Сложные события могут быть агрегатами или обобщениями более простых событий, которые происходят в один и тот же момент времени. С элементарным событием необходимо связывать начальное

состояние сущности, ее конечное состояние и время наступления события. Кроме того, событию как некоторому понятию можно приписать любые другие свойства, которые представляют интерес для отражения динамики системы.

Часто возникновение и завершение события в системе связывают с определенными условиями. Предикат, который характеризует возможность наступления события, называется *предусловием*, а предикат, описывающий результат завершения события, — *постусловием* [18]. С помощью этих предикатов обеспечивается более сжатое представление информации, относящейся к начальному и конечному состояниям системы, так как предикаты в принципе предоставляют возможность указать целую совокупность состояний, приводящих к наступлению события или возникающих в результате его свершения.

Фиксируя множество состояний, предикаты «постусловие» к «предусловие» позволяют абстрагироваться от конкретных значений свойств понятий, что обеспечивает потенциальную возможность раздельного представления статики и динамики системы. При таком подходе можно сначала осуществлять моделирование динамических процессов в системе, а затем дать уточненное описание статики системы. Причем граф событий определяет лишь внешнее поведение системы, а более детальная проработка структур объектов, введение дополнительных понятий и определение структур ситуаций может быть отнесено на более поздний этап концептуального моделирования.

С каждым событием, происходящим в системе или среде, связывается некоторое имя. Аналогично понятиям одновременные события могут быть объединены в классы событий, между которыми могут быть установлены отношения обобщения и агрегации, рассматриваемые ниже. Например, некоторое событие может быть связано с другим событием отношением *включает* (агрегация) или *обобщает* (обобщение). Это значит, что к

событиям могут быть применены такие же методы, анализа и представления, как и к понятиям.

Основной сферой использования событий являются ситуации, приводящие к изменению состояний сложной системы. Так как изменение состояний понятий означает изменение значений их свойств, связанное с выполнение определенных операций, то под интенсионалом элементарного события следует понимать правила (алгоритм), задающие соответствующее преобразование свойств.

В частности, в качестве такого правила может выступать формула, определяющая функциональное отображение значений свойств. Если соответствующую функцию обозначим через φ_i, а декартово произведение сортов значений свойств $A_{s_1} \times A_{s_2} \times \ldots \times A_{s_n}$ — через A, то функция φ_i производит отображение области A в область изменяемого признака

$$\varphi_i: A \rightarrow A_{s_j}.$$

Тогда правила, задающие функцию φ_i будут выступать в качестве интенсионала события *int(evt*). Рассмотрение события *evt* в виде четверки

$$evt = < int\ (evt),\ ext(evt),\ shm(evt),\ t(evt) >,$$

где *t(evt)* — время наступления события *evt* позволяет определить семантику динамических процессов системы.

Объекты, методы и сообщения обеспечивают более универсальный принцип декомпозиции процессов системы, так как ход событий не управляется извне, а осуществляется в зависимости от состояния объектов.

Сообщение может быть специфицировано в виде

<сообщение> :: = <имя объекта> <имя метода> < аргументы>.

Аргументы — это вспомогательные объекты или свойства, необходимые для выполнения метода. В отличие от событий сообщения не требуют для своего описания явного задания момента времени.

У объектов разных типов могут быть одноименные, но реализованные по-разному методы. Если у объекта нет метода, соответствующего данному сообщению, то такое сообщение теряется. После обработки метода объект перестает быть активным, его состояние сохраняется, и он может принимать новые сообщения. В этом смысле метод объекта отличается от обычной, функции, значение которой теряется после того, как оно использовано.

Сообщение — это понятие, интенсионал которого задается совокупностью правил, поддерживаемых вызываемым методом, а экстенсионал сообщения определяется отображением, реализуемым этим методом.

Хотя объекты рассматриваются как независимые сущности, связи между ними могут быть установлены путем посылки сообщений, изменяющих значения валентных свойств объектов. В том случае, когда из соображений эффективности обработки необходимо определить такие связи явно, можно ввести специальный объект, который будет получать сообщения от связываемых объектов всякий раз, когда изменяются значения их валентных, свойств.

Таким образом, в объектно-ориентированном моделировании описание поведения системы задается не в виде процессов, развивающихся во времени, а как вызовы событий (посылки сообщений), управляемых состоянием самих объектов.

5.5 Процессы, цели и состояния задач

На начальном этапе развития системного анализа доминировал функциональный подход при описании систем, что, однако, порождает множество трудностей при его использовании для описания сложных систем. Методологии системного анализа, ориентированные на описание функциональной иерархии, вводят и поддерживают "конвейерный" принцип организации работы. При этом деятельность системы раскладывается на

простые независимые *задачи*, а затем для достижения результата выделенные задачи соединяются при помощи некоторых процедур, часто сложных и запутанных. Каждая задача хорошо оптимизирована, однако итоговый процесс обычно не является оптимизированным. Такая ориентированная на задачи методология в настоящее время явно устарела.

Функция как базовое понятие, может результативно использоваться для описания поведения объектов и достаточно простых систем, однако при анализе и спецификации более сложных систем необходимы понятия, которые охватывают некоторую совокупность последовательно выполняемых *задач* или действий, необходимых для достижения определенной *цели* или получения требуемого результата. Эту роль при описании сложных систем может выполнить понятие *функционального процесса*.

Дело в том, что реальная деятельность системы не всегда определяется организационной иерархией ее функциональных подсистем. Часто она пронизывает систему в виде набора функциональных процессов, направленных на определенный результат.

Функциональный процесс - это логически завершенный набор этапов работы системы, поддерживающий ее деятельность, направленную на достижение поставленных *целей*. Как правило, набор функциональных процессов для каждой системы своеобразен. Такие процессы, направленные на достижение определенной цели, так или иначе, существуют в каждой сложной системе.

Так, например, использование процессно-ориентированного подхода к описанию систем организационного управления обеспечивает простоту проведения оптимизации как самих функциональных процессов, с точки зрения их организации, синхронизации, взаимной согласованности, так и ресурсов, потребляемых процессами. Одним из важнейших факторов успеха любой организации является наличие в ней единой команды, владеющей ситуацией и согласованно работающей на достижение общей цели. Подход,

использующий понятие функционального процесса, позволяет сформировать такую команду, ориентированную на единые цели организации.

Цели системы должны, безусловно, соответствовать имеющимся в ее распоряжении ресурсам. Но успех достижения поставленной цели будет обеспечен в первую очередь оптимальным взаимодействием всех элементов, образующих эту систему, когда деятельность каждого элемента будет подчинена интересам общей цели. Поэтому одной из центральных проблем описания сложных систем является определение этих целей их упорядочивание и спецификация тех процессов, которые способствуют их достижению, то есть в построении модели сценариев функциональных процессов данной системы.

Модель сценария функционального процесса — это особая форма конечного автомата, которая отображает процесс выполнения работ или действий в системе (рис 5.1).

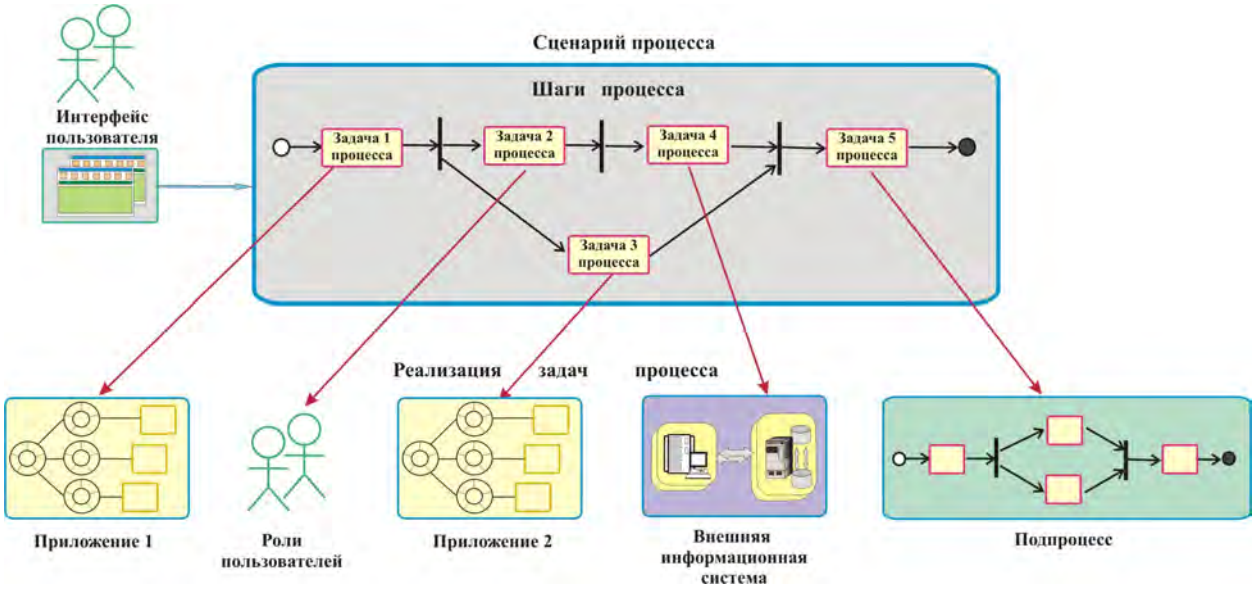


Рис 5.1. Пример сценария функционального процесса

В сценарии процесса моделируются не состояния объектов системы, а состояния производимых действий или решаемых задач. Состояния задач мгновенны и не допускают переходов во время своего выполнения. При возникновении события некоторая задача переходит в активное состояние, в

котором она находится до завершения процесса действия. По завершении данной задачи система переходит к следующей задаче. Завершение сценария процесса осуществляется в тот момент, когда закончилась вся предыдущая деятельность системы.

Задача определяет некоторую работу, которую необходимо выполнить, и она может быть специфицирована различными способами, включая текстовое описание в виде резолюции на документе. Задачи могут быть выполнены вручную или с помощью программных систем. Для определения *сценария* процесса необходимо описать аспекты составляющих их задач (и обрабатывающих устройств, которые их выполняют), связанные с управлением и координированием выполнения задач. Необходимо специфицировать также требования к выполнению задач и связи между ними. Это можно сделать, применяя разнообразные механизмы и методы.

При спецификации сценария процесса ключевыми являются вопросы [45]:

- *Определение задачи*. Структура выполнения каждой задачи определяется указанием множества состояний, которые могут быть наблюдаемы извне, и множества переходов между этими состояниями. Кроме того, могут быть определены те характеристики обрабатывающих объектов, которые могут влиять на требования к выполнению задач.
- *Требования координации задач*. Требования координации задач обычно выражаются через межзадачные зависимости выполнения и зависимости потоков данных.
- *Требования к выполнению (требования корректности).* Требования к выполнению определяются для ограничения выполнения задач таким образом, чтобы удовлетворялись критерии корректности, зависящие от приложения. Они включают требования атомарности выполнения, а также требования к управлению синхронным выполнением задач.

Задача в сценарии процесса - это единица работы, которая может быть выполнена обрабатывающим объектом системы. Задача может быть определена независимо от обрабатывающего объекта, который может ее выполнить, или на основании возможностей и поведения этого объекта. Для спецификации сценария нет необходимости моделировать все аспекты задач. С точки зрения процесса все подробности действий выполняемых задачей и описывающих их последовательность, не являются необходимыми. Каждая задача выполняет некоторые операции над определенным объектом системы. На уровне сценария процесса нет необходимости моделировать внутренние операции задачи – достаточно иметь дело только с теми аспектами задачи, которые видимы извне.

Следовательно, структура задачи может быть определена спецификацией, включающей:

- множества видимых (извне) состояний выполнения задачи;
- множества (корректных) переходов между этими состояниями;
- множества условий, которые делают допустимыми эти переходы (условия переходов могут быть использованы для указания межзадачных требований выполнения).

Абстрактной моделью задачи является конечный автомат смены фазовых состояний задачи, поведение которого может быть определено заданием диаграммы переходов состояний. В общем случае каждая задача (и соответствующий ей автомат) может иметь различную внутреннюю структуру для различных диаграмм смены состояний (рис. 5.2). Это в большой степени зависит от характеристик системы, в которой выполняется эта задача.

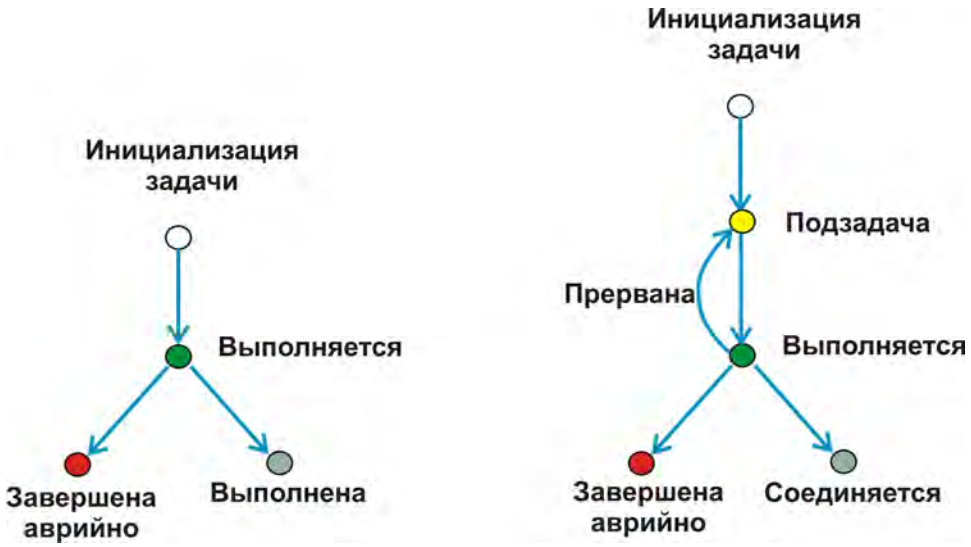


Рис. 5.2. Примеры диаграмм смены фазовых состояний задач.

Переходы между различными состояниями задачи могут вызываться различными событиями в системе или среде. Некоторые из этих переходов контролируются межзадачными зависимостями. Например, задача может быть запущена на выполнение, что вызывает переход из состояния *Начало* в состояние *Выполнение*. Другие переходы контролируются объектом системы, ответственным за выполнение задачи. Например, выполняющаяся задача может быть прервана системой, что вызывает переход из состояния *Выполнение* в состояние *Прервана*. Одно или более состояний задачи могут быть определены как состояние *Завершения*. Когда задача достигает такого состояния, никакие переходы больше не разрешены.

Задачи сценария процесса могут взаимодействовать посредством переменных, локальных относительно сценария. Эти переменные могут содержать параметры задач. Различные начальные параметры могут приводить к различному выполнению задач. Поток данных в промежутке между отдельными действиями, реализуемыми задачей, определяется присваиванием значений их входных и выходных переменных.

В любой момент времени состояние сценария процесса может быть определено как набор состояний входящих в сценарий задач и значений всех

переменных. Выполнение сценария процесса начинается в начальном состоянии, которое задает параметры инициализации. Различные начальные состояния сценария процесса могут приводить к различным его выполнениям.

Как только будет определена принадлежность задачи сценарию процесса, внутренняя структура процесса может быть определена путем спецификации требований координации задач. Обычно эти требования формулируются путем определения предусловий для каждой из выполняемых задач. В общем случае эти зависимости могут быть определены статически перед началом выполнения сценария, либо динамически, в процессе его выполнения.

Динамически определяемые зависимости создаются в процессе выполнения потока работ, часто путем выполнения набора правил. События и условия, влияющие на обработку правил, могут меняться с изменением среды выполнения или ранее выполняющихся задач.

Предусловия для выполнения задачи могут быть определены через зависимости, включающие:

- *Состояния выполнения* других задач. Например, задача Z_i не может стартовать, пока задача Z_j не завершится.
- *Выходные значения* других задач. Например, задача Z_i может стартовать, если задача Z_j была прервана.
- *Внешние переменные*, которые изменяются внешними событиями, не являющимися частью сценария (но могут быть связаны с событиями в других задачах процесса или в других процессах). Примерами таких условий являются: задача Z_i должна стартовать через 2 часа после завершения задачи Z_j.

5.6 Реализация методов и полиморфизм

В зависимости от тех свойств и методов, которые объявлены в спецификациях объектов, они могут объединяться в типы или классы. Свойства и методы объектов одного класса совпадают и вследствие этого объекты данного класса обладают одинаковым поведением. Классы могут иметь один или несколько интерфейсов, которые определяют методы, доступные другим объектам, через данный интерфейс. С помощью интерфейсов осуществляется динамическое взаимодействие данного класса с другими объектами системы. Тело класса содержит модули реализации всех методов, перечисленных в интерфейсе.

У объектов разных классов в общем случае свойства различны, но некоторые из свойств или операций могут совпадать. Множество всех потенциальных объектов, которые могут удовлетворять спецификации (интенсионалу) класса, является его экстенсионалом. Множество имен свойств и методов объектов класса можно определить как его схему. Таким образом, можно считать, что *класс объектов* — это некоторое понятие, которое имеет интенсионал, экстенсионал, схему, методы и может принимать и передавать сообщения.

Спецификации классов также могут рассматриваться как некоторые абстрактные объекты некоторого класса, который называется *метаклассом.*

Кроме того, множество объектов, соответствующих спецификации данного класса, называют экземплярами этого класса. Таким образом, сложная система рассматривается как совокупность классов, их экземпляров и объектов.

На этапе концептуального описания системы объектно-ориентированный подход позволяет отложить вопрос о способе реализации методов класса. Этот вопрос может быть решен позднее на этапе логического описания системы, когда будут ясны все типы данных, используемых методом. Это свойство объектно-ориентированного подхода

позволяет осуществлять динамическую компоновку вычислений путем связывания с методом на этапе исполнения тех модулей, которые нужны для обработки структур данных определенного типа. Так, методы подкласса могут при одних и тех же именах иметь другие алгоритмы обработки, чем операции суперкласса. Это свойство ООМ называется полиморфизмом методов.

Полиморфизм методов — это возможность трактовать один и тот же метод различным образом в зависимости от класса объектов, в котором он в данный момент рассматривается. Полиморфизм достаточно часто используется в математике. Например, операция сложения имеет одно и то же обозначение +, но совершенно различный смысл, зависящий от типа используемых данных (действительных чисел, комплексных чисел, векторов, матриц, и т. д.). Другими словами, можно сказать, что семантика операции + зависит от способа реализации ее тела.

5.7 Объектно-ориентированные представления систем в системантике

Реальный мир состоит из систем, которые пребывают в некотором состоянии, определяемом текущими значениями признаков объектов. Все объекты реального мира также обладают идентичностью, - постоянным свойством, с помощью которого мы отличаем один объект от другого. Реальные объекты, обладающие состоянием, поведением и идентичностью, образуют естественные системы, которые являются самыми сложными из всех известных. Несмотря на сложность, естественные системы демонстрируют разнообразное поведение, могут приспосабливаться к внешним и внутренним изменениям, могут эволюционировать во времени и т. д. Очевидно, что описание сложных систем также должно учитывать их структурные (статические) и поведенческие (динамические) характеристики.

Наиболее современным подходом к описанию статических и динамических аспектов сложных систем является объектно-ориентированное моделирование, основанное на *объектно-ориентированной методологии (ООМ)*. Объектно-ориентированная методология моделирования систем исторически возникла в системном программировании и пришла на смену процедурной организации программ, когда стало очевидно, что традиционные методы алгоритмического программирования не способны справиться с растущей сложностью разработки программ. В современном состоянии ООМ включает все необходимые понятия для структурного и поведенческого описания сложных систем и имеет мощные средства инструментальной поддержки, реализованные в таких методологиях, как UML, ARIS и BPMN.

С точки зрения ООМ достаточно полная модель сложной системы представляет собой определенное число взаимосвязанных *представлений* (views), каждое из которых адекватно отражает аспект поведения или структуры системы. Принцип иерархического построения *моделей* сложных систем предписывает рассматривать процесс построения *моделей* на разных уровнях абстрагирования или детализации в рамках фиксированных представлений. При этом наиболее общее исходное или первоначальное представление сложной системы относится к концептуальному уровню. Такое представление, получившее название *концептуального*, строится на начальном этапе описания системы и может не содержать многих деталей и аспектов. Последующие представления конкретизируют концептуальный уровень, дополняя его представлениями физического уровня.

Следовательно, процесс описания систем в ООМ можно рассматривать как последовательный переход от разработки наиболее общих *моделей* и представлений концептуального уровня к более частным и детальным представлениям физического уровня. При этом на каждом этапе ООМ данные *модели* последовательно дополняются все большим количеством

деталей, что позволяет им более адекватно отражать различные аспекты конкретной реализации сложной системы.

В ООМ одним из основных принципов построения моделей сложных систем является принцип *мультимодельности*, который представляет собой утверждение о том, что никакая единственная модель не может с достаточной степенью адекватности описывать различные аспекты сложной системы. Это означает, что достаточно полная модель сложной системы допускает некоторое число взаимосвязанных проекций (представлений), каждое из которых адекватно отражает некоторый аспект поведения или структуры системы.

Общая схема взаимосвязей представлений языка UML представлена на рис. 5.3 [11].

Рис. 5.3. Общая схема взаимосвязей моделей и представлений сложной системы в ООМ

При этом наиболее общими представлениями сложной системы считаются статическое и динамическое представления, которые в свою очередь могут подразделяться на другие более частные представления. Феномен сложной системы как раз и состоит в том, что никакое ее единственное представление не является достаточным для адекватного выражения всех особенностей моделируемой системы.

5.8 Концептуальное представление сложной системы

Средством формализации структурных и поведенческих свойств системы служит концептуальный уровень, который поддерживает целый ряд семантических абстракций и может быть использован для представления широкого спектра различных по своему характеру и назначению систем. Состав инструментальных средств для спецификации концептуального представления сложной системы рассмотрим на примере языка UML.

UML - это унифицированный язык моделирования для описания, визуализации и документирования объектно-ориентированных моделей сложных систем. Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных и физических моделей сложных систем самого различного целевого назначения.

UML создан по запросу Object Management Group (OMG) - консорциума поставщиков объектных технологий, ответственного за внедрение стандартов в этой области. В 1997 г. UML утвержден OMG в качестве стандарта представления объектно-ориентированных моделей. Этот язык вобрал в себя наилучшие качества методов программной инженерии, которые с успехом использовались на протяжении последних лет при моделировании больших и сложных систем.

Концепция UML принципиально отличается от более ранних технологий (в частности, от блок-схем и электронных таблиц). Вместо того чтобы иллюстрировать изолированные части процесса, UML отдает предпочтение моделям верхнего уровня, позволяющим разработчикам скрывать детали и концентрировать внимание на функциональных особенностях, а не на последовательности действий. Данный подход позволяет начать с формирования общего взгляда на систему, детали же раскрываются позже.

Моделирование сложной системы средствами UML сводится к ее описанию в различных проекциях или представлениях. Каждая проекция описывает определенный аспект разрабатываемой системы, а все вместе они определяют систему с должной степенью полноты, правильности и адекватности.

Каждая из моделей, использованных в UML, позволяет рассматривать функциональные процессы под различным углом. К примеру, пользователи системы при помощи моделей UML могут оценить основные положения сценария функционального процесса и разобраться в том, кто за что отвечает. Разработчики же применяют модели классов и объектов для получения точного представления о том, как интегрировать данные компоненты в систему.

Интегрированная модель сложной системы в нотации UML представлена на рис. 5.4

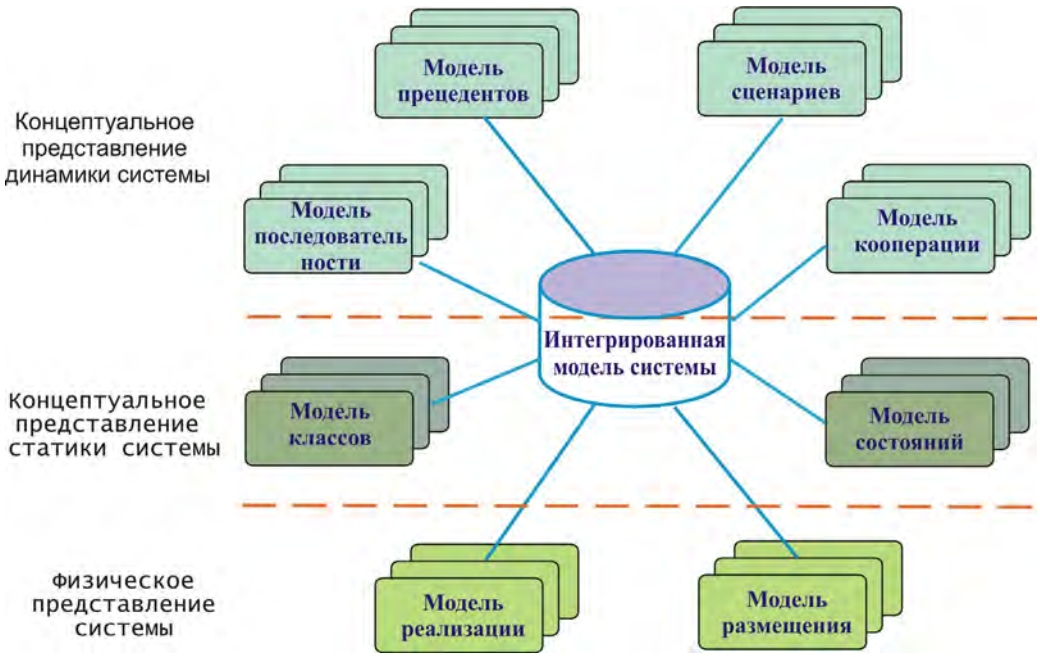


Рис. 5.4 Совокупность моделей, описывающих различные представления сложной системы на языке UML

В результате концептуального моделирования выявляются основные понятия и процессы, необходимые для включения в начальные концептуальные описания системы. Структурные понятия-кандидаты выявляются в результате анализа объектов системы, их свойств, событий и параметров сообщений. Понятия-кандидаты для начальных спецификаций процессов определяются из функциональных описаний системы и используемых объектов и методов.

Концептуальное представление системы определяет состав основных понятий системы, выступающих в роли глобальных поставщиков информации, определяющей целенаправленное поведение системы. Целенаправленное поведение системы определяется соответствующими процедурами принятия решений, которые основываются на полученной извне информации. Обработка этой информации может быть специфицирована в виде совокупности внутренних процессов системы.

С целью проверки информационной полноты концептуальной модели может использоваться технология последовательного согласования полученных концептуальных спецификаций системы. В процессе согласования устраняется синонимия и омонимия имен понятий, уточняются связи между объектами и их состав. Результаты согласования фиксируются в матрицах разных типов, объединенных общим названием «объекты — процессы». Строки матриц соответствуют структурным элементам, а столбцы — процессам. Матрицы являются важным источником информации для построения логического и физического представлений системы.

В результате концептуальное представление системы и протекающих в ней процессов представляет собой формальную высокоуровневую спецификацию ее статических и динамических свойств. Данная спецификация формализует поведение системы на семантическом уровне с

позиций внешнего окружения и служит в качестве исходных данных для физического представления системы.

На этапе создания *концептуального представления поведения* сложной системы используются динамические модели, включающие *модели прецедентов (вариантов использования), сценариев функциональных процессов и модели взаимодействия объектов.*.

Модель прецедентов (вариантов использования, use case diagrams) – это обобщенная модель функционирования системы в окружающей среде. Прецедент – это реакция системы на внешнее воздействие. Модель прецедентов представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных моделей. На моделях прецедентов представляются варианты использования, действующие лица (актеры), а также отношения между ними. Они особенно важны при организации и моделировании поведения системы.

Отправной точкой для описания систем служат прецеденты, которые отражают типичное взаимодействие системы с внешней средой для достижения некоторых целей. На практике, разумеется, нет необходимости определять абсолютно все прецеденты. Как правило, нужно выделить лишь наиболее важные из них. Прецеденты также не следует чересчур детализировать. Обычно вполне достаточно небольшого текстового описания размером в несколько абзацев. Тем не менее, прецеденты не дают картины в целом: для концептуального представления системы необходимы также сценарии процессов, происходящих в системе в ответ на некоторый прецедент.

Модели сценариев функциональных процессов (activity diagrams) – это модели поведения системы в рамках прецедента. На моделях сценариев процессов представляются переходы потока управления от одной

деятельности к другой внутри системы. Модели сценариев процессов относятся к динамическому виду системы; они наиболее важны при моделировании ее функционирования и отражают поток управления между объектами.

Модель взаимодействия объектов (interaction diagrams) – модель процесса обмена сообщениями между объектами системы, представляется в виде диаграмм последовательностей (sequence diagrams) или кооперативных диаграмм (collaboration diagrams). На моделях взаимодействия представляются связи между объектами; и, в частности, показываются сообщения, которыми объекты могут обмениваться. Модели взаимодействия относятся к динамическому виду системы. При этом *диаграммы последовательности* отражают временную упорядоченность сообщений, а *диаграммы кооперации* - структурную организацию обменивающихся сообщениями объектов. Эти диаграммы являются изоморфными, то есть могут быть преобразованы друг в друга.

Модели взаимодействия (последовательности и кооперации) полезнее на более позднем шаге. Кроме того, модели взаимодействия не позволяют выявить параллельные процессы такими способами, как сценарии процессов, в которых можно отразить действия, ограниченные рамками некоторых подсистем, чтобы показать одновременно и участников процессов, и параллелизм. Можно также использовать модели переходов состояний в сочетании со сценариями процессов.

После того как построена концептуальная модель, описывающая динамику системы, разрабатывается логическая модель, которая, с одной стороны, представляет информацию системы, отражающуюся в объектах предметной области, и, с другой стороны, поведение системы, отражающееся в сценариях. Логическая модель специфицирует классы, выполняющие

некоторую реальную работу в системе и образующие повторно используемую архитектурную основу для ее последующих расширений.

На этапе создания статического концептуального описания сложной системы строится *логическое представление,* использующее *модели классов и модели состояний об*ъектов.

Модель классов (class diagrams) – логическая модель базовой структуры системы, отражает статическую структуру системы и связи между ее элементами. Модели классов соответствуют статическому виду системы. На модели классов показывают классы, интерфейсы, объекты и кооперации, а также их отношения. При объектно-ориентированном моделировании систем этот тип моделей используют чаще всего.

Для уточнения моделей классов часто используются *модели объектов* (object diagrams). На моделях объектов представляются объекты и отношения между ними. Они являются статическими "фотографиями" экземпляров сущностей, показанных на моделях классов. Модели объектов, как и модели классов, относятся к статическому виду системы, но с расчетом на настоящую или макетную реализацию.

Модель переходов состояний (statechart diagrams) – модель динамического поведения системы и ее компонентов при переходе из одного состояния в другое. На моделях состояний представляется автомат, включающий в себя состояния, переходы, события и виды действий. Модели состояний относятся к динамическому виду системы; особенно они важны при моделировании поведения интерфейса, класса или кооперации. Они акцентируют внимание на поведении объекта, зависящем от последовательности событий, что очень полезно для моделирования интерактивных систем.

После того как получена исходная информация, необходимая для построения спецификации системы, можно приступить к созданию единой согласованной динамической модели системы, которая будет удовлетворять всем требованиям, зафиксированным в более ранних разрозненных моделях. Эта интегрированная модель может затем использоваться в качестве отправной точки для построения классов и более тщательной спецификации классов в фазе логического проектирования. Построенный таким образом начальный вариант концептуальной динамической модели системы следует рассматривать как некоторую основу для ее уточнения на следующих этапах описания сложной системы.

Таким образом, для построения концептуального представления сложной системы наиболее важными являются следующие два инструмента UML:

- Модели классов, отражающие основные статические понятия исследуемой системы и взаимосвязи между ними, что приводит отдельно взятые понятия в стройную систему.
- Сценарии процессов, которые дополняют модели классов за счет описания потоков действий, т. е. пошаговых процессов выполнения системой некоторых функций. Ключевым аспектом сценариев процессов является то, что они направлены на отражение параллельных действий, важных с точки зрения исключения избыточных последовательностей в функциональных процессах.

Отношения между различными видами моделей UML показаны на рис. 5.5. Приведенная схема является наглядной иллюстрацией итеративного характера представления системы с использованием ООМ на языке UML.

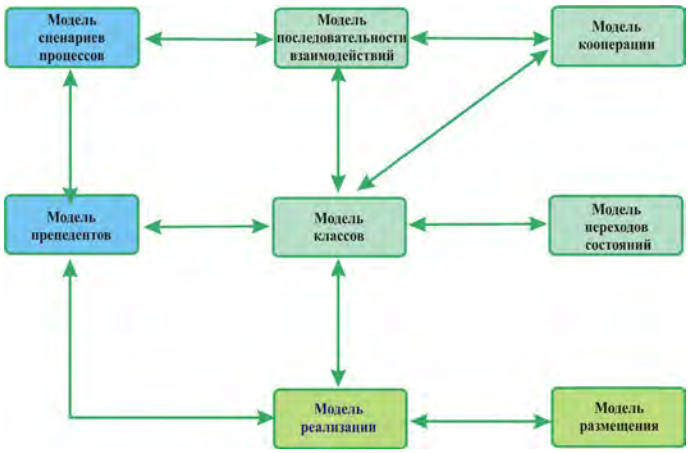


Рис. 5.5. Взаимосвязи между моделями сложной системы

Использование ООМ для описания сложной системы позволяет:

- формализовать функционал системы с помощью моделей прецедентов;
- детализировать функциональные процессы системы путем построения сценариев деятельности;
- построить логическую модель системы в виде моделей классов и их интерфейсов;
- описать процессы взаимодействия объектов при выполнении системных функций;
- описать поведение объектов в виде моделей переходов состояний;
- показать компоненты системы и их взаимодействие через интерфейсы;
- с помощью моделей размещения представить физическую архитектуру реализации системы.

Наиболее существенным обстоятельством в развитии объектно-ориентированной методологии является осознание того факта, что процесс концептуального представления знаний о сложной системе может быть отделен от процесса детального ее описания. Прежде, чем начать рассматривать детальную реализацию классов, их свойств и методов, необходимо определить сами эти классы. Более того, нужно дать ответы на следующие вопросы: сколько и какие классы нужно определить для решения

поставленной задачи, какие свойства и методы необходимы для придания классам требуемого поведения, а также установить взаимосвязи между классами. Абстракции, которые могут использоваться для решения сформулированных проблем, рассматриваются в следующей главе.

5.9 Физическое представление сложной системы

На этапе создания *физического представления* сложной системы используются *модели реализации компонентов и модели размещения*.

Модель реализации (компонентов) (component diagrams) – модель иерархии подсистем, отражает физическое размещение данных, приложений и интерфейсов системы. На модели реализации представляется организация совокупности компонентов системы и существующие между ними зависимости. Модели реализации относятся к статическому виду системы с точки зрения ее физического представления. Они могут быть соотнесены с моделями классов, так как компонент обычно отображается на один или несколько классов, интерфейсов или взаимодействий.

Модели размещения (deployment diagrams) – модель физической архитектуры системы, отображает технологическую конфигурацию системы. На модели размещения обычно представляется конфигурация обрабатывающих узлов системы и размещенных в них компонентов. Модели размещения относятся к динамическому виду архитектуры системы с точки зрения ее развертывания. Они связаны с моделями компонентов, поскольку в узле системы обычно размещаются один или несколько компонентов.

6 Структурное моделирование сложных систем

6.1 Виды абстракций понятий

Одним из центральных принципов описания сложной системы является принцип абстрагирования, который предписывает включать в модель только те аспекты, которые имеют непосредственное отношение к выполнению системой своих функций или своего целевого предназначения. При этом все второстепенные детали опускаются, чтобы чрезмерно не усложнять процесс анализа и исследования полученной модели.

Основной механизм, с помощью которого обеспечивается выделение абстрактных понятий, необходимых для описания систем, - это *абстракция* и *декомпозиция* [33]. В процессе декомпозиции сложная проблема разбивается на ряд более простых, которые могут рассматриваться независимо друг от друга. Декомпозиция, в свою очередь, основана на использовании абстракций.

В процессе абстрагирования предполагается игнорирование ряда подробностей с тем, чтобы упростить проблему путем использования вместо конкретных объектов их абстрактных заместителей – имен объектов. Использование методов абстрагирования и последующей декомпозиции типично для процесса анализа большинства реальных систем. Обычно они используются итерационно до тех пор, пока исходная задача не будет сведена к подзадачам, решение которых известно.

Абстракция — это выделение существенных признаков и связей понятий, используемых при решении некоторой задачи, и игнорирование несущественных. Абстрагирование является основным методологическим приемом системного анализа. Оно позволяет разбить анализируемую систему на подсистемы, каждая из которых, как правило, проще исходной системы. Причем при рассмотрении исходной системы нет необходимости учитывать те детали и ту более подробную информацию, которые используются на этапе рассмотрения подсистем. В поле зрения должны находиться только те

знания, которые позволяют охватить проблему целиком и осуществить ее декомпозицию на более простые подсистемы.

Абстрагирование обеспечивает упорядочение, структуризацию и понимание информации об исследуемой системе. Поэтому методы абстракции широко используются в концептуальном моделировании систем. Для концептуальных моделей систем различны не только множества типов компонент и элементов различных систем, но и связи между типами. Что не позволяет указать универсальный набор типов элементов, отношений и операций, или, более точно, указать сигнатуру отношений и операций, общую для всех систем. Однако некоторые базовые типы, отношения и операции могут использоваться для построения концептуального представления любых систем. В качестве таких базовых конструкций обычно используются следующие механизмы: *типизация, обобщение, агрегация* и *ассоциация* [63].

Перечисленным механизмам абстрагирования соответствуют определенные логические приемы, с помощью которых может быть достигнут необходимый результат. К таким приемам образования понятий обычно относят:

- синтез и анализ понятий;
- объединение понятий на основе их сходства или подобия;
- сравнение и сопоставление конкретных сущностей с целью выявления общих признаков;
- связывание двух или более понятий.

Синтез и анализ понятий используются в абстракции агрегации. В процессе объединения понятий на основе их сходства или выявленного подобия порождается новое понятие, которое является обобщением исходных понятий. При установлении определенного сходства сущностей в процессе сравнения или сопоставления их признаков может порождаться новое более общее понятие, которое объединяет целый класс подобных понятий, что соответствует абстракции типизации. Если в процессе

связывания понятий образуется новое понятие, а исходные понятия выступают в виде членов вновь порожденного понятия, то это абстракция ассоциации.

Вследствие того, что для каждого метода абстрагирования возможно как повышение уровня абстракции, так и его понижение, мы имеем дело с парами: *типизация — конкретизация, обобщение — специализация, агрегация — декомпозиция, ассоциация — индивидуализация.*

Данные виды абстракций фиксируют простейшие базовые отношения между понятиями и поэтому более подробно рассматриваются ниже. Однако необходимо помнить, что анализ сложных систем должен опираться на принцип иерархического построения моделей сложных систем, который предписывает рассматривать процесс построения модели на разных уровнях абстрагирования или детализации в рамках фиксированных проекций или представлений.

При описании некоторой конкретной системы возникает первая основная проблема: формирование лексики абстрактной системы. И, прежде всего, это касается создания четырех словарей: *словаря понятий*, *словаря отношений*, *словаря процессов* и *словаря действий* (операций). Эти словари — основные для представления знаний о всех сложных системах системантики.

Средства структурного описания сложных систем подобны методам, используемым при *спецификации программ*. Спецификация программы – это описание задачи, которую должна решать программа, приводя в действие вычислительную машину. Написать спецификацию - это значит, прежде всего, подобрать точные понятия, адекватные задаче, что тесно связывает спецификацию программ с математикой. Работа с понятиями – главное в построении и использовании спецификаций [4]. Аналогично, понятийные средства должны быть основным инструментарием для описания сложных систем.

Выбирая и оценивая тот или иной способ организации понятийных средств, надо учитывать существенное отличие задачи разработки спецификации для программ от задачи представления знаний для сложных систем. При описании сложной системы от понятийных средств требуются выразительность, богатство и гибкость, но в отличие от задач программирования не обязательна их непосредственная и эффективная машинная реализация.

Описание сложной системы может быть очень большим по объему. Составление такого описания требует много времени и умения отделять существенные моменты от несущественных. Внесение изменений в такое связное описание без использования современных ЭВМ становится практически невозможным из-за его объема и сложности существующих в нем логических связей.

Поэтому современные методы описания сложных систем практически полностью ориентированы на использование инструментальных средств в виде пакетов прикладных программ, функционирующих на ЭВМ. При этом одним из основных допущений является предположение о том, что любая система состоит из совокупности компонент (элементов, частей, подсистем, объектов, предметов или сущностей), которые могут быть уникально идентифицированы и которые можно адекватным образом представить в памяти вычислительной машины, чтобы с ее помощью обеспечить анализ, обработку и выдачу информации в форме, удобной для принятия решений.

6.2 Типизация и конкретизация понятий

Использование понятия *тип* – это одна из основных концепций современного программирования. Естественно, что эта концепция должна быть соответствующим образом представлена и в теории системного анализа.

Понятие-тип обычно выражает то общее, что присуще некоторой совокупности сущностей сложной системы. Причем это общее в первую очередь выражает однородность, однотипность сущностей и игнорирует индивидуальные отличия сущностей друг от друга, определяемые значениями признаков.

Введение понятия тип обусловлено стремлением в процессе системного анализа классифицировать объекты различной природы, объединять в одну группу объекты со сходными признаками и применять к ним одни и те же операции, одни и те же способы обработки. Описание сложных систем будет осуществляться проще и естественнее, если разбивать объекты исследуемой системы на определенные группы (типы) и манипулировать с объектами каждой группы наиболее адекватными средствами.

Из сказанного ясно, что тип объекта определяет некоторое множество (класс) однородных сущностей. Причем, предполагается, что каждый объект принадлежит только одному типу. В зависимости от того, является ли данный объект неделимым или его можно разложить на компоненты, различают *простые* и *структурные* типы. Важно также, что с каждым типом связывается определенный набор операций, по созданию и анализу его значений, которые характеризуют динамические свойства типа. Введение типов позволяет обеспечить непротиворечивое объединение различных представлений системы со стороны специалистов, участвующих в описании сложной системы.

Процесс группировки объектов на основе соответствия их интенсионалов некоторому эталону и образования на этой основе типов, необходимых для описания сложной системы, называется *типизацией*. Например, класс объектов ЛИЧНОСТЬ может быть создан путем объединения таких понятий-сущностей, как Петров, Сидоров, Александров и т. д., при том условии, что совокупности основных признаков этих понятий

совпадают. Обратным по отношению к процессу типизации является процесс *конкретизации* или порождения экземпляров.

В программировании все большее значение приобретает формальный подход к описанию типов, основанный на теории *абстрактных типов данных*. Он заключается в переходе от конкретных множеств, операций и отношений объектов систем к их именам, и описанию соотношений между операциями через имена [20]. При таком подходе тип конструируется путем выполнения некоторого набора операций над исходными множествами, называемыми *основами*. Под операцией понимается функция из декартова произведения основ операндов (область определения) в основу результата (область значения).

Так как тип объекта определяет множество значений посредством множества операций, то при определении типа необходимо на интенсиональном (абстрактном) уровне указывать синтаксис и семантику операций, а на экстенсиональном (конкретном) – способ их реализации. Соответственно, любой конструктив типа состоит из двух разделов: *спецификации* и *реализации*. Спецификация указывает синтаксис (имена объектов и операций, типы операндов операции и ее результат) и семантику (равенства, ограничения и условия применимости) операций. Реализация указывает представление типа, то есть объекты каких типов являются компонентами для значений данного типа и алгоритмы операций в терминах типов представления.

В абстрактных типах данных для каждого имени указывается, в каких операциях это имя может использоваться и к какому типу оно относится. С этой целью в абстрактных типах вводится понятие сигнатуры Ω, которое можно определить в виде пары

$$\Omega = <Sort, Op>,$$

где $Sort=\{T_1, T_2 \cdots T_n\}$ – это множество исходных типов, называемых основами, а $Op = \{F_1, F_2 \cdots F_k\}$ – множество имен операций.

Так, например, понятие абстрактный тип *Nat = {0, 1, 2, ...}* (множество натуральных чисел) может быть описано следующим образом:

Основы: {*bool, nat, setofnat*};

Операции:

Логические константы: {*true, false: bool*;

Отрицание: *not: bool* $\rightarrow$ *bool;*

Логические операции: *or, and: bool* $\times$ *bool* $\rightarrow$ *bool;*

Константа нуль: *zero: nat;*

Следующий: *succ: nat* $\rightarrow$ *nat;*

Суммировать *add: nat* $\times$ *nat* $\rightarrow$ *nat;*

Операции сравнения чисел: $\cong, \leq$: *nat* $\times$ *nat* $\rightarrow$ *nat;*

Пустое множество: *empty: setofnat;*

Вставить: *insert: setofnat* $\times$ *nat* $\rightarrow$ *setofnat;*

Содержит: *has: setofnat* $\times$ *nat* $\rightarrow$ *bool}*

Для придания смысла введенным операциям сигнатуры Ω именам основ сопоставляются конкретные множества A_i, а именам операций конкретные функции f_i на этих множествах. Тем самым задается некоторая алгебра, которая называется *реализацией* данной сигнатуры Ω. Алгебра может быть определена как пара функций, одна из которых отображает имена основ в множества, а другая – знаки операций в функции.

Так, например, при реализации типа *Nat* на ЭВМ с основой *bool* связывается множество из двух машинных значений *0 и 1*. Основе *nat* сопоставляется множество комбинаций битов ячейки памяти. Константа *zero* реализуется ячейкой с нулями по всем ее битам.

Операция *succ* реализуется путем команды сложения исходного числа с единицей, а операции $\cong$ *и* $\leq$ реализуются командами сравнения и вычитания двух чисел, устанавливающими соответствующий признак результата.

Основе *setofnat* сопоставляется множество ячеек памяти, а соответствующие операции *empty, insert и has* реализуются машинными командами над этим множеством ячеек памяти.

Для сигнатуры $\Omega = <Sort, Op>$ существует особая алгебра, называемая алгеброй слов или термов. Она строится следующим образом. В соответствии с количеством имен основ $Sort=\{T_1, T_2 \cdots T_n\}$ сигнатуры Ω, все множество базисных термов *Term* разбивается на подмножество термов каждой из основ. Множество базисных термов основы $T_i \in Sort$ определяется следующим образом:

- все константы основы типа T_i являются базисными термами основы T_i;
- для всех знаков операций $F_i \in Op$ и всех базисных термов $t_1, t_2 \cdots t_n$, принадлежащих основам $T_1, T_2 \cdots T_n$, $F_i(t_1, t_2 \cdots t_n)$ является базисным термом.

Для абстрактного типа *Nat* базисными термами основы *setofnat* будут

empty, insert(empty,zero), succ(zero), *insert(insert(empty,zero), succ(zero*)) и т. д.

В этой же сигнатуре базисными термами основы *nat* будут

zero, succ(zero), *succ(succ (zero*)), *succ(succ(succ (zero*))) и т. д.

По существу алгебра термов – это линейная запись конечных упорядоченных наборов деревьев. Каждое применение операций этой алгебры образует выражение большей длины из выражений меньшей длины.

Каждый терм алгебры слов может быть вычислен (реализован) в некоторой алгебре *A* сигнатуры Ω. Для этого сначала выполняется оценка переменных $ev(x_i)$. А затем для любого данного терма *t* посредством последовательного применения функций оценки осуществляется привязка переменных к конкретным значениям несущих множеств алгебры *A,* и вычисляются значения операций F_i путем оценки термов аргументов операции и сопоставления операции F_i соответствующей функции f_i алгебры *A*.

Например, пусть для абстрактного типа *Nat* константе *zero* соответствует число *0,* операции *succ(n)* соответствует операция *+(n, 1)* которая по числу *n* вычисляет число *n+1,* а имени операции *add(n,m)* соответствует операция сложения чисел *+(n,m)*. Тогда, если для абстрактного типа *Nat* задано множество переменных $X = \{n, m\}$ и $ev(n)=2,$ а $ev(m)=7,$ и необходимо вычислить значение терма *ev(add(succ(n), m))* то по определению

$ev(add(succ(n), m)) = +(ev\ (succ(n)), ev(m)) = +(+(ev(n)+1), 7)= +(2+1, 7) =10.$

Очевидно, что для заданной сигнатуры $\Omega = <Sort, Op>$ может существовать несколько реализаций, несколько подобных алгебр. Для выделения среди этих реализаций алгебры, отражающей семантику абстрактного типа, необходимо задать определяющие соотношения этого типа, которые обычно формулируются в виде равенств с переменными.

Если $E = \{E_1, E_2 \cdots E_k\}$ – множество равенств абстрактного типа, то его можно определить в виде тройки

$AT = <Name, \Omega, E>,$ где

Name – имя типа,

$\Omega = <Sort, Op>$– сигнатура типа,

$E = \{E_1, E_2 \cdots E_k\}$ – определяющие соотношения типа (конечное множество уравнений вида $L_i=R_i,$ где L_i (левая часть равенства) и R_i (правая часть равенства) являются термами одинакового типа).

Алгебра сигнатуры Ω, удовлетворяющая каждому из равенств $E_i,$ является *моделью* абстрактного типа. Совокупность определяющих равенств синтаксическими средствами выражает *абстрактную* семантику типа, а алгебраическая модель – *конкретную*. Принцип абстракции позволяет работать с понятием-типом, зная только имена основ, имена операций и их абстрактную семантику не вдаваясь в детали реализации, что и обеспечивает более простое описание сложной системы.

Для типа данных *Nat* совокупность определяющих соотношений (абстрактная семантика) может быть задана в виде:

$not(true) = false;$

$not(false) = true;$

$\forall b: bool,\quad and(false, b) = false;$

$and(true, b) = b;$

$or(true, b) = true;$

$or(false, b) = b;$

$\forall m, n:nat,\quad zero \leq n = true;$

$succ(m) \leq zero = false;$

$succ(m) \leq succ(n) = m \leq n;$

$add(zero,n) = n;$

$add(n,m) = add(m,n);$

$add(succ(n),m) = succ(add(n,m));$

$m \cong n = and\ (m \leq n,\ n \leq m);$

$\forall x,y:nat,\quad s: setofnat,$

$has(empty, x) = false;$

$has(insert(s, x),y) = or(x \cong y,\ has(s,y)).$

Понятие-тип можно интерпретировать как отношение эквивалентности, которое задается на экстенсионале данного понятия. Так как всякое отношение эквивалентности может быть выражено через отношение *быть эталоном*, то отсюда следует, что сущности принадлежат одному типу в том случае, когда они имеют общий эталон, который может быть отождествлен с их интенсионалом *int(AT)* [54], задаваемым с помощью определяющих равенств $E = \{E_1, E_2 \cdots E_k\}$.

Вследствие ограничений, накладываемых определяющими равенствами, множество моделей, естественно, будет уже множества всех алгебр данной сигнатуры.

Таким образом, абстрактное понятие типа *AT* должно включать следующие компоненты:

- имя типа (имя понятия) *Name* – разным именам типов соответствуют разные понятия;
- набор операций $\{F_1, F_2 \cdots F_k\}$ – сигнатура типа $\Omega = \{ T_1, T_2 \cdots T_n; F_1, F_2 \cdots F_k\}$ определяющая, что с основами данного типа можно делать;
- определяющие соотношения $E = \{E_1, E_2 \cdots E_k\}$ между операциями (интенсионал понятия *intAT*), задаваемые равенствами, выражающими смысл (абстрактную семантику) понятия;
- реализацией (экстенсионал понятия *extP*) – конкретным представлением типа в виде алгебраической системы или алгебры.

Существуют два способа построения абстрактных понятий:

- *конкретизация* – построение от общего к частному, когда первоначально разрабатывается абстрактный тип, а затем создаются различные его реализации;
- *типизация* – когда в конкретных типах выделяются общие имена операций и сортов, и некоторые равенства, верные во всех реализациях.

Например, типы данных, предоставляемые некоторой системой программирования, являются конкретными: для каждого из них транслятор обеспечивает конкретный способ представления значений (в виде последовательностей битов памяти) и конкретные алгоритмы реализации операций, конкретную алгебру. Причем, в разных трансляторах одному и тому же абстрактному типу сопоставляются различные конкретные типы, различные алгебры. В то же время типы, предоставляемые языком программирования, являются абстрактными. Для каждого из них описание языка задает только имена основ, имена и типы операций, синтаксис (правила записи операций) и абстрактную семантику семантику (смысл операций).

Если в процессе абстракции типизации понятий-сущностей *P* и *Q* порождается понятие-тип *R*, то для интенсионалов, схем и экстенсионалов должны выполняться соотношения [36]:

$$int\ R = int\ P = int\ Q,$$

$$shm\ R = shm\ P = shm\ Q,$$

$$ext\ R = ext\ P \cup ext\ Q.$$

Если объекты P и Q принадлежат к различным типам, то их экстенсионалы не пересекаются $extP \cap extQ = \varnothing$. В этом случае вся совокупность объектов системы распадается в каждый момент времени на непересекающиеся множества и, следовательно, каждый тип соответствует классу эквивалентности объектов. Отнесение объекта к определенному типу осуществляется разными способами. Тип объекта может быть определен при первом упоминании этого объекта, или установлен при помощи различных классификационных процедур, если известны свойства объекта. Имена всех типов являются частью тезауруса (словаря) системы, и, подобно тезаурусу, множество типов системы обладает определенной структурой.

Принадлежность сущностей одному типу позволяет переносить знания, полученные в результате описания системы, с одной сущности на другую. Так, можно утверждать, что сущности, принадлежащие экстенсионалу одного типа, имеют одинаковое внутреннее устройство, а значит, и обладают одной и той же схемой. Абстракция типизации дает возможность при описании систем работать, зная только синтаксис и семантику операций соответствующего типа, не вдаваясь в детали реализации. Что, в свою очередь, позволяет осуществлять модуляризацию системы путем использования более крупных компонент – введенных типов объектов.

Семантически типизация позволяет также выразить отношение *есть экземпляр* между понятием-сущностью и понятием-типом и абстрагироваться от различий между описываемыми экземплярами.

Понятие типа широко используется в современных языках программирования, где оно часто трактуется как некоторое множество с

заданной на этом множестве совокупностью операций. Конечно, при таком определении понятие типа оказывается неполным, так как игнорируется информация об интенсионале и схеме типа данных. Для простых типов данных (целый, вещественный, символьный, логический и т. д.) приведенного определения бывает достаточно, однако для сложных типов (массивы, записи, файлы) необходима информация как об интенсионале, так и о схеме описываемых объектов. Как указывается в [60], хороший механизм типов является ключевым фактором обеспечения ясного и надежного описания системы.

Введение в описание системы механизма определения произвольных типов позволяет формулировать информацию о системе с помощью более крупных семантических модулей, не вникая в подробности реализации используемых понятий. При работе с типами, представленными в виде многосортных алгебраических систем, исследователю может быть неизвестна или неважна конкретная реализация типа, но необходимо знание имен его составных частей (основ) и знание некоторых свойств операций и отношений, отражающих семантику вводимого понятия.

С помощью операций из известных базовых типов можно строить новые более сложные производные типы. Для этого обычно используются теоретико-множественные операции $\cup$ (объединения), $\cap$ (пересечения), / (разности) и $\times$ (декартова произведения), а наиболее употребительными абстракциями, порождающими новые производные типы на основе базовых, являются абстракции обобщения и агрегации.

6.3 Обобщение и специализация понятий

Описывая систему, особенно большую и сложную, необходимо представить ее в виде относительно крупных и осмысленных модулей. Каждому такому модулю должна соответствовать емкая смысловая единица в виде понятия. Введение подобной модуляризации системы придает ее

описанию упорядоченную и прозрачную структуру. К выделенному понятию можно применить тот же прием, и тогда мы получим несколько уровней описания с разной степень детализации. Причем, желательно, чтобы данный процесс не требовал явного использования реализации типа в виде его экстенсионала, то есть указания конкретной модели типа в виде алгебры.

Процесс выделения многоуровневой совокупности понятий, который абстрагируется от конкретной реализации типов, – это процесс *обобщения понятий*, в результате которого порождается иерархия типов, описывающих данную систему. Иерархия типов конструируется на основе одного или нескольких базовых понятий-типов, которые в свою очередь могут быть иерархическими. Каждое базовое понятие-тип может быть проанализировано и реализовано отдельно, без использования информации о тех типах, для которых оно может быть использовано. Причем, вновь созданное понятие-тип снова может рассматриваться как базовое, поведение которого определяется операциями, удовлетворяющими абстрактной семантике.

Обобщение понятий — это такая форма порождения нового понятия R па основе одного или нескольких подобных понятий P и Q, когда порождаемое понятие R сохраняет общие признаки и операции исходных понятий P и Q, но игнорирует их более тонкие различительные признаки. Для интенсионалов и экстенсионалов понятий P, Q и R, участвующих в данном процессе абстракции, справедливы следующие соотношения [36]:

$intR = intP \cap intQ$,

$ext\ R = ext\ P \cup ext\ Q$.

При этом предполагается, что $intP \cap intQ \neq \varnothing$.

Эти формулы непосредственно следуют из определения, так как пересечение интенсионалов понятий P и Q обеспечивает выделение их общих признаков и операций. Противоположный процесс, когда исходное понятие может делиться на несколько более узких понятий, называется *специализацией,* или *ограничением понятий.*

Следовательно, при обобщении подобные видовые понятия соотносятся с родовым понятием более высокого уровня, а при специализации, наоборот, родовые понятия делятся на два или более видовых понятий низкого уровня. Это означает, что объем (экстенсионал) понятия в процессе обобщения увеличивается и содержит в качестве своей части объемы (экстенсионалы) исходных понятий, что и находит свое выражение в формуле $ext\ R = ext\ P \cup ext\ Q$.

В связи с этим обобщение понятий можно определить как такую логическую операцию, в процессе выполнения которой происходит переход от понятий меньшего объема к понятию большего объема путем исключения из содержания исходных понятий тех признаков и операций, которые позволяют отличать их друг от друга [16]. Так, исходя из совокупности понятий СТОЛ, КРЕСЛО, КРОВАТЬ и абстрагируясь от их видовых признаков, мы можем образовать родовое понятие МЕБЕЛЬ.

При специализации понятий, напротив, к признакам исходного понятия добавляются дополнительные признаки и операции, которые позволяют выделить из всего объема обобщенного понятия некоторую его часть. Например, путем добавления к содержанию обобщенного понятия ЛИЧНОСТЬ дополнительных признаков МЕСТО РАБОТЫ и МЕСТО УЧЕБЫ можно выделить такие понятия, как СЛУЖАЩИЙ, РАБОЧИЙ, ПРЕПОДАВАТЕЛЬ, СТУДЕНТ и т. д.

В результате абстракцию обобщения часто связывают с отношением *есть-некоторый (is-a)* между видовым и родовым понятиями. В процессе обобщения более общее понятие относится к исходному как род к виду, а в процессе специализации ограниченное понятие соотносится с исходным как вид с родом.

Для схемы понятия-обобщения R и обобщаемых понятий P и Q, которые называют также категориями, выполняется соотношение

$shm\ R = shm\ P \cap shm\ Q$.

В качестве примера абстракции обобщения рассмотрим понятия СЛУЖАЩИЙ, ПРЕПОДАВАТЕЛЬ, СТУДЕНТ, которые можно считать категориями понятия ЛИЧНОСТЬ. Сопоставление интенсионалов данных понятий позволяет выявить их значительное сходство, как в именах используемых признаков, так и в операциях, что служит основанием для применения абстракции обобщения, так как пересечение схем этих понятии не пусто. В результате процесса абстрагирования может быть получено обобщенное понятие ЛИЧНОСТЬ, абстрактная семантика которого определяется на основе пресечения определяющих соотношений для исходных видовых понятий. На практике перед выполнением абстрагирования необходимо провести тщательный семантический анализ имен признаков и операций с целью устранения возможной синонимии.

Абстракция обобщения задает на множестве всех понятий, используемых для описания системы, отношение частичного порядка.

Процесс обобщения допускает возможность многократного его применения, в результате чего возникает иерархия обобщений, которую можно изобразить в виде ориентированного графа (рис. 6.1).

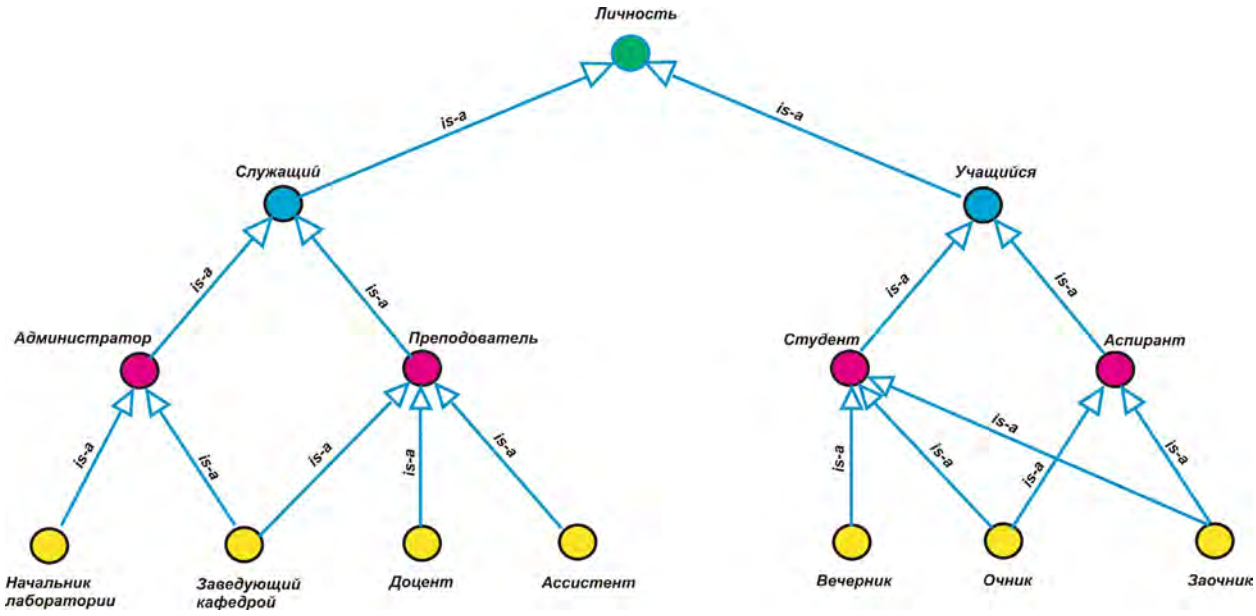


Рис. 6.1. Иерархия обобщений понятия ЛИЧНОСТЬ для системы ВУЗ

Особенностью примера иерархии обобщений, представленного на рис. 6.1, является наличие для понятия ЗАВЕДУЮЩИЙ КАФЕДРОЙ двух понятий более высокого уровня, с которыми данное понятие связано отношением *есть-некоторый*. Если понятия более низкого уровня связаны отношением обобщения только с одним понятием более высокого уровня, тогда возникает древовидная структура обобщений.

Следовательно, понятия, описывающие сложную систему, можно организовать в виде иерархической структуры, которая по внешнему виду напоминает классификационную схему в понятийной логике. Иерархия понятий строится следующим образом. В качестве наиболее общего понятия или категории берется понятие, имеющее наибольший экстенсионал (объем) и, соответственно, наименьший интенсионал (содержание). Это самый высокий уровень абстракции для данной иерархии. Затем данное общее понятие конкретизируется, то есть уменьшается его экстенсионал и увеличивается интенсионал. Появляется менее общее понятие, которое на схеме иерархии будет расположено на уровень ниже исходного. Этот процесс конкретизации понятий может быть продолжен до тех пор, пока на самом нижнем уровне не будет получено понятие, дальнейшая конкретизация которого в данном контексте либо невозможна, либо нецелесообразна.

6.4 Обобщение классов, наследование поведения и семантики

В ООМ абстракция обобщения понятий принимает форму обобщения классов. *Обобщение классов* – это отношение между более общей сущностью (суперклассом, или родителем) и более частным или специальным элементом (субклассом, или потомком). Обобщение классов означает, что объекты субкласса (класса-потомка) могут использоваться всюду, где встречаются объекты суперкласса (класса-родителя), но не наоборот. Другими словами, потомок может быть подставлен вместо родителя. При этом он наследует

поведение родителя, в частности его свойства, операции, а также определяющие соотношения, относящиеся к абстрактной семантике [11].

Применительно к модели классов данное отношение описывает иерархическое строение классов и *наследование* их свойств, поведения и семантики. Используя абстракцию обобщения классов, можно организовать реализацию классов в виде алгебраической структуры определенного вида, если руководствоваться специальными правилами наследования.

Наследование свойств и операций – это принцип, в соответствии с которым знание о наиболее общей категории разрешается применять для более частной категории. Наследование тесно связано с иерархией классов, определяющей, какие классы следует считать наиболее абстрактными и общими по отношению к другим классам. Причем подкласс обладает всеми свойствами и поведением суперкласса, а также имеет собственные свойства и поведение, которые могут отсутствовать у суперкласса.

Механизм наследования обеспечивает возможность новому определяемому классу объектов автоматическое заимствование свойств и операций (методов) класса, определенного как его суперкласс. В этом случае при описании подкласса нет необходимости указывать все его свойства и операции. Часть общих свойств и операций может наследоваться из суперкласса данного класса. Это упрощает описание сложных систем и позволяет локализовать необходимую информацию о свойствах и операциях только в одном месте, что облегчает возможность внесения последующих изменений.

Часто, хотя и не всегда, у потомков есть и свои собственные свойства и операции, помимо тех, что существуют у родителя. При этом реализация операции потомка с той же сигнатурой, что и у родителя, но с другими

определяющими соотношениями, может замещать реализацию операции родителя (*полиморфизм операции)*.

В UML отношение обобщения классов обозначается сплошной линией с треугольной незакрашенной стрелкой на одном из концов, направленной на родителя, как показано на рис. 6.2

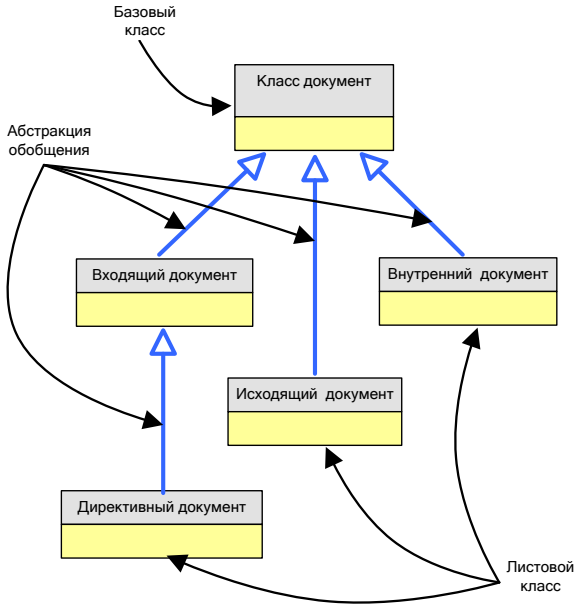


Рис. 6.2 Иерархия абстракции обобщения классов в ООМ

Класс может иметь одного или нескольких родителей или не иметь их вовсе. Класс, у которого нет родителей, но есть потомки, называется *базовым* или *корневым*, а тот, у которого нет потомков, - *листовым*. О классе, у которого есть только один родитель, говорят, что он использует одиночное наследование; если родителей несколько, речь идет о множественном наследовании.

Графическое изображение отношения обобщения понятий по форме соответствует графу специального вида - иерархической структуре. В этом

случае суперкласс является корнем дерева, а подклассы - его листьями. Отличие заключается в возможности указания на диаграмме классов дополнительной семантической информации, которая может отражать различные теоретико-множественные характеристики данного отношения. Так, например, могут быть указаны некоторые ограничения, которым должны удовлетворять все подклассы данного класса. В частности, может быть указано, что подклассы не могут содержать объектов, одновременно являющихся экземплярами двух или более классов. Или, например, что в данном отношении специфицированы все подклассы, и других подклассов у данного класса быть не может.

Для абстракции обобщения можно указать три семантических ограничения, которые должны выполняться для всех понятий [43].

Так как предполагается, что категория — это подмножество класса-обобщения, то каждый экземпляр, содержащийся в этой категории, должен быть представлен в экстенсионале класса-обобщения. Это утверждение следует непосредственно из равенства $ext\ R = ext\ P \cup ext\ Q$. Например, если обобщение – *Класс-документ*, то документ с именем *D1* компании *Microsoft,* указанный в категории *Входящий документ,* должен присутствовать как экземпляр и в обобщении – *Класс-документ.*

Если категория является атрибутом некоторого экземпляра класса-обобщения, то в этой категории должно быть имя этого экземпляра, то есть любой экземпляр, принадлежащий $ext\ R$, принадлежит либо $ext\ P$, либо - $ext\ Q$. Это утверждение также является следствием равенства $ext\ R = ext\ P \cup ext\ Q$. Например, документ с именем *D1* компании *Microsoft,* указанный в обобщении *Класс-документ* должен присутствовать как экземпляр и в категории *Входящий документ.*

Используемые имена атрибутов, их значения и имена категорий должны быть согласованы между классом-обобщением и его категориями.

Это требование следует из необходимости обеспечения целостности информации, представленной в концептуальной модели.

Категории обладают особым свойством: все они повторно не пересекаются. В результате никакой экземпляр обобщения *Класс-документ* не представлен более чем в одной категории.

Фактически процесс организации алгебраической структуры классов в объектно-ориентированном подходе это не что иное, как классификация понятий, описывающих систему. Если топологическая структура классификации объектов — древовидная иерахия, то у класса может быть только один суперкласс, но несколько подклассов.

При топологической структуре классификации типа решетка у одного класса допускается наличие нескольких суперклассов. Причем класс наследует свойства всех суперклассов, а не только тех, которые находятся на ближайшем к нему уровне. В этом случае возникают различные стратегии наследования, связанные с порядком обхода классификационной решетки.

Однако наиболее часто используют две из них: стратегия наследования в глубину и стратегия наследования по уровням. В стратегии в глубину наследование осуществляется в соответствии с правилами сначала «сверху вниз» и затем «слева направо», а в стратегии по уровням — сначала «слева направо», а затем «сверху вниз».

Концепция наследования считается одной из основных особенностей объектно-ориентированного подхода [20]. Особенности классов состоят именно в том, что они могут наследовать и наследоваться.

6.5 Агрегация и декомпозиция понятий

Обобщение классов позволяет установить отношения между объектами вида *есть-некоторый*. Для того чтобы описать такие ситуации, когда один из объектов является составной частью другого объекта, в объектно-

ориентированном моделировании используется отношение *есть-часть*, т. е. кроме иерархии обобщения поддерживается иерархия агрегации.

Отношение *агрегации* имеет место между несколькими классами в том случае, если один из классов представляет собой сущность, которая включает в себя в качестве составных частей другие сущности. Данное отношение имеет фундаментальное значение для описания структуры сложных систем, поскольку применяется для представления системных взаимосвязей типа *часть-целое*. Раскрывая внутреннюю структуру системы, отношение *агрегации* показывает, из каких компонент состоит система, и как они связаны между собой.

С точки зрения концептуальной модели отдельные части системы могут рассматриваться как компоненты, или как подсистемы, которые, в свою очередь, тоже могут состоять из подсистем или компонент. Таким образом, данное отношение по своей сути описывает декомпозицию или разбиение сложной системы на более простые составные части, которые также могут быть подвергнуты дальнейшей декомпозиции, если в этом есть необходимость.

Очевидно, что рассматриваемое в таком аспекте деление системы на составные части представляет собой иерархию, но принципиально отличную от той, которая порождается отношением *обобщения*. Отличие заключается в том, что части системы никак не обязаны наследовать ее свойства и поведение, поскольку являются самостоятельными сущностями. Более того, части целого обладают собственными свойствами и операциями, которые существенно отличаются от свойств и операций целого. Кроме того, отношение агрегации требует, чтобы между компонентами и объектом существовала функциональная связь типа *N:1*. Это характеристическое свойство абстракции агрегации, отличающее ее от других видов абстракций. В частности, если даны два объекта, то функциональное соответствие можно

использовать для того, чтобы решить, может ли один из них быть компонентой другого. Например, *Группа* не может быть компонентой объекта *Курс лекций*, так как одна и та же группа студентов может слушать несколько курсов лекций, что нарушает функциональность связи между этими понятиями.

Следовательно, агрегация понятий используется в тех случаях, когда вновь порожденное понятие включает исходные понятия в качестве своих компонент или составных частей. Например, понятие АВТОМОБИЛЬ включает в качестве своих составных частей такие компоненты, как КУЗОВ, ШАССИ, ДВИГАТЕЛЬ.

Агрегация понятий — это такая форма связи понятий, при которой на основе исходных понятий P и Q образуется понятие агрегат R более высокого уровня, включающего в свой состав информацию, содержащуюся в понятиях P и Q. Причем, в отличие от абстракции обобщения, в абстракции агрегации допускается, что $intP \cap intQ = \varnothing$.

При агрегации вновь образованное понятие R наследует все свойства и операции входящих в него понятий P и Q, так что для интенсионалов и экстенсионалов выполняются следующие соотношения:

$$intR = intP \cup intQ,$$

$$extR = extP \times extQ.$$

Эти выражения можно интерпретировать следующим образом. Для принятия решения о принадлежности некоторой сущности экстенсионалу понятия-агрегата R необходимо, чтобы эта сущность удовлетворяла как интенсионалу понятия P, так и интенсионалу понятия Q. Легко видеть, что это требование удовлетворяется, если экстенсионал понятия-агрегата $extR$ определяется в соответствии с выражением $extR=extP \times extQ$.

Сущность $e_r \in extR$ обладает свойствами и поведением обоих понятий P и Q, что соответствует сформулированному выше правилу наследования для абстракции агрегации. Так, понятие АВТОМОБИЛЬ наследует все свойства и

поведение, относящиеся к понятиям ДВИГАТЕЛЬ, КУЗОВ и ШАССИ. При этом множество свойств понятия АВТОМОБИЛЬ является объединением свойств своих составных частей.

Пример иерархии агрегации представлен на рис. 6.3. Графически отношение *агрегации* изображается сплошной линией, один из концов которой представляет собой не закрашенный внутри ромб. Этот ромб указывает на тот класс, который представляет собой "целое" или класс-контейнер. Остальные классы являются его "частями" (рис 6.3).

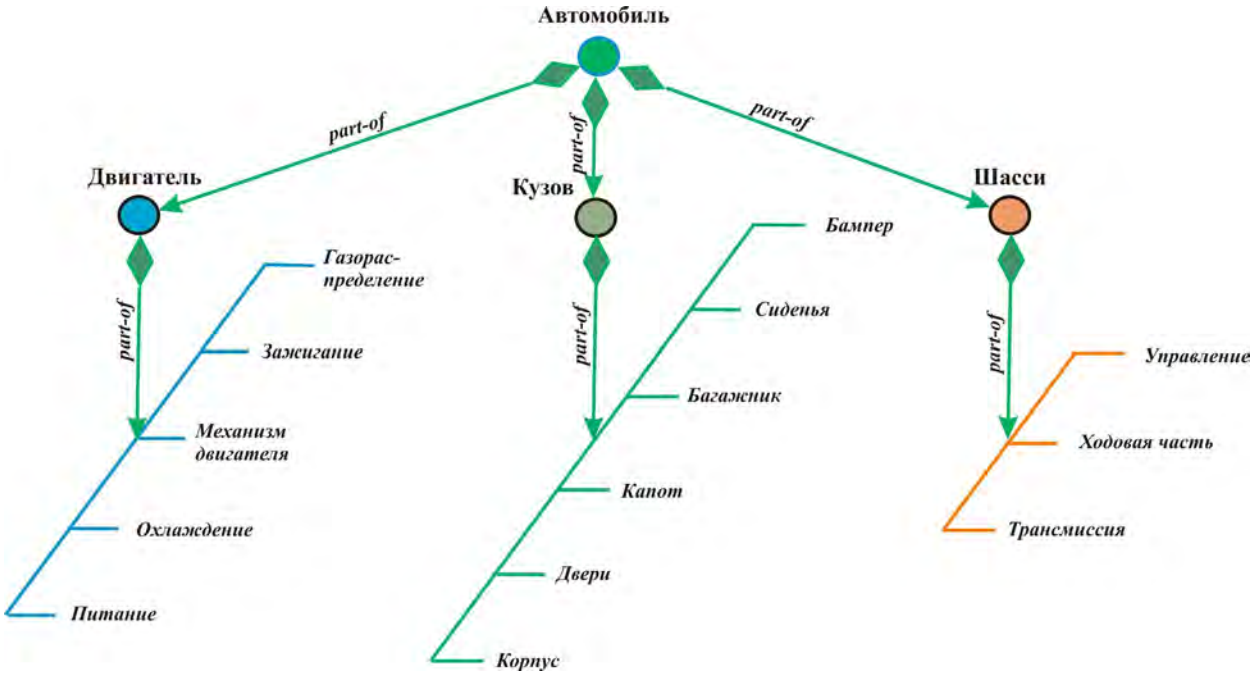


Рис. 6.3. Иерархия агрегации для понятия АВТОМОБИЛЬ

Так как свойства составляющих понятий наследуются понятием-агрегатом, то схемы понятий при абстракции агрегации связаны между собой выражением

$shmR = shmP \cup shmQ.$

Допустима, в частности, точка зрения, когда свойства также рассматриваются как составные части понятия. Это означает, что понятие является агрегатом, состоящим из своих свойств. Семантическое отличие агрегации свойств от агрегации понятий заключается в том, что свойства не

определяются как независимые понятия и, следовательно, могут быть представлены в описании системы только в том случае, когда определены соответствующие базовые понятия.

Семантическое отношение *часть-целое (part-of)*, выражаемое с помощью абстракции агрегации, обеспечивает возможность декомпозиции сложных, составных объектов системы. Объекты, принадлежащие составному объекту, называют *зависимыми*. Разновидность отношения *агрегации*, при которой составные части целого имеют такое же время жизни, что и само целое, называется *композицией***.** Эти части уничтожаются вместе с уничтожением целого.

В композиции зависимые объекты не могут существовать самостоятельно без тех составных объектов, в которые они входят как агрегаты, т. е. без наличия объектов-владельцев. Кроме того, зависимый объект не может быть создан до тех пор, пока не определен объект-владелец. При этом зависимый объект может быть как простым (не содержащим зависящих от него объектов), так и составным.

Отношение *композиции* - частный случай отношения *агрегации*. Это отношение служит для спецификации более сильной формы отношения *часть-целое*, при которой составляющие части тесно взаимосвязаны с целым. Особенность этой взаимосвязи заключается в том, что части не могут выступать в отрыве от целого, т.е. с уничтожением целого уничтожаются и все его составные части. Наглядным примером отношения композиции является живая клетка, в отрыве от которой не могут существовать ее составные части.

В UML графически отношение *композиции* изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот класс, который представляет собой класс-композит. Остальные классы являются его "частями".

Процессом, противоположным абстракции агрегации, является *декомпозиция.* При декомпозиции исходное понятие расчленяется на ряд независимых компонент, каждая из которых обладает лишь частью признаков исходного понятия.

Абстракция агрегации используется в тех случаях, когда необходимо синтезировать сложное понятие, состоящее из совокупности более простых понятий. Если в результате данного процесса мы получим понятия, состоящие из других понятий, то можно говорить об иерархии агрегации. Так, понятие КУРС может включать понятие ГРУППА, которое в свою очередь включает понятие СТУДЕНТ.

Иерархическая структура агрегации классов не обязательно является древовидной. Так, для класса объектов КАФЕДРА в качестве владельцев можно определить одновременно два класса АДМИНИСТРАЦИЯ и ФАКУЛЬТЕТ. Это будет означать, что в одних вузах кафедры непосредственно подчинены администрации, а в других — факультетам. Но на уровне экземпляров каждая кафедра вуза всегда подчинена либо администрации, либо факультету.

Так как экземпляр составного объекта образует иерархическую структуру типа дерево, то к каждой его компоненте может быть обеспечен доступ путем обхода этого дерева по определенному правилу (например, «сверху вниз», «слева направо»), а затем применить к выбранному экземпляру агрегата те операции, которые описаны в соответствующем классе. Кроме того, каждый экземпляр принадлежит экстенсионалу понятия-агрегата, что позволяет применить к нему операции, определенные для типа данных множество.

Семантика абстракции агрегации предполагает обязательную поддержку следующих определяющих соотношений. Каждый компонент агрегата должен именоваться, и должен существовать набор ключевых атрибутов, который позволяет определять компонент однозначно. Имена

существующих экземпляров компонент должны быть значениями соответствующих атрибутов понятия-агрегата

Таким образом, абстракция агрегации позволяет выразить семантику внутренних связей, существующих между отдельными элементами системы. При этом структура сложного понятия раскрывается путем его декомпозиции на совокупность составляющих понятий, называемых компонентами. Такая декомпозиция приводит к представлению анализируемого понятия в виде многоуровневой иерархической системы компонент, дающих описание внутреннего устройства сложного понятия.

Отличительная особенность агрегации по сравнению с обобщением состоит в порядке наследования свойств и операций понятий. Агрегация представляет такой процесс абстрагирования понятий, который можно рассматривать как синтез сложного понятия из более простых компонент. При этом понятие-агрегат, рассматриваемое как единое целое, включает все свойства и операции исходных компонент. В абстракции обобщения, напротив, понятия-категории наследуют свойства и операции обобщенного понятия. Если для обобщения наследование свойств и операций признаков для частных понятий производится от более общих понятий, то для агрегации, наоборот, — от простых понятий к более сложным.

Абстракция обобщения характеризуется переходом от конкретных понятий к более общим. Следовательно, в процессе обобщения игнорируются те свойства и операции исходных понятий, которые позволяют различать их между собой, и, наоборот, остаются только такие признаки и поведение, которые являются общими для всех этих понятий. Очевидно, что общие свойства и операции, используемые для построения интенсионала обобщенного понятия, могут относиться не только к данному обобщенному понятию, но и, в случае необходимости, наследоваться понятиями-категориями. Так, видовые понятия ПТИЦЫ и РЫБЫ характеризуются

общим свойством — обладают дыханием, который наследуется от родового понятия ЖИВОТНЫЕ.

Обобщенное понятие может иметь ряд дополнительных свойств, определяемых его схемой. Чаще всего в качестве дополнительных свойств выступают свойства, определяемые с помощью агрегатных функций: СРЕДНЕЕ, КОЛИЧЕСТВО, МАКСИМУМ, МИНИМУМ и т. д.

Если для агрегации характерно наследование свойств и операций понятий «снизу вверх», то для обобщения, наоборот, «сверху вниз». Например, все свойства и операции понятия ЛИЧНОСТЬ могут последовательно наследоваться понятиями СЛУЖАЩИЙ, ПРЕПОДАВАТЕЛЬ и АССИСТЕНТ. Наследование свойств и операций «сверху вниз» в иерархии обобщений позволяет избежать повторного указания большей части информации, так как соответствующие данные могут быть определены на верхних уровнях иерархии без их дублирования на нижних уровнях, которые содержат гораздо большее число элементов.

Иерархия агрегации в объектно-ориентированных моделях позволяет осуществить структуризацию сущностей системы и получить информацию об их взаимосвязях, которая дополняет знания, полученные в процессе построения классификационной схемы сущностей на основе иерархии обобщения. Взаимодействие абстракций агрегации и обобщения позволяет рассматривать каждый класс объектов либо как категорию в иерархии обобщения, либо как компонент в иерархии агрегации.

Использование абстракций обобщения и агрегации и правил наследования признаков обеспечивает мощные механизмы накопления и обработки информации об исследуемой системе. Абстракции агрегации и обобщения обеспечивают возможность структурированного описания сложной системы без дублирования информации и возникновения противоречий и, вместе с тем, позволяют путем использования процедур

логического вывода, основанных на правилах наследования признаков, произвести ее корректную переработку.

6.6 Ассоциация и индивидуализация понятий

Ранее мы рассматривали такие отношения между понятиями, для которых определены специальные правила, позволяющие по интенсионалам исходных понятий перейти к интенсионалу искомого понятия, а также указать формулы, позволяющие определить его схему на основе схем базовых понятий. Однако в концептуальном моделировании очень часто встречается ситуация, когда нет необходимости в такой сильной интеграции понятий, хотя определенные связи между понятиями следует отразить. Данная возможность обеспечивается путем использования абстракции ассоциации.

Отношение между двумя независимыми понятиями, при которой необходимо учесть статические взаимосвязи между экземплярами сущностей, принадлежащих экстенсионалам понятий одного или разных типов, называется *ассоциацией.* Для ассоциации одним из основных моментов является выделение того обстоятельства, что экстенсионал понятия-ассоциации R является подмножеством декартова произведения экстенсионалов исходных понятий P и Q

$$extR = extP \times extQ.$$

Связи между интенсионалами и схемами для ассоциации в общем случае не определяются, однако для каждого описания конкретной системы они должны быть специфицированы. Наиболее подходящей формой для указания данной информации является использование правил в виде хорновских дизъюнктов, что позволяет применять стандартные методы логического вывода.

Любая ассоциация обладает двумя ролями, каждая из которых указывает направление ассоциации. Роль в ассоциации может быть явно

поименована. Двунаправленная ассоциация позволяет выполнять операцию перехода от одного понятия к другому в обоих направлениях. При этом эти роли являются инверсными (обратными) по отношению друг к другу, подобно обратным функциям в математике.

Роль обладает также множественностью, которая показывает, сколько сущностей может участвовать в данной ассоциации. Обычно в ассоциации различают три варианта множественности между отдельными сущностями 1:1, 1:M и N:M, которые указывают верхнюю и нижнюю границы количества сущностей, участвующих в ассоциации. Существенно, что спецификация ассоциации не должна содержать указаний на способ ее физической реализации. Если же рассматривать точку зрения, отражающую модель реализации ассоциации, то можно исходить из предположения, что между связанными сущностями существуют явные указатели в обоих направлениях. Бинарная ассоциация в UML изображается сплошной линией с указанием числа множественности на ее концах.

При взаимнооднозначном отображении сущностей возникает ассоциация типа 1:1. Примером служит связь между понятиями ЛИЧНОСТЬ и ПАСПОРТ. Эти понятия связаны между собой бинарной ассоциацией с именем *Имеет*.

Ассоциация вида 1:M возникает в тех случаях, когда одна из сущностей позволяет идентифицировать несколько других сущностей. Однако если рассматривать данную связь в другом направлении, то можно указать только одну сущность, с которой связана каждая из сущностей ассоциации. Например, между понятиями ОТДЕЛ и СЛУЖАЩИЙ может быть определена ассоциация типа 1:M. Они связаны между собой бинарной ассоциацией *Работает*.

Бинарная ассоциация является частным случаем *n*-арной ассоциации, когда каждый экземпляр ассоциации представляет собой упорядоченный набор (кортеж), содержащий *n* экземпляров соответствующих понятий. N-

арная ассоциация связывает отношением несколько понятий, при этом одно понятие может участвовать в ассоциации более чем один раз.

В качестве примера тернарной ассоциации можно рассмотреть отношение между тремя понятиями: СОТРУДНИК, КОМПАНИЯ и ПРОЕКТ. Данная ассоциация указывает на наличие отношения между этими тремя классами, которое может представлять информацию о проектах, реализуемых в компании, и о сотрудниках, участвующих в выполнении отдельных проектов.

Ассоциация вида N:M возникает в тех случаях, когда в обоих направлениях ассоциации можно указать более одной сущности, с которой возможна связь, т. е. связи в ассоциации в обоих направлениях не уникальны. Примером такой ассоциации служит связь между ИЗДЕЛИЯМИ и ПОСТАВЩИКАМИ, когда одно ИЗДЕЛИЕ поставляется несколькими ПОСТАВЩИКАМИ и ПОСТАВЩИК может поставлять несколько ИЗДЕЛИЙ.

Если от ассоциации понятий осуществляется переход к отдельным понятиям, то такой процесс называют *индивидуализацией.* При этом происходит абстрагирование от имеющихся связей между двумя понятиями, что позволяет рассматривать их независимо друг от друга и, следовательно, значительно упростить представление информации о системе.

Когда абстракции применяются к системе многократно, то возникают иерархии представлений системы. Концептуальное представление сложной системы может включать абстракции, типизации, обобщений, агрегаций и ассоциации для всех понятий, которые требуются для точек зрения различных исследователей. В принципе некоторые из иерархий в концептуальном представлении могут быть определены отдельно. Однако большая степень прозрачности и понимаемости модели достигается, если структура системы определяется как единое целое.

На рис. 6.6 дан пример концептуального представления структуры системы электронного документооборота.

Сопоставление различных методов абстрагирования понятий показывает, что фундаментальными видами абстракций являются агрегация и обобщение. Абстракция типизации — лишь частный случай абстракции обобщения, когда игнорируются различия в интенсионалах и схемах обобщаемых понятий, а абстракция ассоциации эквивалентна абстракции агрегации по способу формирования экстенсионалов, но использует более частные методы комбинирования интенсионалов и схем, зависящие от исследуемой системы. Кроме того, семантика композиции заключается в том, что операция удаления понятия-агрегата приводит к удалению всех его компонент, в то время как при удалении понятия-ассоциации компоненты ассоциации должны сохраняться.

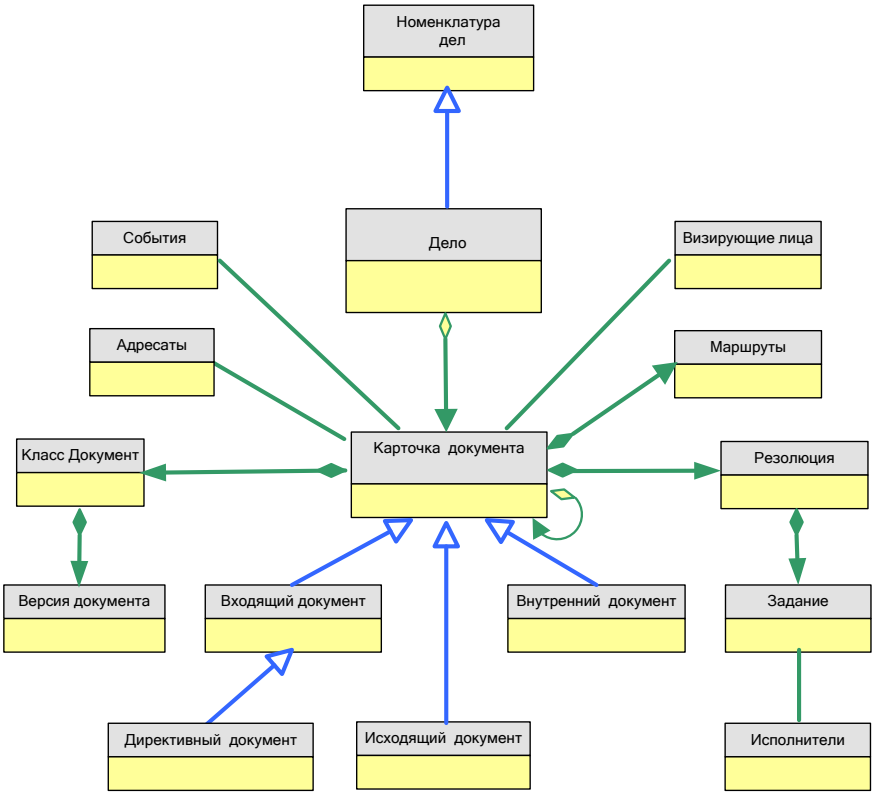


Рис. 6.6. Концептуальное представление структуры системы электронного документооборота.

Коренное различие между агрегацией и обобщением состоит в следующем. Агрегация обеспечивает понимание строения одних понятий

через строение других понятий, связанных с первыми определенным отношением. В обобщении, напротив, понимание строения вводимых понятий достигается через сравнение, сопоставление понятий между собой и выделение в них общей структуры, которая имеется во всех обобщаемых понятиях.

7 Моделирование динамики сложных систем

7.1 Модель прецедентов и прагматика использования систем

Рассмотренные выше понятия представляют собой модель статического представления состояний моделируемой системы. Речь идет о том, что с их помощью выражаются только взаимосвязи структурного характера, не зависящие от времени и реакции системы на внешние события. Абсолютно статичная система не представляет ни малейшего интереса, в ней ничего не происходит. Однако для большинства систем, кроме самых простых и тривиальных, статических представлений совершенно недостаточно для моделирования процессов их функционирования как в целом, так и их отдельных подсистем и элементов.

В целом ООМ можно представить как некоторый процесс постепенного перехода от наиболее общих абстрактных концептуальных представлений исходной системы к логическим и физическим представлениям. Так, например, для моделирования поведения в языке UML могут использоваться сразу несколько канонических моделей: прецедентов, сценариев деятельности, переходов состояний и последовательности взаимодействий объектов, каждая из которых фиксирует внимание на отдельном аспекте динамики системы.

Системы не существуют изолированно. Как правило, они взаимодействуют с другими системами, которые используют систему в своих целях, причем каждый участник взаимодействия ожидает, что система будет вести себя определенным, вполне предсказуемым образом. В UML на концептуальном уровне поведение моделируется с помощью прецедентов, специфицируемых в отрыве от реализации.

Прецедент представляет собой внешне наблюдаемую деятельность системы (то есть последовательность сообщений между системой и одной

или несколькими ролями). Всякий прецедент должен выполнять некоторый объем работы. С точки зрения участника взаимодействия, прецедент делает нечто представляющее для него определенную ценность, например, вычисляет результат, создает новый объект или изменяет состояние другого объекта.

Прецедент специфицирует поведение системы или ее части, представляя собой описание множества последовательностей действий (включая варианты), выполняемые системой для получения участником определенного результата. Следовательно, на системном уровне прецеденты представляют собой ответную реакцию системы на действия участника взаимодействия [11, 44].

В [61] прецедент определяется как совокупность всех сценариев поведения системы в ответ на запрос некоторого участника процесса, преследующего этим конкретную цель. Модель прецедентов, таким образом, представляет собой систему взаимосвязанных целей участников процесса и является основой для организации всей совокупности функциональных процессов системы, показывая взаимосвязи между участниками процесса и их целями, отражая прагматический характер использования системы.

Модель *прецедентов* описывает назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Поэтому обычно концептуальное описание системы должно начинаться с построения моделей прецедентов. Хорошо структурированные прецеденты описывают только существенное поведение системы или подсистемы.

Модель прецедентов является исходным концептуальным представлением системы в процессе ее описания. Разработка модели прецедентов преследует цели:

- Определить общие границы и контекст моделируемой системы на начальных этапах ее спецификации.
- Сформулировать общие требования к функциональному поведению моделируемой системы.
- Разработать исходное концептуальное представление системы для ее последующей детализации в форме сценариев функциональных процессов системы.

Прецедент (Use Case) – это типичное ответное действие системы на события, инициируемые извне. Он отражает представление о поведении системы с точки зрения окружающей среды. Суть модели прецедентов состоит в следующем. Моделируемая система представляется в виде множества внешних по отношению к системе действующих лиц или участников, взаимодействующих с системой с помощью прецедентов. Действующим лицом может быть пользователь-человек или внешняя система.

Действующее лицо – это идеализированная внешняя сущность (процесс или человек), вступающая во взаимодействие с системой или подсистемой. С помощью действующих лиц определяются те взаимодействия, которые могут осуществляться между системой и ее окружением. В реальном мире один участник взаимодействия может выполнять функции нескольких действующих лиц или *ролей*. И наоборот, несколько участников могут выполнять одну и ту же роль, то есть быть ее экземплярами.

Каждая роль может вступать во взаимодействие с одним или несколькими прецедентами. Это взаимодействие производится путем обмена сообщениями с системой или подсистемой, к которой относится прецедент. При этом не важно, как роль будет реализована в системе, достаточно описать свойства, которые определяют её состояние.

Роли можно описывать иерархически, постепенно дополняя общее описание более конкретными чертами. В модели прецедентов роли изображаются в виде схематического человечка, под которым указано его имя, а прецеденты – в виде эллипсов.

Прецедент описывает некоторую часть поведения системы, не вдаваясь при этом в особенности ее внутренней структуры. Прецедент определяет все виды поведения системы: основные последовательности, различные варианты стандартного и нестандартного поведения, исключительные ситуации, включая ответные реакции на них. С точки зрения действующих лиц, некоторые из видов поведения системы могут выглядеть как ошибочные. Однако для системы ошибочная ситуация является одним из вариантов поведения, который должен быть описан и обработан.

В процессе описания системы каждый прецедент моделируется независимо от остальных. Однако для реализации прецедентам могут понадобиться объекты, которые задействуются другими прецедентами. Так между прецедентами возникают неявные зависимости. В действительности, каждый из них представляет собой неотъемлемую часть деятельности системы, выполнение которой тесно связано с другими прецедентами [44].

В ответ на внешнее воздействие система выполняет некоторый законченный функциональный процесс, предоставляя действующим лицам определенный сервис. Другими словами, каждый прецедент определяет некоторый сценарий (набор действий), совершаемый системой при диалоге с действующим лицом при выполнении определенной целевой задачи. Модель прецедентов также показывает, какие действующие лица инициируют прецеденты и какие действующие лица получают данные в ходе выполнения прецедента.

Пример графического изображения модели прецедентов для банковского автомата показан на рис. 7.1.

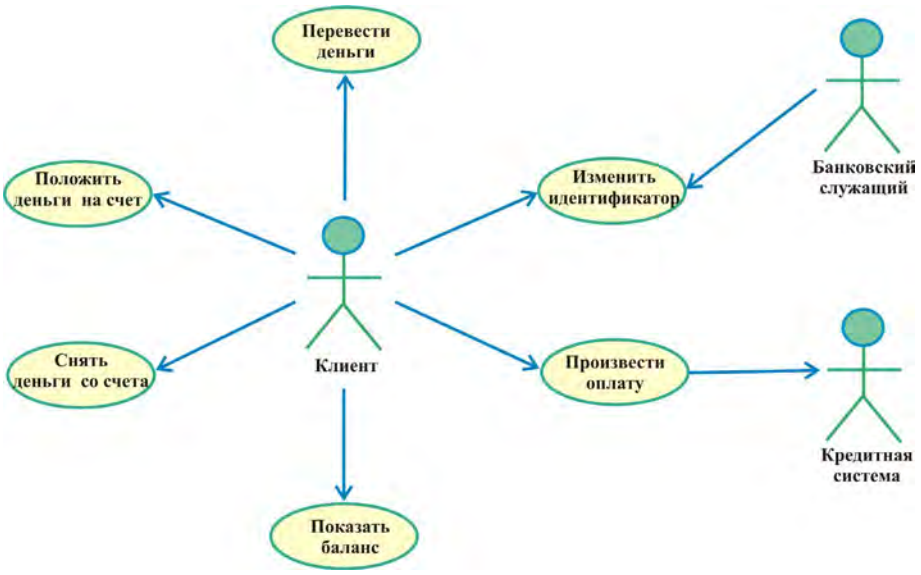


Рис. 7.1. Модель прецедентов для банковского автомата

В рамках данной модели действующее лицо *Клиент* инициирует различные прецеденты: *Снять деньги со счета, Перевести деньги, Положить деньги на счет, Показать баланс, Изменить идентификатор, Произвести оплату.* Действующее лицо *Банковский служащий* может инициировать прецедент *Изменить идентификатор клиента*. От прецедента *Произвести оплату* идет стрелка к *Кредитной системе*, которая является внешней для данного автомата. Стрелка, направленная от прецедента к действующему лицу, показывает, что прецедент предоставляет некоторую информацию действующему лицу. В данном случае прецедент *Произвести оплату* предоставляет *Кредитной системе* информацию об оплате по кредитной карточке. Из модели прецедентов можно получить довольно много информации о системе, так как она описывает общую функциональность системы. В результате все, кого интересует система в целом, могут, изучая модель прецедентов, понять, что система должна делать.

То есть модель прецедентов отображает взаимосвязь между выполняемыми системой функциями и действующими лицами,

получающими или передающими в данную систему информацию. Эта модель отражает требования к системе с точки зрения пользователя. Таким образом, прецеденты — это функции, выполняемые системой, действующие лица — это заинтересованные в системе субъекты а модель прецедентов показывает, какие действующие лица инициируют видимые извне функции системы. Модель позволяет визуализировать прецеденты отдельно от их реализации.

В модели прецедентов цели участников функциональных процессов являются своего рода основой понимания того, что именно требуется сделать, чтобы достичь той или иной цели. При использовании других методологий описания систем бывает трудно понять, зачем применяется тот или иной процесс, а также увидеть недостатки модели или самого функционального процесса — нужна ли эта функция, можно ли ее заменить другой и чьи интересы это затронет.

Элементы модели прецедентов являются первичными по отношению к тем, с помощью которых могут быть описаны сущности, такие как системы и подсистемы. Однако внутренняя структура этих сущностей в рамках данной модели никак не описывается.

Обычно сложная система содержит несколько десятков прецедентов, каждый из которых может разворачиваться в несколько десятков сценариев. Для любого прецедента необходимо выделить основные сценарии, описывающие важнейшие последовательности событий, и вспомогательные, описывающие альтернативные последовательности событий.

Сценарий - это некоторая последовательность действий, иллюстрирующая поведение системы. Сценарии находятся в таком же отношении к прецедентам, как экземпляры к классам, то есть сценарии - это экстенсионал прецедента. С помощью сценариев в принципе можно

отобразить прецедент целиком, включая основной и все альтернативные сценарии. При этом наиболее подходящим способом представления событий, образующих прецедент, является структурированный текст. Подобный подход описан, в частности, в [61].

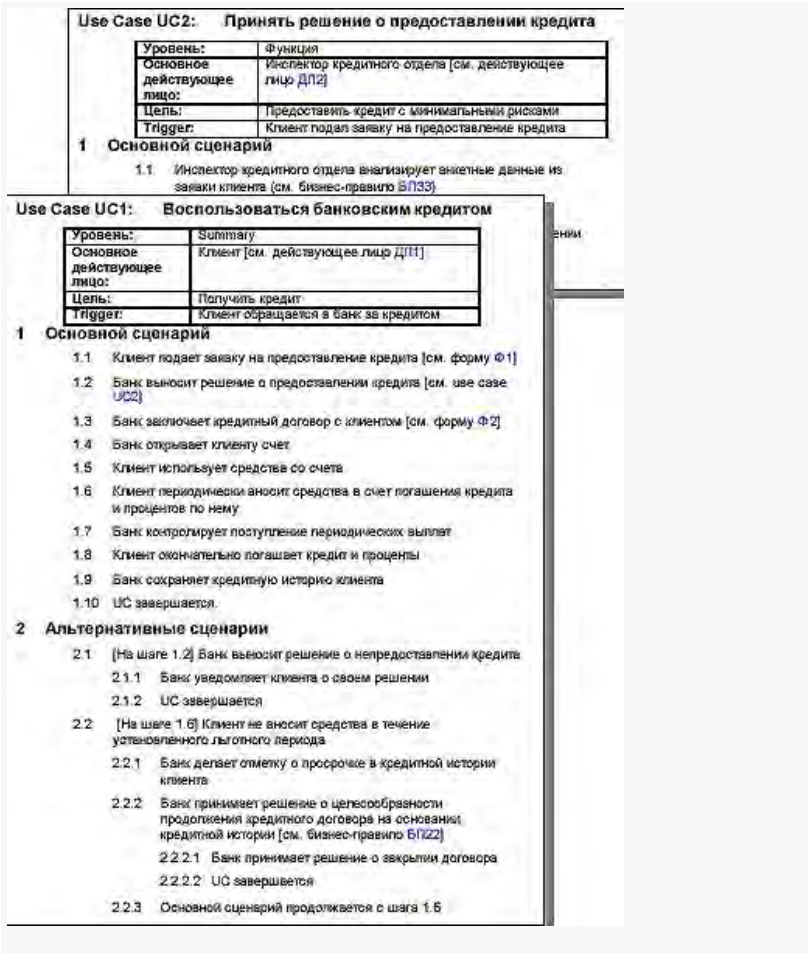
Поведение прецедента специфицируется путем описания потока событий в текстовой форме - в виде, понятном для пользователя. В описание прецедента необходимо включить указание на то, как и когда прецедент начинается и заканчивается, когда он взаимодействует с действующими лицами и какими объектами они обмениваются. Важно обозначить также основной и альтернативный потоки поведения системы. Это значит, что один прецедент описывает несколько последовательностей, или сценариев, каждый из которых представляет один из возможных вариантов реализации потока событий.

В качестве примера на рис. 7.2 приведена часть модели прецедентов, которая могла бы быть результатом описания работы кредитного отдела банка [2]. Она состоит из основного прецедента UC1 *Воспользоваться банковским кредитом* и связанного с ним прецедента UC2 *Принять решение о предоставлении кредита*.

Как видно из рис. 7.2, каждый прецедент указывает основное действующее лицо — участника процесса, преследующего некоторую цель и инициировавшего для этого взаимодействие с системой. Например, клиент, желающий получить в банке кредит, является основным действующим лицом в прецеденте UC1 *Воспользоваться банковским кредитом*.

Кроме того, модель содержит совокупность всевозможных сценариев, которые могут последовать при достижении действующим лицом своей цели. При этом единственный основной сценарий соответствует нормальному развитию событий, то есть такому, при котором не возникает никаких

нештатных ситуаций. Например, в прецеденте UC1 *Воспользоваться банковским кредитом* основным является сценарий, при котором банк дает положительное решение о предоставлении кредита, клиент получает кредит, а затем погашает его по установленному графику.



Use Case UC2: Принять решение о предоставлении кредита

Уровень:	Функция
Основное действующее лицо:	Инспектор кредитного отдела [см. действующее лицо ДЛ2]
Цель:	Предоставить кредит с минимальными рисками
Trigger:	Клиент подал заявку на предоставление кредита

1 Основной сценарий

1.1 Инспектор кредитного отдела анализирует анкетные данные из заявки клиента (см. бизнес-правило БП33)

Use Case UC1: Воспользоваться банковским кредитом

Уровень:	Summary
Основное действующее лицо:	Клиент [см. действующее лицо ДЛ1]
Цель:	Получить кредит
Trigger:	Клиент обращается в банк за кредитом

1 Основной сценарий

1.1 Клиент подает заявку на предоставление кредита [см. форму Ф1]
1.2 Банк выносит решение о предоставлении кредита [см. use case UC2]
1.3 Банк заключает кредитный договор с клиентом [см. форму Ф2]
1.4 Банк открывает клиенту счет
1.5 Клиент использует средства со счета
1.6 Клиент периодически вносит средства в счет погашения кредита и процентов по нему
1.7 Банк контролирует поступление периодических выплат
1.8 Клиент окончательно погашает кредит и проценты
1.9 Банк сохраняет кредитную историю клиента
1.10 UC завершается.

2 Альтернативные сценарии

2.1 [На шаге 1.2] Банк выносит решение о непредоставлении кредита
2.1.1 Банк уведомляет клиента о своем решении
2.1.2 UC завершается
2.2 [На шаге 1.6] Клиент не вносит средства в течение установленного льготного периода
2.2.1 Банк делает отметку о просрочке в кредитной истории клиента
2.2.2 Банк принимает решение о целесообразности продолжения кредитного договора на основании кредитной истории [см. бизнес-правило БП22]
2.2.2.1 Банк принимает решение о закрытии договора
2.2.2.2 UC завершается
2.2.3 Основной сценарий продолжается с шага 1.6

Рис. 7.2. Описание прецедентов кредитного отдела банка

Альтернативные сценарии начинаются так же, как и основной, но в какой-то момент развитие событий идет по другому пути. Например, это может быть сценарий, при котором клиент не выплачивает очередной взнос, предусмотренный графиком погашения кредита (рис. 7.2, UC1, п. 2.2). При этом опять же вероятны альтернативные сценарии более низких уровней. Так, если клиент в течение льготного срока восстанавливает график погашения, то сценарий воссоединяется с основным сценарием, если же клиент не выплачивает задолженность, кредит считается просроченным и т. д.

Каждый прецедент состоит из ряда шагов. Причем на каждом шаге определенный участник процесса также преследует свою определенную цель, т. е. каждый шаг сценария может являться прецедентом более низкого уровня. Например, при кредитовании банку необходимо принять обоснованное решение о возможности выдачи кредита с целью минимизации риска. Процесс принятия такого решения в модели отражен в описании прецедента UC2 (рис. 7.2).

Последовательно разрабатывая множество таких прецедентов, мы постепенно получаем модель функциональных процессов, построенную вокруг действующих лиц и их целей. Причем на каждом шаге мы имеем модель определенного уровня детальности, которая, скорее всего, не будет изменяться при уточнении модели — к ней лишь будут добавляться дополнительные уровни при детализации шагов сценариев в модели.

Динамическая модель системы, помимо прецедентов и действующих лиц, может содержать и другие элементы, помогающие лучше понять предметную область системы или обосновывать те или иные модельные решения. Описание UC1, например, ссылается на формы документов (пп. 1.1 и 1.3) и на правила предметной области системы (п. 2.2.2).

Прецеденты могут быть применены ко всей системе или к ее части, в том числе к подсистемам или даже к отдельным классам. Внутренний прецедент представляет собой поведение какого-либо элемента системы. Например, прецедент использования класса представляет собой некую деятельность, которую класс совершает по отношению к другим классам системы. Каждый класс может иметь несколько прецедентов.

Прецедент является логическим описанием определенной части деятельности системы. Он не представляет собой четкую конструкцию, которую можно напрямую реализовать. Реализацию прецедента можно смоделировать в виде одной или нескольких моделей последовательности

взаимодействия объектов. При этом каждый прецедент отображается на классы, необходимые для реализации системы. Поведение, описанное в прецеденте, переносится на переходы и операции классов. Каждый класс может играть несколько ролей в реализации системы и участвовать в реализации нескольких прецедентов.

7.2 Структурная организация прецедентов

Структурная организация прецедентов может выполняться по-разному. В любой хорошо продуманной системе существуют прецеденты, которые либо являются специализированными версиями других, более общих, либо входят в состав прочих прецедентов, либо расширяют их поведение.

Общее поведение множества прецедентов, допускающее повторное применение, можно выделить, организуя их в соответствии с абстракцией обобщения. В частности, при моделировании работы банка базовый прецедент, описывающий обработку кредита клиентов, подразделяется на несколько сценариев, от оформления крупной закладной до выдачи кредита. Однако все эти прецеденты имеют нечто общее в особенностях поведения, например оценку платежеспособности клиента.

Для сложной системы прецеденты можно организовать, определив между ними отношения *ассоциации, обобщения, включения и расширения* [44]. Эти отношения применяют, чтобы выделить некоторое общее поведение (извлекая его из других прецедентов) или, наоборот, поместив такое поведение в другие прецеденты. Допустимые типы отношений для прецедентов представлены в табл. 7.1.

Табл. 7.1. Допустимые типы отношений для прецедентов

Отношение	Функция	Нотация
Ассоциация (Association)	Отношение, указывающее на связь между действующим лицом и прецедентом	
Обобщение (Generalization)	Отношение между более общим и специфическим прецедентом, наследующим черты общего	
Расширение (Extend)	Включение добавочного поведения в исходный прецедент. Включаемый прецедент существует автономно от исходного прецедента.	
Включение (Include)	Включение добавочного поведения в исходный прецедент, который явно описывает включение. Включаемый прецедент существует только как часть исходного прецедента.	

Отношение ассоциации показывает некоторую общую связь между действующим лицом и прецедентом.

Отношение обобщения между прецедентами аналогично отношению обобщения между классами. Это означает, что прецедент-потомок наследует поведение и семантику своего родителя, может замещать его или дополнять его поведение, а, кроме того, может быть подставлен всюду, где появляется его родитель (как родитель, так и потомок могут иметь конкретные экземпляры). Обобщение между прецедентами изображается точно так же, как и обобщение между классами - в виде линии с незакрашенной стрелкой.

Отношение включения (include) между прецедентами означает, что в некоторой точке исходного прецедента использовано поведение другого прецедента. Включаемый прецедент никогда не существует автономно, а только как часть объемлющих его прецедентов. Можно считать, что исходный прецедент заимствует поведение включаемого, что аналогично

отношению композиции для классов, с учетом того обстоятельства, что жизненный цикл включаемого прецедента завершается только с удалением последнего исходного прецедента.

Благодаря наличию отношений включения удается избежать многократного описания одного и того же потока событий, поскольку общее поведение можно описать в виде самостоятельного прецедента, включаемого в исходный. Отношение включения является примером делегирования, при котором ряд обязанностей системы описывается в одном месте (во включаемом прецеденте), а остальные прецеденты, когда необходимо, включают эти обязанности в свой набор.

Один прецедент может также рассматриваться как расширение и дополнение исходного прецедента. *Отношение расширения (extend)* подразумевает, что исходный прецедент неявно содержит поведение другого прецедента в точке, которая косвенно задается расширяющим прецедентом. Исходный прецедент может быть автономным, но при определенных обстоятельствах его поведение расширяется за счет другого. Можно считать, что расширяющий прецедент передает свое поведение исходному, но в отличие от отношения включения расширяющий прецедент не исчезает при удалении исходного, что соответствует отношению агрегации для классов. У исходного прецедента может быть несколько расширяющих прецедентов, которые вносят дополнения в его семантику. Все эти варианты могут применяться вместе.

Отношение расширения применяют для моделирования таких частей прецедента, которые действующее лицо воспринимает как необязательное поведение системы. Тем самым можно разделить обязательное и необязательное поведение. Отношения расширения используются также для моделирования отдельных субпотоков, выполняемых лишь при определенных обстоятельствах. Наконец, их применяют для моделирования

нескольких потоков, которые могут включаться в некоторой точке сценария в результате явного взаимодействия с действующим лицом.

Организация прецедентов путем выделения отношения включения и отношения расширения является важной составной частью процесса разработки простого, сбалансированного и понятного набора прецедентов сложной системы [44].

В целом, совокупность прецедентов является не чем иным, как оглавлением к модели сложной системы. В модели прецедентов могут быть показаны участники процессов и цели, которые они преследуют, могут быть показаны взаимосвязи прецедентов, и промежуточные цели, которые могут достигаться при достижении цели более высокого уровня. Однако прецедент не отображает целиком структуру того или иного процесса в виде последовательности некоторых действий. Эта задача решается в модели сценариев функциональных процессов.

С помощью прецедентов можно описать поведение системы, не определяя ее реализацию. Прецеденты специфицируют желаемое поведение, но ничего не говорят о том, как его достичь. И это позволяет обсуждать характеристики системы, не углубляясь в детали реализации. Подробности могут быть рассмотрены позже, а на этапе концептуального описания системы можно сконцентрироваться на наиболее существенных проблемах.

7.3 Сценарии функциональных процессов системы

Важную роль при построении моделей, описывающих динамические (поведенческие) аспекты системы, играет модель сценариев ее функциональных процессов, отражающая результаты функционирования системы в целом, то есть специфицирующая систему на стратегическом уровне. Главное назначение моделей этого уровня заключается в описании

основных задач, выполняемых системой, и в формировании моделей сценариев процессов, реализующих основную деятельность системы и обеспечивающих достижение некоторого результата для действующих лиц.

Любая сложная система может быть представлена как совокупность взаимодействующих подсистем, каждая из которых может иметь свою структуру. Подсистемы связаны между собой функционально, т.е. они выполняют отдельные виды задач или работ в рамках единого процесса, а также информационно, обмениваясь определенными сообщениями. Кроме того, эти подсистемы взаимодействуют с внешними системами, причем их взаимодействие также может быть как информационным, так и функциональным. И эта ситуация справедлива практически для всех типов сложных систем, каким бы видом деятельности они не занимались.

При моделировании сложной системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности логической последовательности реализации выполняемых системой действий и операций. В ООМ эта задача решается путем создания моделей сценариев функциональных процессов. Каждый такой сценарий акцентирует внимание на последовательности выполнения определенных действий, которые в совокупности приводят к получению желаемого результата.

Сценарии процессов определяют прохождение потоков действий (функций) независимо от иерархии и границ подсистем, которые их выполняют, и отражают как внутренние, так и внешние взаимосвязи системы, т.е. взаимодействие системы со средой. При таком подходе нет необходимости в первоочередном описании организационного строения системы, описывающего иерархию подсистем, и устанавливающего функциональные и административные границы между подсистемами. Такое описание может быть выполнено позднее.

Необходимо отметить, что с увеличением сложности системы строгое соблюдение последовательности выполняемых действий приобретает все

более важное значение. Так, например, отклонение от штатной последовательности действий при взлете или посадке авиалайнера, запуске ракеты, регламентных работах на АЭС может привести к катастрофическим последствиям. Здесь можно напомнить об аварии на Чернобыльской АЭС, произошедшей именно вследствие нарушения предписанной процедуры проведения регламентных работ.

В сценарии процесса могут присутствовать разветвления, развилки и слияния параллельных потоков управления. Параллельные потоки представляют собой деятельности, которые объекты системы или участники могут совершать одновременно. Параллельные деятельности могут осуществляться как одновременно, так и в любой последовательности. В целом, сценарий процесса напоминает обычную блок-схему, с одним существенным отличием — в нем допускается не только последовательное, но и параллельное управление потоком работ [62].

При формализации модели функциональных процессов должны быть четко определены соображения, по которым те или иные черты сложной системы отражаются в модели или исключаются из нее. При этом необходимо сформировать определённый, достаточно строгий взгляд на то, что такое функциональный процесс, из чего он состоит и каков его жизненный цикл. Такой подход требует введения новых понятий (абстракций), позволяющих описать функциональный процесс как в статике, так и в динамике.

Данная задача была поставлена перед инициативной группой (Business Relocess Management Initiative(BMPI)) по управлению процессами, которая разработала стандарт Business Relocess Modeling Notation (BPMN). Основное назначение стандарта – создание нотации, понятной всем действующим лицам и участникам описания систем, в виде совокупности графических объектов, используемых для моделирования процессов.

Другой не менее важной целью стандарта является визуализация посредством нотации BPMN языка реализации процессов Business Relocess Execution Language for Web Services (BPEL4WS). Таким образом, BMPN является связующим звеном между разработкой (спецификацией) процессов и их реализацией.

Спецификация BMPN [62] раскрывает понятия и определяет семантику схем процессов и объединяет лучшие методы, разработанные в сфере моделирования процессов. Цель BPMN – стандартизировать нотацию моделирования функциональных процессов. В терминологии BPMN процесс является сложным действием, которое, в свою очередь, состоит из действий, переходов между ними и т. д. Процесс можно вызывать, приостанавливать, прерывать, также он может завершаться сам, процессы могут выполняться параллельно и обмениваться сообщениями.

Применяемая при описании процессов графическая нотация BPMN во многом похожа на нотацию UML для модели переходов состояний, поскольку в сценарии также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления состояний действий, а не состояний объектов системы. Каждое состояние действия в модели сценария процесса соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии. Графически модель сценария процесса может быть представлена в форме графа, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому.

Механизм сценариев процессов позволяет описать динамику системы в целом, путем отделения описания функциональной логики процесса (логики сценария) от реализации действий. Логика сценария процесса отвечает за

вызовы отдельных функций системы и помогает связать отдельные подпроцессы в единую систему путем интеграции многих подсистем.

7.4 Средства спецификации сценариев процессов

Ключевой элемент BPMN – это выбор форм и значков, используемых в графических элементах, указанных в данной спецификации. Цель BPMN – создание стандартного визуального языка, который будет узнаваем и понятен для всех действующих лиц. Так как схема BPMN может отображать процессы разных участников, то некоторые действия с точки зрения участника будут внутренними (выполняться самим участником или под его контролем), а другие действия - внешними.

Следует подчеркнуть, что одним из особенностей стандарта BPMN является создание простого механизма для создания моделей сценариев, в то же время способного к отображению сложных реальных процессов. Способ решения проблемы сочетания этих двух противоречащих друг другу требований состоит в создании графических аспектов нотации по конкретным категориям. При этом совокупность категорий нотации получается небольшой, так что можно легко узнать основные типы понятий и правильно интерпретировать схему.

В BPMN можно выделить четыре основные категории понятий:

- *Объекты процесса (flows objects).* Объекты процесса – основные графические элементы, которые служат для определения поведения процесса. В BPMN представлены три типа объектов процесса: *функции или действия (activity)* – ссылаются на работу, которая исполняется, либо как единая задача, либо как набор задач (подпроцесс), *шлюзы или соединения (gateway)* действий друг с другом,

события (event) – используются для представления того, что происходит в процессе деятельности*;*

- *Связи элементов процесса (connectors).* Объединяют разные действия и данные в единый поток исполнения процесса. Существует три способа соединения объектов друг с другом или с другой информацией: *последовательный переход (sequence flow)* – определяет последовательный порядок перехода от одного действия к другому, *поток сообщений (message flow)* – отражает обмен сообщениями между действиями и подпроцессами, расположенными в различных подсистемах процесса, *ассоциация (association)* – определяет различные вспомогательные связи действий процесса (например, связи действий с артефактами, комментариев с любым элементом диаграммы, откат действия, возникновение исключительной ситуации или ошибки);

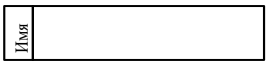
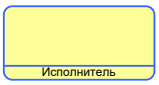
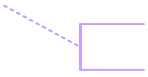
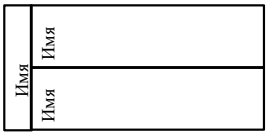
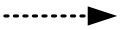
- *Артефакты процесса (artifacts).* Артефакты – это результаты деятельности, которые используются для введения дополнительной информации по процессу. Существует три стандартных артефакта: *объекты данных (data object), группы (groups), комментарии (annotations);*

- *Подсистемы процесса (swimlanes).* Используются для разграничения процессов и систем. Существует два способа группировки основных элементов моделирования посредством этих понятий: *внешние подсистемы (pools)* – используются, чтобы выделить внешние подсистемы, для которых важна только стандартная реакция, определяемая теми ролями, которые они играют, и несущественны

детали их индивидуальных внутренних процессов, *и внутренние подсистемы или дорожки (lanes)* – используются для группирования действий по функциям или ролям участников.

Графическое изображение перечисленных основных понятий моделирования, описанных в нотации BPMN, представлено в табл. 7.2.

Табл. 7.2. **Основные понятия моделирования, описанные в нотации BPMN,**

Объекты	**Связи**	**Артефакты**	**Подсистемы**
Шлюзы/порты (соединения) 	Последовательный переход 	Объекты данных 	Внешняя подсистема 
Действия 	Поток сообщений 	Комментарий 	Внутренняя подсистема 
События 	Ассоциация 	Группы действий 	

Рассмотрим более детально нотацию *объектов процесса,* используемую BPMN: *действий, шлюзов/портов и событий*.

Действие (activity) - это атомарная функция процесса, неделимая на более элементарные части. В BPMN возможны три типа действий:

- *циклическое действие (loop)* - это действие, которое выполняется в цикле. В параметрах этого дествия можно указать, какой цикл имеется в виду - с пред- или постусловием, определить это условие и указать некоторые дополнительные свойства цикла;

- *множественное действие (multiple instance)* - это циклическое действие, которое выполняет в цикле целый набор однотипных задач. Текстовыми параметрами можно задать условие цикла, количество однотипных задач, а также последовательный или параллельный порядок их выполнения;

- *откат (compensation)* – это действие, которое вызывается в случае отмены задачи.

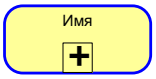
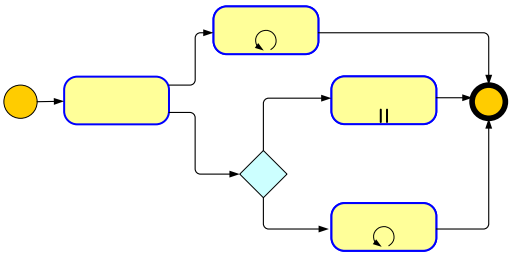
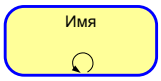
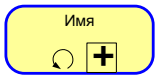
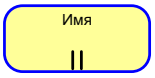
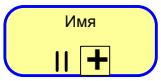
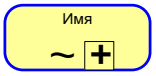
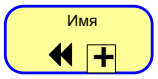
Кроме того, у действия могут быть определены атрибуты, которые не имеют графического представления и могут быть отражены, например, в имени действия.

Еще одним типом действия является *подпроцесс*. Он позволяет разбить сложные процессы на более мелкие. Допускаются *свернутые* и *развернутые* подпроцессы. Так же как и действия, подпроцессы могут быть циклическими, множественными и с откатом. Возможен также тип произвольного подпроцесса. Он означает, что действия и другие подпроцессы, входящие в состав данного, исполняются в произвольном порядке.

Свернутый подпроцесс является ссылкой на другую диаграмму модели, где он определяется в виде действий и, возможно, других подпроцессов. Развернутый подпроцесс позволяет задать прямо на родительской диаграмме один или несколько детализированных подпроцессов.

Различные типы действий, используемые в BPMN, представленных в табл. 7.3.

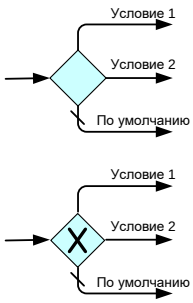
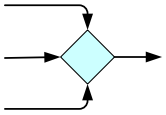
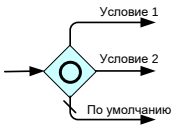
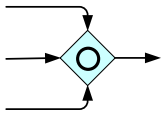
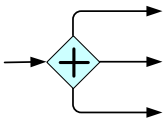
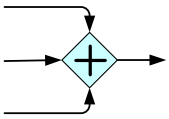
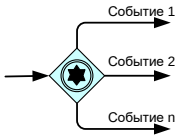
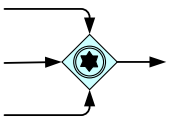
Табл. 7.3 Типы действий, используемые в BPMN

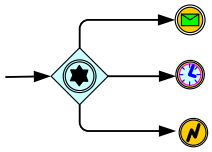
Действия	Свернутый подпроцесс	Развернутый подпроцесс
Действие 	Простой свернутый подпроцесс 	
Циклическое действие 	Циклический свернутый подпроцесс 	
Множественное действие 	Множественный подпроцесс 	
	Произвольный подпроцесс 	
Откат действия 	Откат подпроцесса 	

Шлюзы и порты (соединения) используются для контроля расхождения и схождения последовательного потока, которое может обозначать ветвление, раздвоение, слияние и соединение маршрутов действий или подпроцессов. Внутренние маркеры внутри графического изображения шлюза/порта указывают на тип контроля управления процессом.

Графическое изображение различных вариантов соединений и их описание представлено в табл. 7.4.

Табл. 7.4. Графическое изображение шлюзов и портов

Тип соединения	Шлюзы	Порты	Описание
Основанное на данных	Альтернативное ветвление по условиям 	Альтернативное соединение 	Для альтернативного ветвления по условиям BPMN предлагает два варианта нотации - обычный ромбик и ромбик с крестиком внутри. Первый вариант удобен, если никаких других типов соединений в сценарии нет, второй - если на диаграмме есть иные варианты соединений. В этом случае ромбик с крестиком используется, чтобы разные ромбики можно было легко отличать друг от друга. Альтернативное соединение означает объединение разных веток процесса в соответствии с логическим оператором OR.
	Параллельное ветвление с условием 	Параллельное соединение потоков 	Параллельное ветвление с условием обеспечивает ветвление потока управления по всем веткам, логические условия которых оказались выполнены к моменту проверки. Порт параллельного соединения с условием ждет все те потоки, которые были до этого запущены, а не вообще все потоки, которые направлены к данному порту.
Ветвление и объединение потоков управления	Параллельное ветвление потока 	Параллельное соединение потоков 	Данный тип соединения обеспечивает распараллеливание и синхронизацию параллельных потоков управления (выполнение логической операции AND).
Основанное на событиях	Настраиваемый разветвитель потока 	Настраиваемый соединитель потоков 	Этот тип соединения обеспечивает более сложную семантику ветвления потоков управления. Поэтому он должен обязательно сопровождаться некоторым выражением, точно определяющим семантику соединения. Например, можно определить такой порт как соединяющий три параллельных потока, с условием, что он "пропускает" выполнение процесса дальше, если дождался любых двух из трех.

	Логический переключатель по событиям 		Данный тип соединения переключает поток управления в зависимости от получения того или иного события. Можно выбрать только один из альтернативных маршрутов. Сами события обозначаются в начале соответствующей ветки. В качестве порта этот соединитель не используется.

Событие (event) - это некоторое изменение состояния системы, возникшее во время исполнения процесса. События влияют на ход процесса и обычно имеют предусловие (триггер) или постусловие (воздействие или результат). Событиями могут быть инициация/завершение процесса, прием/посылка сообщения, завершение какого-либо действия или подпроцесса и т. д. События способны влиять на порядок выполнения процесса, активировать и прерывать те или иные его действия.

На диаграммах сценариев процессов события изображаются в виде кругов с внутренними маркерами для отражения типа пред- или постусловия события. Существует три типа событий, классифицированных по времени воздействия на ход процесса:

- *начальное (start)* - событие, с которого начинается процесс или подпроцесс;
- *промежуточное (intermidiate),* которое непосредственно случается внутри процесса;
- *конечное (end),* наступление которого означает завершение процесса или подпроцесса.

Графическое изображение событий показано в табл. 7.5.

Табл. 7.5. Нотация событий в BPMN.

Виды событий	Начальные события	Промежуточные события	Конечные события
Общий вид событий			
Прием, посылка сообщений			
Истечение определенного промежутка времени (таймер)			
Наступление исключительного события, ошибка			
Отмена действия и возврат процесса к состоянию, которое было до начала исполнения этого действия			
Выполнение специальных отменяющих действий, компенсация			
Выполнение правила			
Переключение процессов			
Множественный триггер - при наступлении некоторого события порождает несколько других событий			
Завершение процесса			

Внизу, сразу под символом события, указывается его имя или источник. В BPMN существуют многочисленные правила, которые определяют детали того, где и при каких условиях может размещаться то или иное событие.

В качестве примера использования нотации BPMN рассмотрим модель описания функционального процесса для системы электронного

документооборота. Документооборот представляет собой процесс движения документов с момента их создания или получения от других организаций до завершения исполнения и сдачи в архив.

В рамках процесса электронного документооборота обеспечивается контроль жизненного цикла всех документов, регистрация и учет исполнительных документов, поступающих от структурных подразделений и внешних организаций, и принимаются решения по их согласованию, утверждению и дальнейшему исполнению.

В качестве документа, инициирующего процесс документооборота, выступает входящая корреспонденция. Продуктом документооборота является исходящая корреспонденция. Процесс документооборота производится на основании регламентов согласования, определенных в нормативных документах организации.

Язык, на котором должна быть описана модель общего документооборота, должен включать в себя следующие ключевые элементы нотации:

- шаги (задачи) обработки документов;
- участвующие в процессе подразделения и роли пользователей;
- условные и безусловные переходы между шагами процесса;
- сообщения, передаваемые между шагами процесса;
- данные (артефакты), передаваемые при переходе между шагами процесса;
- события, вызывающие переход к данному шагу процесса;
- подпроцессы.

На диаграмме (рис. 7.3) с использованием нотации стандарта BPMN представлен обобщенный процесс системы электронного документооборота организации.

Как следует из рис. 7.3 типовая система электронного документооборота включает следующие основные подпроцессы:

- прием и регистрация входящей корреспонденции,
- наложение резолюций и выбор исполнителей,
- ознакомление с входящим/внутренним документом,
- подготовка, утверждение и регистрация исходящей/внутренней корреспонденции,
- передача документов в архив,
- контроля исполнения (постановки на контроль, промежуточного контроля и снятия с контроля.

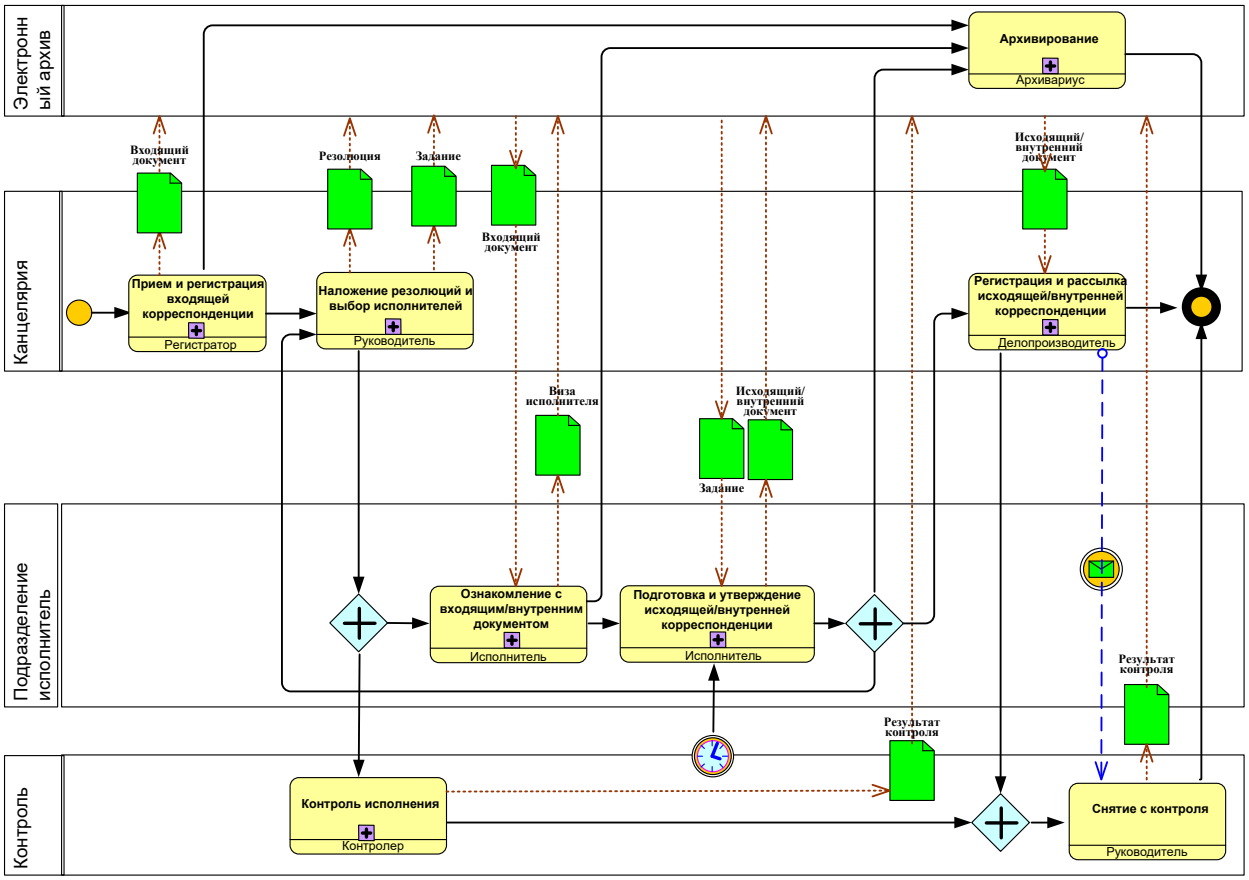


Рис. 7.3. Обобщенная модель процессов системы электронного документооборота

Кроме того, на диаграмме модели системы электронного документооборота также органически представлены, внутренние подразделения организации и основные артефакты, которые должны быть

получены в результате функционирования системы, что обеспечивает интегрированное представление о данной системе в целом.

7.5 Модель переходов состояний как конечный автомат

Модель переходов состояний описывает динамическое поведение объектов системы, отражая их жизненные циклы [44]. Каждый объект при этом рассматривается как изолированная сущность, которая общается с другими объектами, реагируя на различные события. Событиями являются любые виды изменений, которые могут повлиять на объект и которые он способен распознать (получение вызова, передача сигнала, изменение значений определенных свойств, истечение промежутка времени).

Именно для описания реакции объекта на подобные внешние события и используются модели переходов состояний. Каждая модель переходов состояний представляет некоторый *автомат*, используемый для представления поведения моделируемой сущности в виде дискретного пространства с конечным числом *состояний* и *переходов* из одного состояния в другое. Состояния - это метод запоминания *предыдущих* событий, а переходы - метод организации реагирования на *будущие* события.

Понимание семантики понятия состояния представляет определенные трудности. Дело в том, что характеристика состояний системы не зависит (или слабо зависит) от логической структуры, зафиксированной с помощью взаимосвязей понятий. Поэтому при рассмотрении состояний системы необходимо абстрагироваться от особенностей ее объектной структуры и использовать совершенно другие категории, относящиеся к динамическому контексту моделируемой системы.

Состоянием называется множество объектов одного класса, которые реагируют на событие одинаковым образом, совершают одно и тоже действие. Состояние может быть задано в виде набора конкретных значений свойств объекта, при этом изменение их отдельных значений будет отражать

изменение состояния моделируемого объекта. Таким образом, состояние стабильно до тех пор, пока над объектом не будет произведено действие и если над объектом выполняется некоторое действие, его состояние может измениться.

При этом не каждое свойство объекта необходимо учитывать чтобы характеризовать его состояние в рамках модели переходов состояний. Как правило, имеют значение только такие свойства элементов системы, которые отражают специфический семантический аспект ее поведения (динамический или функциональный). В этом случае состояние будет характеризоваться некоторым инвариантным условием, определяющем только значимые для поведения объекта свойства, которые тем самым задают целый класс микросостояний объекта, выделяя этот класс в виде макросостояния или фазового состояния объекта.

Инвариант может определять статическую ситуацию, когда объект находится в состоянии ожидания появления некоторого внешнего события. С другой стороны, инвариант используется для моделирования динамических аспектов, когда в ходе процесса выполняются некоторые действия. В этом случае моделируемый элемент переходит в рассматриваемое состояние в момент начала соответствующей деятельности и покидает данное состояние в момент ее завершения. Примером инвариантного условия может быть состояние ожидания объектом такого внешнего события, как запрос или передача управления.

В разных фазовых состояниях объекты на одно и тоже событие могут реагировать по-разному и выполнять при этом различные действия. Это значит, что объекты осуществляют семантическую обработку событий, зависящую от того состояния, в котором они находятся. Так, например, объект самолет может характеризоваться следующими фазовыми состояниями: *Находится в ангаре, На рулевой дорожке аэродрома, На взлетно-посадочной полосе, Взлет, Полет, Приземление.* Операции, которые

могут быть выполнены с самолетом, находящимся в перечисленных фазовых состояниях существенно различаются.

Кроме состояния другим важным понятием модели переходов состояний является *переход*. Главное различие между состоянием и переходом заключается в том, что длительность нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое. Предполагается, что в пределе время перехода из одного состояния в другое равно нулю, или, другими словами, переход объекта из состояния в состояние происходит мгновенно.

Главное предназначение модели переходов состояний – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Модель переходов состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий. Поведение системы с помощью модели переходов состояний описывается как последовательное перемещение по графу состояний: от вершины к вершине по связывающим их дугам с учетом их ориентации.

Вершинами графа модели являются состояния, а дуги графа служат для обозначения переходов из состояния в состояние. Модели переходов состояний могут быть вложены друг в друга, образуя вложенные диаграммы более детального представления отдельных элементов описания системы.

Удобным математическим описанием модели переходов состояний является конечный автомат. Конечные автоматы моделируют поведение, при котором реакции на будущие события зависят от предыдущих событий.

Конечный автомат — это математическая абстракция, позволяющая описывать траекторию изменения состояния объекта в зависимости от его текущего состояния и входных данных, при условии что общее возможное

количество состояний конечно. Существуют различные варианты задания конечного автомата. Например, конечный автомат может быть задан с помощью пятерки

$$\Sigma = <X, Y, Q, \mu, \lambda>,$$

где:

X —конечное множество допустимых входных символов автомата;

Y —конечное множество допустимых выходных символов автомата;

Q — конечное множество состояний автомата;

μ — функция переходов автомата, задающая отображение множества $Q \times X$ во множество Q

$$\mu: Q \times X \rightarrow Q;$$

λ — функция выхода автомата, задающая отображение множества $Q \times X$ во множество Y

$$\lambda : Q \times X \rightarrow Y.$$

Автомат начинает работу в состоянии q_0 считывая по одному символу входной строки. Считанный символ переводит автомат в новое состояние из Q в соответствии с функцией переходов. Если по завершении считывания цепочки входных символов автомат оказывается в одном из допустимых состояний, то эта цепочка обрабатывается автоматом. В противном случае — отвергается. Можно сказать, что конечный автомат представляет собой граф состояний и переходов, который описывает, каким образом экземпляр класса реагирует на получение событий.

Формализм автоматов допускает вложение одних автоматов в другие для уточнения внутренней структуры отдельных более общих состояний (макросостояний). Вложенные автоматы могут использоваться для внутренней спецификации процедур и функций, образующих поведение исходного объекта.

Автомат не должен содержать конфликтующих переходов, т. е. таких переходов из одного и того же состояния, когда объект одновременно может

перейти в два и более последующих состояния (кроме случая параллельных подавтоматов). В языке UML исключение конфликтов возможно на основе введения так называемых сторожевых условий.

Правила поведения объекта, моделируемого некоторым автоматом, определяются, с одной стороны, общим формализмом автомата, а с другой – его графическим изображением в языке UML в форме конкретной диаграммы переходов состояний.

Таким образом, конечный автомат модели переходов состояний состоит из:

- *Событий*, на которые он реагирует;
- *Состояний*, в которых автомат пребывает между событиями;
- *Переходов* между состояниями при реагировании на события;
- *Действий*, выполняемых в процессе переходов;
- *Переменных*, которые содержат значения, необходимые для выполнения действий.

Модель переходов состояний обеспечивает моделирование различных фазовых состояний, в которых может находиться объект. В то время как модель классов показывает статическую картину классов и их связей, модель переходов состояний применяется при описании динамики поведения объектов системы.

Так, класс *Документ* может иметь несколько различных фазовых состояний. Он может находиться в стадии подготовки, согласования, регистрации, исполнения или готовности. Поведение класса *Документ* меняется в зависимости от состояния, в котором он находится. На модели переходов состояний показывают именно эту информацию.

На рис. 7.4 приведен пример модели переходов состояний для класса *Документ*. Модель показывает возможные фазовые состояния класса

Документ, а также процесс перехода класса *Документ* из одного фазового состояния в другое. Например, если исполнитель подготовил документ, то последний переходит в состояние *Регистрируется*. Завершение подготовки документа исполнителем – это событие, вызывающее переход объекта *Документ* из одного фазового состояния в другое.

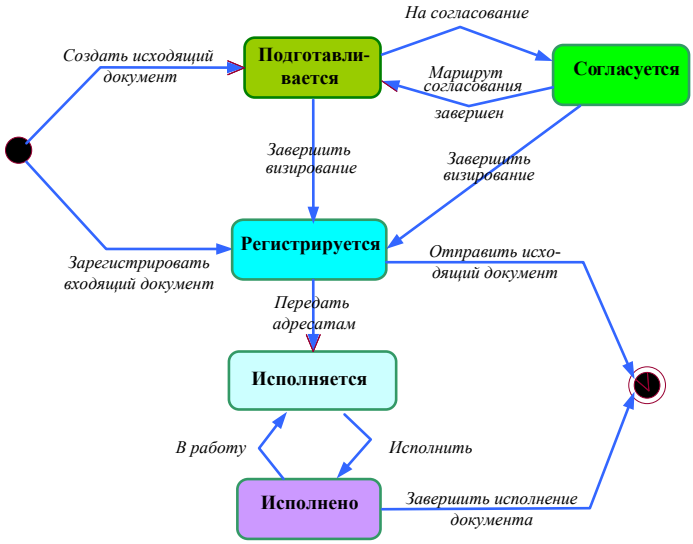


Рис. 7.4. Модель переходов фазовых состояний класса *Документ*

На диаграмме переходов состояний имеются два специальных состояния — начальное и конечное. Начальное состояние выделяется черной точкой, оно соответствует состоянию объекта в момент его создания. Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. В модели переходов состояний может быть одно и только одно начальное состояние. В то же время может быть столько конечных состояний, сколько необходимо, или их может не быть вообще.

Когда объект находится в каком-то конкретном фазовом состоянии, могут выполняться те или иные процессы. Например, при превышении кредита клиенту посылается соответствующее сообщение. Процессы, происходящие, когда объект находится в определенном фазовом состоянии, называются действиями или операциями. Модели переходов состояний нет

необходимости создавать для каждого класса, они применяются только для объектов, имеющих значительное число фазовых состояний. Если объект класса может существовать в нескольких состояниях и в каждом из них ведет себя по-разному, для него, вероятно, потребуется такая модель.

Семантика конечного автомата заключается в реакции экземпляра класса объектов на получаемые от внешнего мира события. Объект реагирует на события в соответствии с тем состоянием, в котором он находится в настоящий момент времени. Возможными реакциями могут быть выполнение какого либо действия или переход в новое состояние. При этом изменение состояния объекта может быть вызвано внешними воздействиями со стороны других объектов или извне. Кроме того, автомат может описывать поведение отдельного объекта в форме последовательности состояний, которые охватывают все этапы его жизненного цикла, начиная от создания объекта и заканчивая его уничтожением.

Хотя процесс изменения состояний модели происходит во времени, явно концепция времени не входит в формализм модели, предполагается, что последовательность изменения состояний упорядочена во времени. Другими словами, каждое последующее состояние всегда наступает позже предшествующего ему состояния. Это означает, что длительность нахождения модели в том или ином состоянии, а также время достижения того или иного состояния никак не специфицируются.

Параллельные состояния независимы друг от друга и действуют на различных множествах значений. Любые взаимодействия должны моделироваться явно, в виде отправки сигналов. В случае если два параллельных состояния должны иметь доступ к одному общему ресурсу, они должны явно отправить атому ресурсу сигнал, который обрабатывается в соответствии с очередностью поступления.

Семантика модели переходов состояний предполагает выполнение следующих обязательных условий:

- В каждый момент времени модель может находиться в одном или нескольких своих состояний. При этом модель может находиться в отдельном состоянии как угодно долго, если не происходит никаких событий. Если состояние активно, то может запуститься исходящий переход из этого состояния, что приведет к выполнению действия и активации другого состоянии или состояний. Если активных состояний больше одного, это указывает на существование внутреннего параллелизма. У параллельных состояний существуют ограничения, устанавливаемые структурой конечного автомата и его переходами.
- Модель обрабатывает по одному событию за один раз и перед обработкой следующего события завершает все, что касается предыдущего. Другими словами, при обработке событий они не взаимодействуют друг с другом и обрабатываются асинхронно. Два события никогда не происходят в один и тот же момент. Если объект получает событие во время выполнения действия, то это событие помещается в очередь до тех пор, пока это выполнение не закончится. Событие обрабатывается только в том случае, если в это время не осуществляется выполнение каких-либо действий. Если действие посылает сигнал другому объекту, то его будут обрабатывать так же, как и любое другое событие, то есть после завершения действия и перехода, который является его частью.
- Во время обработки события при запуске перехода происходит оценка предусловия. Если оно истинно, то переход действительно запускается. Предусловие может включать в себя аргументы переключающего события или атрибуты объекта. Предусловия не имеют побочных эффектов. Другими словами, они не могут

изменять состояние объекта или остальной системы. Следовательно, порядок их вычисления не влияет на результат. Предусловие вычисляется только во время обработки события. Если в этот момент предусловие ложно, то оно не проверяется повторно, даже если какая-либо переменная со временем сделает его истинным.

- Модель переходов состояний не запоминает историю перемещения из состояния в состояние. С точки зрения моделируемого поведения определяющим является сам факт нахождения объекта в том или ином состоянии, но никак не последовательность состояний, в результате которой объект перешел в текущее состояние. Другими словами, модель «забывает» все состояния, которые предшествовали текущему в данный момент времени.
- Граф модели переходов состояний не должен содержать изолированных состояний и переходов. Это условие означает, что для каждого из состояний, кроме начального, должно быть определено предшествующее состояние. Каждый переход должен обязательно соединять два состояния модели. Допускается переход из состояния в себя.

Модель переходов состояний наиболее полезна в ситуациях, в которых поведение управляется многими различными типами событий, а реакция на определенное событие зависит от последовательности предыдущих событий. События, которые управляют объектами системы, могут быть внешними по отношению к системе, или внутренними, исходящими от других компонентов системы.

Хотя модель переходов состояний чаще всего используется для описания поведения отдельных экземпляров классов (объектов), но она также может быть применена для спецификации функциональности других

динамических аспектов систем, таких как прецеденты, роли пользователей, подсистемы и действия.

7.6 Модель последовательности взаимодействия объектов

Различные составные элементы систем не существуют изолированно, а оказывают определенное влияние друг на друга, что и отличает систему как целостное образование от простой совокупности элементов. В языке UML взаимодействие элементов систем рассматривается в информационном аспекте их коммуникации, т. е. взаимодействующие объекты обмениваются между собой некоторой информацией. При этом информация принимает форму законченных сообщений. Другими словами, хотя сообщение и имеет информационное содержание, оно приобретает дополнительное свойство оказывать направленное влияние на своего получателя. Это полностью согласуется с принципами ООМ, когда любые виды информационного взаимодействия между элементами системы должны быть сведены к отправке и приему сообщений между ними.

Необходимо отметить, что хотя модели переходов состояния и деятельности и используются для спецификации динамики поведения систем, время в явном виде в них не присутствует. Однако временной аспект поведения может иметь существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов.

Для моделирования последовательности взаимодействия объектов в языке UML используются диаграммы взаимодействия, в которых рассматриваются временные особенности передачи и приема сообщений между объектами. На таких диаграммах показывается последовательность действий пользователя, объекты, которые участвуют в данном прецеденте, и их операции.

Модель последовательности взаимодействия объектов отражает поток событий, происходящих в рамках прецедента. Например, прецедент *Снять деньги* предусматривает несколько возможных последовательностей: *Снятие денег, Попытка снять деньги при отсутствии их достаточного количества на счету, Попытка снять деньги по неправильному идентификационному номеру* и некоторые другие. Нормальный сценарий снятия денег со счета (при отсутствии таких проблем, как неправильный идентификационный номер или недостаток денег на счету) показан на рис. 7.5

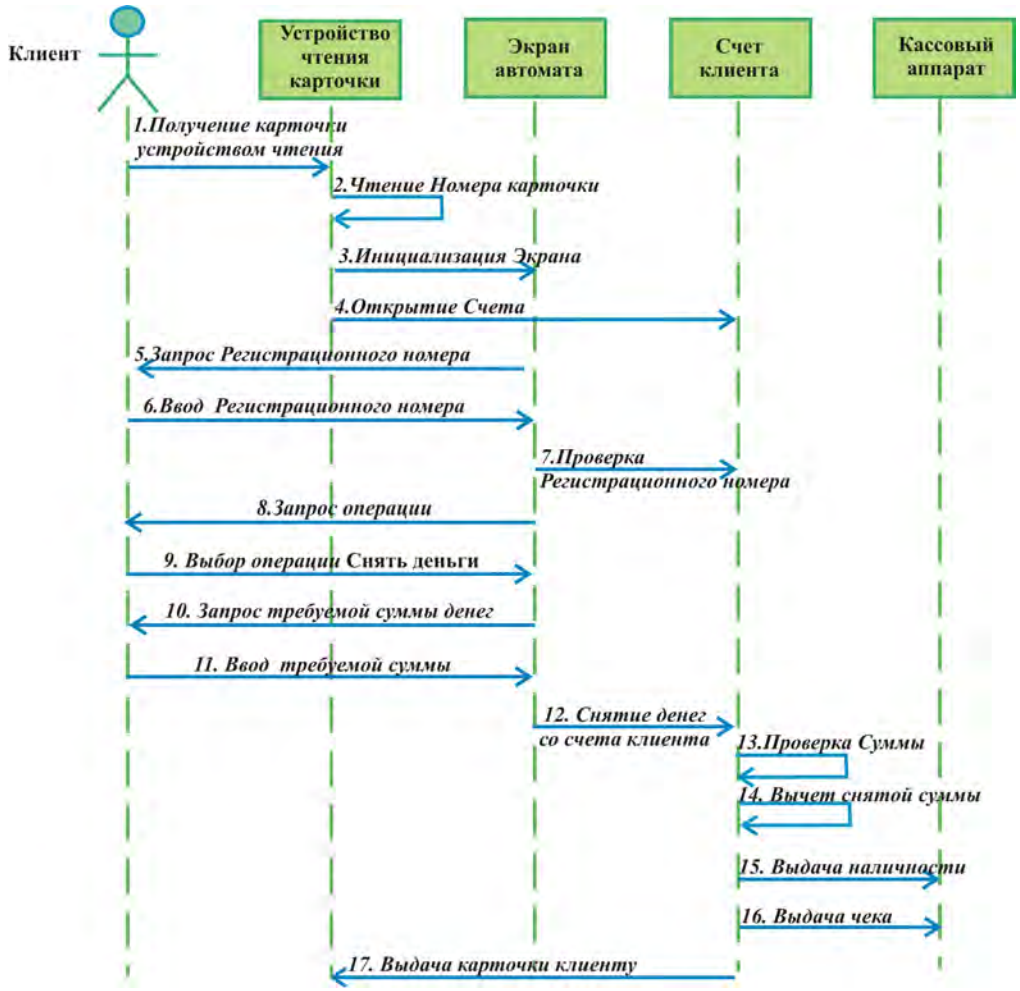


Рис.7.5 Пример модели последовательности взаимодействия для снятия клиентом денег со счета

Эта модель последовательности отображает поток событий в рамках прецедента *Снять деньги*. В верхней части модели показаны все действующие лица и объекты, требуемые системе для выполнения

прецедента *Снять деньги*. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций. Следует отметить также, что в модели последовательности взаимодействия показаны именно объекты, а не классы.

Прецедент начинается, когда *Клиент* вставляет свою карточку в устройство для чтения — этот объект показан в прямоугольнике в верхней части диаграммы. Он считывает номер карточки, открывает объект *Счет* и инициализирует экран банковского автомата. Автомат запрашивает у клиента его регистрационный номер. *Клиент* его вводит. Автомат проверяет номер у объекта *Счет* и обнаруживает, что он правильный. Затем автомат предоставляет *Клиенту* меню для выбора, и тот выбирает операцию **Снять деньги**. Автомат запрашивает требуемую сумму. *Клиент* указывает конкретное число. Автомат снимает деньги со счета. При этом он инициирует серию процессов, выполняемых объектом *Счет*.

Сначала осуществляется проверка, что на счету клиента имеется требуемая сумма. Затем из *Счета* вычитается требуемая сумма и *Кассовый аппарат* получает инструкцию выдать чек и тебуемую сумму наличными. Наконец все тот же объект *Счет* дает устройству для чтения карточек инструкцию вернуть карточку.

Литература

1. Алгебраическая теория автоматов, языков и полугрупп. Под ред. М. Арбиба. – М.: Статистика, 1975.- 336 с.
2. Агапов А. Еще раз о моделировании бизнес-процессов. Директор, №3, 2001.
3. Агафонов В. Н. Типы и абстракции данных в языках программирования (обзор).//Данные в языках программирования. Сб. статей. – М.: Мир,1982. – С. 265-327.
4. Агафонов В. Н. Спецификация программ: понятийные средства и их организация. Новосибирск: Наука, Сиб. отделение, 1987. – 240 с.
5. Акоф Р. С., Эмери Ф. И. О целеустремленных системах. – М.: Сов. Радио, 1974.- 272 с..
6. Бениаминов Е. М. Алгебраические методы в теории баз данных и представлении знаний. – М.: Научный мир, 2003. – 184 с.
7. Берталанфи Л. Общая теория систем - критический обзор // Исследования по общей теории систем. Сборник переводов с польского и английского. – М.: Прогресс, 1969. – С. 23-82.
8. Богданов А.А. Тектология. Всеобщая организационная наука. Книга 1. – М.: Экономика. 1989.
9. Букур И., Делану А. Введение в теорию категорий и функторов. – М.: Мир,1972. – 266 с.
10. Бусленко Н. П., Калашников В. В., Коваленко И. Н. Лекции по теории сложных систем. – М.: Сов. Радио, 1973. – 440 с.
11. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя. Унифицированный язык моделирования UML. – М.: ДМК, 2000 г. – 432 с.
12. Винер Н. Кибернетика. – М.: Наука, 1983.

13. Войшвили Е. К. Понятие как форма мышления. Логико гносеологический анализ. – М.: Изд. МГУ, 1989. – 240 с.
14. Г. А. Голицын, А. П. Левич. Принцип максимума информации и вариационные принципы в научном знании. http://www.chronos.msu.ru/RREPORTS/golitsyn_princip.htm
15. Гольденблат Р. Топосы: Категорный анализ логики. – М: Мир, 1983. – 486 с.
16. Горский Д. П. Обобщение и познание. – М.: Наука,1985. – 182 с.
17. Гренандер У. Лекции по теории образов. В 3 т. – М: Мир, 1979 - 1983.
18. Дейкстра Э. Дисциплина программирования. – М.: Мир, 1978. – 278 с.
19. Дружинин В.В., Конторов Д.С. Проблемы системологии. – М.: Сов. Радио, 1976. – 296 с.
20. Замулин А. В. Системы программирования баз данных и знаний. – Новосибирск: Наука, Сиб. отделение, 1990. – 352 с.
21. Калман Р., Фальб П., Арбиб М. Очерки по математической теории систем. – М.: Наука, 1986.
22. Калиниченко Л. А. Методы и средства интеграции неоднородных баз данных. – М.: Наука,1983. – 424 с.
23. Каплан Р., Нортон Д. Организация, ориентированная на стратегию. – М.: Олимпик-Бизнес, 2004.
24. Касти Дж. Большие системы. Связность, сложность и катастрофы. – М.: Мир,1982. – 216 с.
25. Ковальски Р. Логика в решении проблем. – М.: Наука,1990. – 290 с.
26. Кон П. Универсальная алгебра. – М.: Мир, 1968. – 352 с.
27. Конторов Д. С., Конторов М. Д., Слока В. К. Радиоинформатика. – М.: Радио и связь, 1993. – 296 с.
28. Левич А. П. Теория множеств, язык теории категорий и их применение в теоретической биологии. – М.: Изд. МГУ, 1978. – 190 с.
29. Левич А. П. Информация как структура систем. Семиотика и информатика. Вып. № 10. – М.: Изд. ВИНИТИ, 1978. – С. 116-131.

30. Левич А.П., Соловьев А.В. Категорно-функторное моделирование естественных систем // Анализ систем на пороге XXI века. М.: Интеллект. 1997. С.66-78.
31. Левич А.П. Принцип максимума энтропии и теоремы вариационного моделирования // Успехи современной биологии, 2004, т. 124, № 6, с. 3-21.
32. Левич А.П. Общая теория систем как метатеория теоретического научного знания и темпорологии // Пространство и время: физическое, психологическое, мифологическое. М.: КЦ "Новый Акрополь", 2006. С. 70-88.
33. Лисков Б., Гатег Дж. Использование абстракций и спецификаций при разработке программ. – М.: Мир,1989. – 424 с.
34. Леоненков А. В. Самоучитель UML. – М.: BHV, 2006. – 432 с.
35. Линдон Р. Заметки по логике. – М.: Мир, 1968. – 128 с.
36. Макетирование, проектирование и реализация диалоговых информационных систем/ Под ред. Ломако Е. И. – М.: Финансы и статистика, 1993. – 320 с.
37. Мальцев А. И. Алгебраические системы. – М.: Наука, 1970. – 302 с.
38. Месарович М., Такахара Я. Общая теория систем: математические основы. – М.: Мир, 1978. – 312 с.
39. Месарович М., Мако Д., Такахара И. Теория иерархических многоуровневых систем. – М.: Мир, 1973. – 344 с.
40. Плоткин Б. И. Универсальная алгебра, алгебраическая логика и базы данных. – М.: Наука, 1991. – 448 с.
41. Поспелов Д. А. Логико-лингвистические модели в системах управления. – М.: Энергоиздат, 1981. – 232 с.
42. Представление и использование знаний / Под ред. Уэно Х., Исидзука М. – М.: Мир,1989. – 220 с.

43. Смит Дж., Смит Д. Принципы концептуального проектирования баз данных//Требования и спецификации в разработке программ. Сб. статей под ред. Агафонова В. Н. –М.: Мир, 1984. – С. 165-198.
44. Рамбо Дж., Якобсон А., Буч Г. UML: Специальный справочник. – СПб.: Питер, 2002. – 656 с.
45. Рузинкевич М., Цикоцки А. Определение и выполнение потоков транзакций. Системы Управления Базами Данных # 2/95 стр. 106-115:
46. Требования и спецификации в разработке программ. Сб. статей под ред. Агафонова В. Н. –М.: Мир, 1984. – 320 с.
47. Цаленко М. Ш. Моделирование семантики в базах данных. – М.: Наука, 1989. – 288 с.
48. Цаленко М. Ш. Семантические и математические модели баз данных//Итоги науки и техники. Информатика. – Т. 9. – М.: ВИНИТИ, 1985. – 207 с.
49. Цикритзис Д., Лоховски Ф. Модели данных. – М.: Финансы и статистика, 1985. – 344 с.
50. Шемакин Ю.И. Системантика. М., 2006.
51. Шенфилд Дж. Математическая логика. – М.: Наука, 1975. – 528 с.
52. Шилейко А. В., Кочнев В. Ф., Химушкин Ф. Ф. Введение в информационную теорию систем. – М.: Радио и связь, 1985. – 280 с.
53. Шрейдер Ю. А. Семантические основы информатики. – М.: ИПКИР, 1974. – 80 с.
54. Шрейдер Ю. А. Типология как основа классификации//НТИ. – Сер. 2. – 1981. – № 11. – С. 1-5.],
55. Шрейдер Ю. А., Шаров А. А. Системы и модели. – М.: Радио и связь, 1982. – 152 с.
56. Флейшман Б. С. Основы системологии. – М.: Радио и связь, 1982. – 368 с.

57. Фурсова П.В., Левич А.П., Алексеев В.Л. Экстремальные принципы в математической биологии // Успехи современной биологии, 2003, том 123, № 2, с. 115-137.
58. Хоар И. Взаимодействующие последовательные процессы. – М.: Мир, 1989. — 264 с.
59. Эшби У. Р. Введение в кибернетику. – М.: ИЛ, 1959. – 432 с.
60. Янг С. Алгоритмические языки реального времени. Конструирование и разработка. — М.: Мир, 1985. — 400 с.
61. Alistair Cockburn. Writing Effective Use Cases. Addison-Wesley, 2000.
62. Business Process Modeling Notation Specification// http://www.omg.org/spec/BPMN/1.1/PDF
63. On Conceptual Modelling/ Ed by M. Brodie, J. Mylopoulos, J. Schmid. — New York: SRelinger Verlag, 1984. — 510 p.