ИНТУИТ
НАЦИОНАЛЬНЫЙ ОТКРЫТЫЙ УНИВЕРСИТЕТ

Национальный Открытый Университет "ИНТУИТ": www.intuit.ru
Екатерина Лаврищева, Владимир Петрухин
Лекция 5. Методы объектного анализа и построения моделей предметных областей

Главная цель объектного анализа – представить предметную область как множество объектов со свойствами и характеристиками, которые достаточны для их определения и идентификации, а также для задания поведения объектов в рамках выбранной системы понятий и абстракций. На произвольном шаге объектного анализа все понятия (сущности) ПрО – суть объекты. Каждый *объект* – это уникальный элемент, имеет, *по* крайней мере, одно свойство или характеристику и уникальную идентификацию во множестве объектов.

Предметная область сама является самостоятельным объектом или может быть объектом в составе другой *предметной области*.

Анализ ПрО проводится с помощью объектно-ориентированных методов и соответствующих стандартов. Конечная цель объектно-ориентированного анализа ПрО – *определение* объектной модели (ОМ) с помощью выделенных объектов, отношений между ними и их свойствами и характеристиками.

При построении модели ОМ в *предметной области* также выявляются функциональные задачи, формулируются требования к их проектированию и реализации. Требования, задачи и модель ОМ – необходимые условия построения архитектуры системы для анализируемой ПрО.

4.1. Краткий обзор объектно-ориентированных методов анализа и построения моделей

На данный момент известно более пятидесяти объектно-ориентированных методов анализа ПрО, которые прошли проверку практикой. Приведем некоторые основные из них:

- *метод объектно-ориентированного системного анализа OOAS* (Object-Oriented *system analysis*), позволяющий выделить сущности и объекты ПрО, определить их свойства и отношения, а также построить на их основе информационную модель, модель состояний объектов и процессов представления потоков данных (dataflow) [4.1];
- *метод объектно-ориентированного анализа OOA* (*Object-Oriented analysis*), обеспечивающий моделирование ОМ и формирование требований к ПрО с помощью понятия "*сущность-связь*" (entity-relationship ER), спецификацию потоков данных и соответствующих процессов [4.2];
- *метод SD* (*Structured Design*) структурного проектирования системы, данных и программ преобразования входных данных в выходные с помощью структурных карт Джексона [4.3–4.5];
- *методология объектно-ориентированного анализа и проектирования OOAD* (*Object-oriented analysis* and design), которая основывается на ER-моделировании сущностей и отношений в объектной модели ПрО, она обеспечивает определение системы и организацию данных с использованием структурных диаграмм, диаграмм "*сущность-связь*" и матрицы информационного управления [4.6, 4.7];
- *технология объектного моделирования OMT* (Object Modeling Technique) включает в себя процессы (анализа, проектирования и реализации), набор нотаций для задания четырех моделей (объектной, динамической, функциональной и взаимодействия) [4.8, 4.9];
- *объединенный метод UML*, включающий средства и понятия метода Г. Буча (объекты, классы, суперклассы), принципы наследования, полиморфизма и упрятывания информации об объектах, а также варианты использования метода Джекобсона для задания сценариев работы системы при выполнении задач ПрО и диаграммные средства взаимодействия объектов Румбауха [4.10, 4.11];
- *метод определения распределенных объектов на основе объектной модели CORBA* и набора сервисных системных компонентов общего пользования, обеспечивающих их функционирование в среде распределенных приложений [4.16];
- *метод генерации (generative) частей системы из семейства ПрО* с помощью готовых объектов, аспектов, компонентов, программ многоразового использования и приложений, а также модели характеристик, в которой представлены функциональные и нефункциональные требования к семейству систем [4.13].

Наиболее используемая *объектная модель ПрО* реализована в системе *CORBA*. Каждый *объект* модели инкапсулирует некоторую сущность ПрО и определяет один или несколько сервисов (методов) ее реализации. Объекту соответствует одна или несколько операций обращения к методам. Объекты группируются в типы, а их экземпляры – в подтипы/*супертипы*.

Они инкапсулируют методы реализации, которые невидимы во внешнем интерфейсе, т.е. ОМ не содержит информации о способах реализации типа, а только о наличии его реализации. Во внешнем интерфейсе содержатся *операции*, которые вызывают методы объектов для их выполнения. Специализация типа определяется постепенно на этапах стратегии, анализа, проектирования и реализации объекта.

Взаимодействие объектов осуществляет *брокер объектных запросов* и операций (Особенности системы *CORBA* изложены в "Интерфейсы, взаимодействие и изменение программ и данных").

Приведенная общая характеристика разновидностей объектно-ориентированных методов показывает, что они имеют много общих черт (например, ER-*моделирование*, Dataflow), а также свои специфические особенности. Каждый разработчик метода объектно-ориентированного анализа вводил необходимые новые понятия, которые зачастую семантически совпадали с аналогичными понятиями в других методах. Поэтому у авторов *UML* возникла идея объединить свои индивидуальные методы объектного анализа (Буча, Джекобсона и Рамбауха) для создания единого метода объектного моделирования *UML*.

4.1.1. Основные понятия методов объектного анализа ПрО

К основным понятиям методов объектного анализа ПрО отнесем следующие [4.12, 4.13].

Объект ПрО – это абстрактный образ с поведением, которое обусловлено его характеристиками и взаимоотношениями с другими объектами ПрО.

Согласно теории Фреге [4.14] спецификацию объекта можно трактовать как треугольник:

<имя объекта > <денотат > <концепт>,

где <имя объекта> – идентификатор, строка из литер и десятичных чисел; <денотат> – сущность реального мира ПрО, которую обозначает идентификатор; <концепт> – смысл (семантика) *денотата* в соответствии с интерпретацией сущности моделируемой ПрО.

Объект интерпретируется как понятийная структура, состоит из идентификатора, *денотата* – образа предмета и концепта, отображающего смысл этого *денотата*, исходя из цели объектного моделирования ПрО. Денотат можно идентифицировать различными символами алфавита. Одному объекту могут соответствовать несколько концептов в зависимости от избранного уровня абстракции. Объект определяется через его внешнее отличие от других объектов. Внутренняя особенность объекта (его структура, внутренние характеристики) не влияет на внешнее отличие и для объектного моделирования значения не имеет.

Сущность – это семантически важный объект или тип объекта, существующий реально в ПрО или является абстрактным понятием, информацию о котором необходимо знать и/или сохранять [4.12, 4.13]. Имя сущности должно быть уникальным и может представлять тип или класс объектов. Сущность может иметь синонимы, записываемые через знак "/" (например, аэропорт/аэродром).

Концепт – значение некоторой абстрактной сущности ПрО, обозначается уникальным именем или идентификатором. Группа подобных концептов – это родительский концепт, который заведомо определяется некоторым набором общих атрибутов. Концепт вместе со своими атрибутами представляется графически в ОМ или в текстовом виде.

Атрибут – это абстракция, которой владеют все абстрагированные концепты сущности. Каждый атрибут обозначается именем, уникальным в границах описания концепта. Множество объединенных в группу атрибутов обозначает идентификатор этой группы. Группа атрибутов может объединяться в класс и иметь идентификатор класса.

Отношение – это абстракция набора связей, которые имеют место между разными видами объектов ПрО, абстрагированных как концепты. Каждая связь имеет уникальный идентификатор. Отношения могут быть текстовыми или графическими. Для формализации отношений между концептами добавляются вспомогательные атрибуты и ссылки на идентификаторы этих отношений. Некоторые отношения образуются как следствие существования других отношений.

Класс – это множество объектов, обладающих одинаковыми свойствами, операциями, отношениями и семантикой. Любой объект – это экземпляр класса. Класс представляется различными способа-ми (например, списками объектов, операций, состояний). Измеряется класс количеством экземпляров, операций и т.п.

Предметная область – это то, что анализируется с целью выделения специфичного множества понятий (сущностей, объектов) и связей между ними. На множестве этих понятий определяются задачи в целях автоматизированного их решения. Пространство ПрО можно разделить на пространство задач (*problem space*) и пространство решений (solution space) [4.13]. *Пространство задач* – это сущности, концепты ПрО, а *пространство решений* – это множество программных реализаций задач, в том числе функциональные компоненты, которые обеспечивают решение задач и функций ПрО, представленных в этом пространстве.

Выделение сущностей ПрО проводится с учетом отличий, определяемых соответствующими понятийными структурами. Объект как абстракции реального мира и понятийная структура обладает поведением, обусловленным свойствами и отношениями данного объекта с другими объектами. Выделенные в ПрО объекты структурно упорядочиваются *теоретико-множественными операциями* (принадлежности, объединения, пересечения и др.).

Модель ПрО – это совокупность точных определений понятий, концептов, объектов и их характеристик, а также множества синонимов и классифицированных логических взаимосвязей между этими понятиями.

Концептуальная модель – это модель ПрО, она создается без ориентации на программные и технические средства выполнения задач ПрО в операционной среде.

Для объектов модели устанавливаются отношения или связи. Различаются статические (постоянные) связи, которые не изменяются или изменяются редко, и динамические связи, которые имеют определенные состояния и изменяются во время функционирования системы.

Связи между объектами могут быть следующие:

связь один к одному (1:1) существует тогда, когда один экземпляр объекта некоторого класса связан с единственным экземпляром другого класса, т.е. в связи принимают участие по одному экземпляру из классов;

связь один ко многим (1:N), существует тогда, когда один экземпляр объекта некоторого класса связан одновременно с одним или более экземплярами другого класса или того же самого класса;

связь многие ко многим (M:N) существует тогда, когда в связях принимают участие несколько экземпляров объектов двух классов, т.е. один или больше экземпляров другого класса связан с одним или более экземплярами первого класса.

Состояние связей между объектами с течением времени может эволюционировать, и они могут существенно влиять на ход решения задачи. Для таких случаев связи строится ассоциативный объект и определяется модель состояний этого объекта путем добавления атрибута, фиксирующего текущее состояние.

Среди действий, которые сопровождают переходы объектов в определенные состояния в модели состояний, должны быть операции создания нового экземпляра ассоциативного объекта, если новая пара экземпляров вступает в связь или его уничтожения в случае, если объект или связь перестают существовать.

Используя приведенные базовые понятия методов объектного анализа ПрО, далее излагаются: объектный метод анализа ПрО и построения моделей [4.1], визуальный метод моделирования – UML [4.17] и проектирование архитектуры системы на основе стандартов.

4.1.2. Объектный метод построения моделей ПрО

Наибольшее распространение среди методов анализа ПрО получил метод OOAS Шлеера и Меллора [4.1], предназначенный для отображения ПрО следующими моделями:

информационная модель системы;

модель состояний объектов в информационной модели системы, определяемая набором правил поведения, предписаний и физических законов;

модель процессов, которая отображает процессы и действия, совершающиеся в системе и обеспечивающие прохождение моделей состояний через жизненные циклы – получение, порождение и уничтожение событий в системе.

Согласно этого методу ПрО анализируется в три этапа: информационное моделирование, моделирование состояний, моделирование процессов. В результате их выполнения создается система в виде совокупности этих моделей. Информационная модель отображает ПрО как мир объектов с характеристиками и атрибутами.

При переходе от этого этапа к этапу моделирования состояний для объектов информационной модели определяются связи объектов и их поведение. Создается модель состояний, которая отображает динамику состояния объектов системы и их поведение. На третьем этапе определяются действия и процессы, которые порождают события. Действия имеют функциональную природу. Цель моделирования процессов состоит в том, чтобы расчленить процессы на действия, которые вместе взятые определяют функциональное содержание системы. Рассмотрим модели метода подробнее.

Под **информационной моделью** понимается совокупность объектов (сущностей) ПрО, их характеристик (атрибутов) и связей между ними. Она создается по принципу реляционной модели данных, т.е. представления данных в виде отношений между ними.

Анализ ПрО состоит в выявлении объектов, предоставлении им уникальных и значимых названий, соответствующих смысловым понятиям в этой предметной области. В качестве объектов могут выступать:

реальные предметы мира ПрО как абстракции фактически существующих физических объектов ПрО;

роли как абстракции целей или назначения человека, части организации;

взаимодействия – объекты, получаемые путем установления отношений между другими объектами или частями системы;

спецификации, используемые для представления правил, критериев и ограничений на применение объектов в системе.

Таким образом, элементами информационной модели могут быть объекты, их атрибуты и идентификаторы, а также связи между объектами.

Для объектов ПрО определяются их характерные признаки или свойства, называемые атрибутами. Каждый атрибут – это абстракция одной характеристики объекта, которая присуща всем представителям класса объектов. Для классов объектов выбираются уникальные имена, устанавливаются атрибуты и связи. Атрибут получает имя, уникальное в рамках класса. Различаются описательные, указывающие и вспомогательные атрибуты.

Описательный атрибут устанавливает реальную характеристику, которая может определяться одним из таких возможных способов:

- заданием числового диапазона;
- перечислением возможных значений, которые может принимать атрибут;
- ссылкой на документ, который определяет возможные значения;
- заданием правил генерации допустимых значений.

Указывающий атрибут задает форму, назначение, перечисление или ссылку.

Дополнительный атрибут задает дополнительные значения, которые может принимать атрибут объекта.

Идентификаторы объекта содержат один или несколько атрибутов, значения которых позволяют однозначно выделить экземпляр объекта в данном классе (например, табельный номер сотрудника, номер паспорта и др.).

Ссылка на некоторый атрибут может уточняться именем класса, задаваемым через точку, а атрибуты – отношениями, которые определяются по следующим правилам:

- каждый объект – экземпляр класса или более чем одного класса, обладает одним значением своего атрибута,
- идентификатор может составляться из нескольких имен атрибутов (через точку), первое имя относится к классу, остальные к имени объекта.

Связи объектов устанавливаются между объектами одного или другого класса и характеризуются количеством экземпляров объектов, которые одновременно могут принимать участие в этих связях.

В информационной модели связи между объектами изображаются стрелками, указывающими направление связи. Возле рамки объекта, принимающего участие в связи, на линии стрелки указывается роль, которую этот объект поддерживает в данной связи. Связь 1:1 обозначается двунаправленной стрелкой, имеющей по одному "наконечнику" с каждой стороны; связь 1:N представляется стрелкой, имеющей два "наконечника" со стороны объекта, который состоит в связи с несколькими объектами; и, наконец, по два "наконечника" с каждой стороны имеет стрелка, означающая связь N:M.

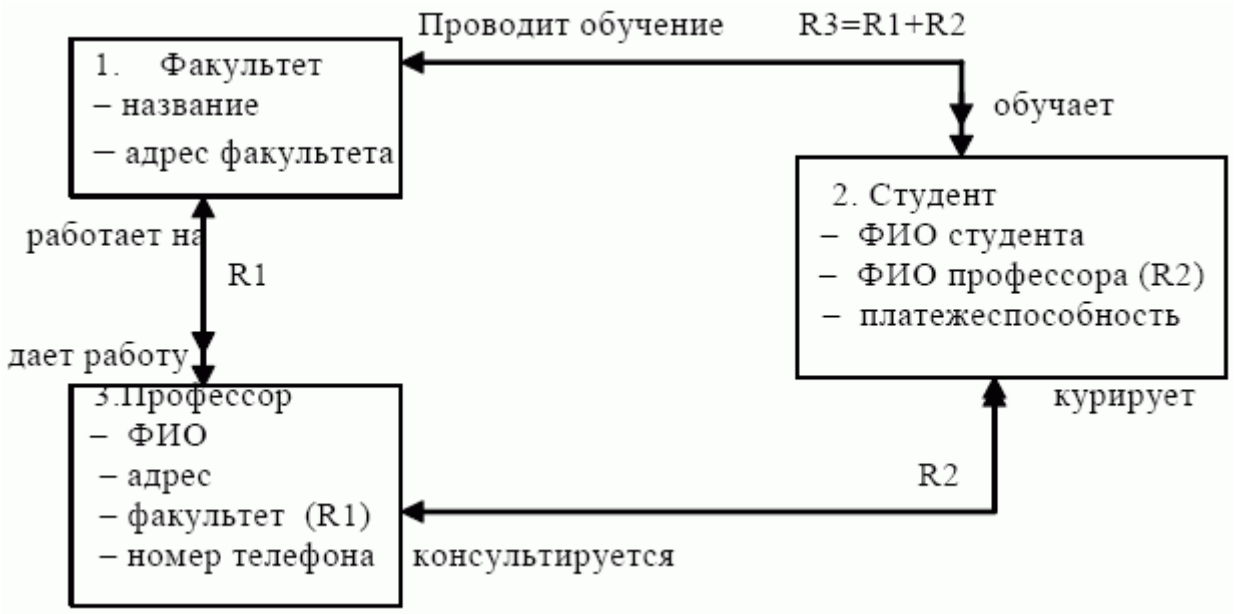

Рис. 4.1. Пример информационной модели.

Над стрелкой может указываться название связи. Связи могут быть безусловными, если каждый экземпляр объекта класса принимает участие в связи, и условными, когда отдельные экземпляры объектов класса не принимают участия в связи. Пример информационной модели с отображением связей приведен на рис. 4.1.

В этом рисунке, связь R3 – логическое следствие связей R1 и R2.

Построенная информационная модель сопровождается неформальным описанием всех объектов, их атрибутов и связей, в которых объекты принимают участие.

Модель состояний предназначена для отображения динамического поведения и изменения состояний каждого из объектов информационной модели и жизненного цикла поведения объектов. Состояние в модели – это положение или ситуация объекта, определяемая правилами и линией поведения. Событие – это инцидент, который заставляет объект переходить из одного состояния в другое. Экземпляры класса имеют поведение, которое определяется:

состоянием, зависящим от текущих значений отдельных его атрибутов;

состоянием, изменяемым в результате выполненных над объектами действий;

состоянием ПрО, зависящим от совокупности состояний ее объектов;

некоторыми процессами и действиями, которые изменяют жизненный цикл состояния объекта.

Построение модели состояний начинается после выделения в информационной модели отдельных объектов, обладающих динамическим поведением (например, изменение состояния с течением времени), создания экземпляра объекта или его уничтожения после прекращения существования (например, электрическая лампочка перегорает, тем самым закончился ее ЖЦ).

В данном методе предусмотрены две нотации для представления динамических аспектов поведения объектов: диаграмма перехода состояний и таблица перехода в состояния.

При построении модели состояний для каждого объекта информационной модели определяется следующее:

1. множество состояний, в которых объект может находиться;
2. множество инцидентов или событий, которые побуждают экземпляры класса изменять свое состояние;
3. правила перехода объекта из зафиксированного состояния на новое состояние при условии, что произойдет некоторое событие из множества событий;
4. действие на каждое из состояний выполняется при переходе в новое состояние.

Эта информация представляется в диаграмме перехода состояний (рис. 4.2) исходя из следующих условий:

каждое состояние, определенное для класса объектов, получает номер, уникальный идентификатор (ID) и название;

состояние обозначается рамкой, содержащей номер и название;

переход от состояния к состоянию изображается направленной дугой, помеченной меткой и названием события, обусловившего переход;

начальное состояние обозначается стрелкой, направленной к соответствующей рамке, и является состоянием, которое экземпляр объекта приобретает после своего создания;

заключительное состояние жизни экземпляра объекта (продолжение или разрушение) обозначается пунктирной рамкой;

указание на действия, которые должны быть выполнены экземпляром объекта для перехода в другое состояние.

Изменение состояния экземпляра класса объектов осуществляется при выполнении таких действий:

обработка информации, переданной в систему, что может повлиять на некоторое событие;

изменение поведения атрибута объекта;

вычисление атрибута;

генерация некоторой операции для одного из экземпляров класса объектов;

генерация события, сообщение о котором передается объекту, внешнему по отношению к данному;

прием сообщения о событии от внешних объектов;

взаимодействие с таймером, измеряющим время, истечение которого приводит к созданию некоторого события.

Изменение состояния экземпляра класса объектов осуществляется при выполнении таких действий:

обработка информации, переданной в систему, что может повлиять на некоторое событие;

изменение поведения атрибута объекта;

вычисление атрибута;

генерация некоторой операции для одного из экземпляров класса объектов;

генерация события, сообщение о котором передается объекту, внешнему по отношению к данному;

прием сообщения о событии от внешних объектов;

взаимодействие с таймером, измеряющим время, истечение которого приводит к созданию некоторого события.

Рис. 4.2. Модель состояний для обслуживания клиентов

Для отдельного экземпляра объекта может быть установлен таймер, который сообщит о наступлении события, соответствующего значению таймера (например, остановка работы прибора).

Альтернативой графической диаграммы перехода состояний является табличная нотация (табл. 4.1 для модели состояний на рис. 4.2).

В таблице каждое состояние представляется строкой, а каждое событие, воздействующее на объект – столбцом. Клетка таблицы перехода состояний – это состояние объекта, если соответствующее столбику событие произойдет, когда объект находился в состоянии, соответствующем строке. При этом допускается, что некоторые комбинации событие/состояние не приведут к изменению состояния экземпляра объекта, они содержат указание "событие игнорируется. При выборе формы представления – диаграмма или таблицы состояний перехода преимущество имеет диаграмма из-за наглядности и определенности действий, тогда как табличная форма служит для фиксации всех возможных комбинаций состояние/событие. Этим обеспечивается полнота и непротиворечивость заданных требований к системе.

Таблица 4.1. Таблица переходов состояний – обслуживание клиентов

	A1 –клиент ожидает	A2 – клиент ожидает	A3 – известен клиенту
1. Ожидание клиента	2	Событие игнорируется	Не может произойти
2. Ожидание свободного клерка	Событие игнорируется	3	Не может произойти
3. Определение клеерка клиентом	Событие игнорируется	Событие игнорируется	1

Важным принципом объединения объектов и компонентов в систему является наличие у них общих событий, причем чаще всего один из них порождает событие, а другие на него реагируют. На этом принципе базируется способ объединения отдельных объектов и компонентов в систему. Взаимодействие (внешнее и внутреннее) объектов рассматривается через обмен сообщениями для задания определенных событий и данных к ним. Внешний объект посылает сообщение, которое приводит к запуску системы и образованию внешнего события. Ему направляется сообщение о наступлении или отсутствии события.

Поведение отдельного объекта представляется в модели диаграммой перехода в состояния, а поведение системы – в виде схемы взаимодействия отдельных диаграмм, каждая из которых получает название в соответствующем овале (рис. 4.4). Овалы, отображающие отдельные диаграммы перехода состояний, связаны между собою стрелками, на которых задаются сообщения для возбуждения события. На стрелке указывается метка события (например, C1, C2, . . . , C8), а ее направление соответствует направлению передачи сообщения. Внешние объекты обозначаются прямоугольниками с названиями.

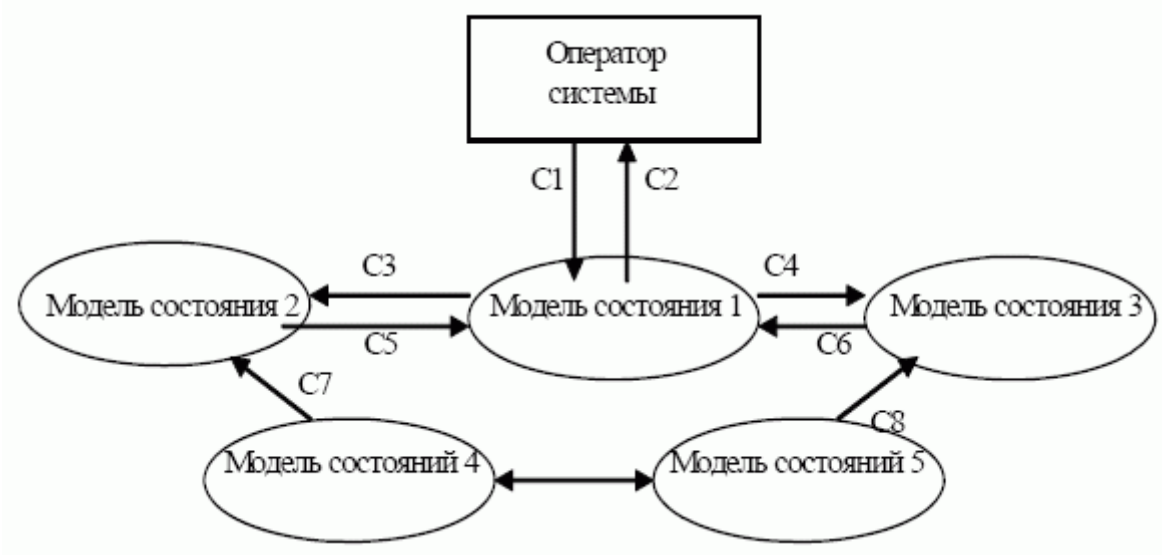

Рис. 4.4. Схема взаимодействия моделей поведения объектов

С помощью событий, указанных на стрелках данной схемы, инициируются модели состояний 1–5, каждая из которых посылает соответствующее сообщение.

Таким образом, модель состояний представляется диаграммами перехода в состояния, *таблицей состояний*, описанием действий и событий, изменяющих состояния объектов.

Модель процессов отражает изменения в моделях состояний. Каждое действие определяется в терминах процессов и архивов данных объектов. Т.е. процесс является фундаментальным модулем операции, а архив данных соответствует атрибутам объектов в информационной модели. Процессы действий имеют доступ к данным модели состояний, где они представлены. Для каждого действия модели состояний создается диаграмма процесса. Действия инициируют выполнение событий с помощью функций, которые реализуются в системе и отображаются в соответствующих диаграммах действий. В качестве источников данных для процессов могут выступать:

- атрибуты объектов, которые продолжают существовать после завершения работы системы;
- системные часы;
- таймер;
- данные происходящих событий;
- сообщения от внешних объектов.

Последовательность выполняемых процессов образует поток управления, а каждый процесс образует поток данных.

Для представления потоков данных используют диаграммы действий, правила построения таких диаграмм таковы:

- каждой из диаграмм перехода состояний может отвечать только одна диаграмма действий потоков данных;
- процесс изображается овалом с указанием содержания или названия процесса;
- потоки данных процессов изображаются стрелками, на которых указываются имена данных, передаваемых другому процессу;

стрелка к овалу процесса указывает на его входные данные, направление от овала – на выходные данные;

источники данных изображаются прямоугольниками;

данным, имеющим своим источником архивные объекты, соответствуют потоки с названиями атрибутов объектов, которые передаются потоками;

потоки данных отмечаются таймером или системными часами (час, минута);

событие, сообщение о котором получает процесс, изображается стрелкой с названиями данных.

В данном методе различаются следующие процессы общего назначения:

доступ к архивам;

подготовка и верификация объектов;

обработка потоков данных и генерация событий;

накопление объектов в архиве;

организация вычислений функций системы.

Потоки обозначаются пунктирными стрелками. Если процесс выполняет проверку определенного условия для передачи управления и входных данных другому процессу, то соответствующий поток изображается пунктирной линией с перечеркиванием. Фрагмент диаграммы действий процесса создания репозитария (типа библиотеки, архива) объектов приведен на рис. 4.5.

К диаграммам действий потоков данных добавляется неформальное описание функций процессов, которые входят в их состав. Для описания подробностей действий процессов нотация не регламентируется.

После завершения описания диаграммы действий потоков данных для всех объектов системы составляется общая *таблица процессов*, состоящая из следующих колонок:

идентификатор процесса;

тип процесса;

название процесса;

название состояния, для которого определен процесс;

название действия состояния.

Таблица дает возможность проверить:

непротиворечивость названий и идентификаторов процессов;

полноту определенных событий и соответствующих процессов;

генерацию события или его обработку соответствующим процессом.

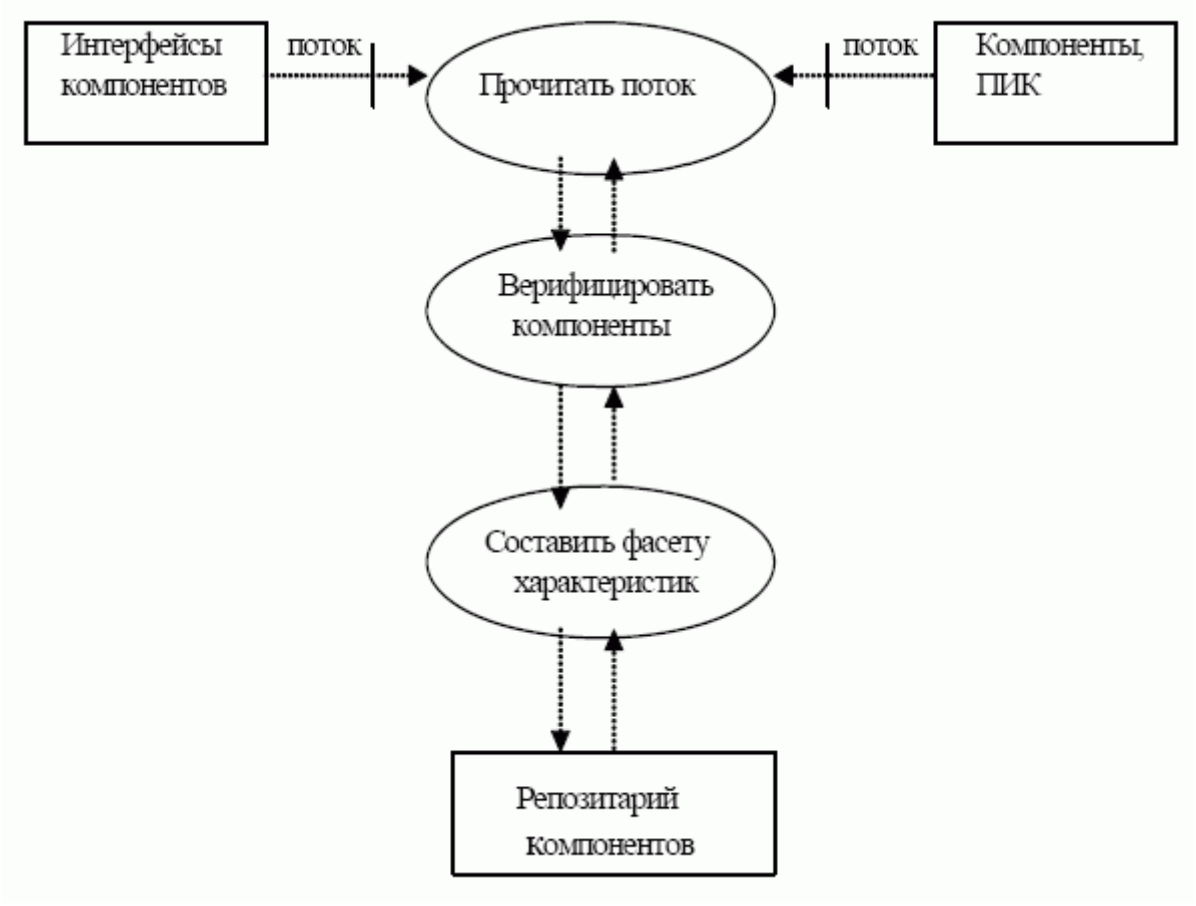

Рис. 4.5. Пример диаграммы действий процессов создания репозитария

К диаграммам действий потоков данных добавляется неформальное описание функций процессов, которые входят в их состав. Для описания подробностей действий процессов нотация не регламентируется.

После завершения описания диаграммы действий потоков данных для всех объектов системы составляется общая *таблица процессов*, состоящая из следующих колонок:

- идентификатор процесса;
- тип процесса;
- название процесса;
- название состояния, для которого определен процесс;
- название действия состояния.

Таблица дает возможность проверить:

- непротиворечивость названий и идентификаторов процессов;
- полноту определенных событий и соответствующих процессов;
- генерацию события или его обработку соответствующим процессом.

На данном этапе создается модель доступа к объектам, которая отображает взаимодействие объектов через модель состояний. Модель состояний обращается к данным экземпляра другого объекта во время выполнения действия. Этот вид взаимодействия считается *синхронным*. Если модель состояний получает событие после того, как действие завершилось, то это взаимодействие – *асинхронное*.

Результатом процесса моделирования процессов является: модель доступа к данным, диаграмма потоков данных действий, *таблица процессов*, содержащая процессы и действия над объектами ПрО, описание процессов путем их упорядочения по ID.

Определение архитектуры ПрО. После построения трех моделей, выполняется следующий этап метода – проектирование. На нем проводится разделение ПрО на подсистемы, определение их функций и принципов выполнения этих функций на процессах. В каждую подсистему включаются построенные модели или их фрагменты и соответственно выделенные объекты со всеми характеристиками. Между подсистемами устанавливаются соединительные связи, на которых указываются имена передаваемых данных или участвующих объектов. В результате создается графическое *представление архитектуры* системы. Преобразование результатов объектно-ориентированного анализа по данному осуществляется в языки C++, Ada и др.

Данный метод имеет много общего с методом Буча, реализован в ряде проектов (например, EaseCASE Plus 4.0). Применяется в различных областях (банковские операции, управление летательными аппаратами, оформление кредитных карт и др.) в США, Европе и Японии.

4.2. Методы проектирования архитектуры ПО

Проектирование ПО – это процесс разработки, следующий за этапом анализа и формирования требований. Задача проектирования – это преобразование требований к системе в требования к *ПО* и построение архитектуры системы [4.16].

Архитектура системы – это *структурная схема* компонентов системы, взаимодействующих между собой через интерфейсы. Компоненты могут составляться из последовательности более мелких компонентов и интерфейсов. Разработка архитектуры основывается на общем наборе справочников, классификаторов и т.п. В ней идентифицированы общие части, в том числе готовые программные продукты и вновь разработанные компоненты, а также многократно используемые в производстве других приложений.

Основное условие построения архитектуры системы – это *декомпозиция* системы на компоненты или модули, а также

- определение целей и проверка их выполнимости;
- определение входных и выходных данных;
- иерархическое представление абстракции системы и скрытие тех деталей, которые будут отработаны на следующих уровнях.

Основные решения *по* структуре системы принимаются группой архитекторов и аналитиков. Проект разбивается на *разделы* или отдельные части для их выполнения небольшими группами разработчиков, каждая из которых отвечает за одну или несколько частей системы.

Другой вариант определения архитектуры – это множество представлений, каждое из которых отражает некоторый аспект, интересующий группу участников проекта – аналитиков, проектировщиков, конечных пользователей и др. Представления фиксируют проектные решения *по* проектированию структуры и отражают аспект разделения приложения на отдельные компоненты и их связи. Эти решения проистекают из требований функциональности и влияют на проектные решения для нижних уровней структуры.

Проектирование архитектуры системы может проводиться структурным, объектно-ориентированным, компонентным и др. методами, каждый из которых предлагает свой *путь* построения архитектуры, включая концептуальную, объектную и др. модели и соответствующие им конструктивные элементы (блок-схемы, графы объектов и компонентов и др.).

При применении объектно-ориентированного подхода в качестве компонентов выступают отдельные объекты, а процесс конструирования объектной структуры превращается в процесс выявления имеющихся в ПрО объектов и определения их поведения и взаимодействия.

К методам проектирования относятся стандартный подход к проектированию, основанный на сформировавшейся общесистемной технологии традиционного проектирования программных систем.

4.2.1. Стандартный подход к проектированию

Разработка автоматизированных систем (АС) выполнялась на основе стандарта ГОСТ 34.601-90, регламентирующего стадии и этапы процесса разработки АС с учетом особенностей АС и средств объединения подсистем. Основание для разработки АС – это договор между разработчиком системы и заказчиком.

Этапами данного стандарта являются: формирование требований, разработка концепции системы, проектирование эскизного, технического и рабочего проекта. В эскизном и техническом проекте на основе сформулированных требований и концепций определяются конкретные задачи системы, строится структура системы, а также определяются пути и алгоритмы реализации подсистем. Эти этапы заканчивается созданием и утверждением отчета о научно-исследовательской работе, в котором дается оценка необходимых для реализации АС ресурсов, вариантов и порядка проведения оценки качества системы.

На этапе разработки *эскизного проекта* используются проектные решения ко всей системе или к ее части, определяется перечень задач, концепция информационной базы, функции и параметры основных компонентов системы, а также основные алгоритмы обработки информации.

Этап технического проектирования предусматривает разработку проектных решений относительно системы и ее частей, разработку документации, комплектацию АС, а также способов реализации технических требований на систему, алгоритмов задач, их распределения по смежным частям проекта и обмена данными между ними.

Проектные решения определяют организационную структуру, функции персонала АС, набор необходимых технических средств, *языка и системы программирования*, типы СУБД, систему классификации и кодирования, справочники, а также варианты ведения информационной базы системы.

Данный стандарт обеспечивает:

концептуальное проектирование,которое состоит в построении концептуальной модели, уточнении и согласовании требований;

архитектурное проектирование,которое состоит в определении главных структурных особенностей создаваемой системы;

техническое проектирование – это отображение требований определение задач и принципов их реализации в среде функционирования системы;

детальное рабочее проектирование,которое состоит в спецификации алгоритмов задач, построении БД и программного обеспечения системы.

Рассмотрим каждый вид проектирования более подробно.

При концептуальном проектировании определяются:

источники поступления данных от заказчика, который несет ответственность за их достоверность;

объекты системы и их атрибуты;

способы материализации связей между объектами и виды организации данных.

интерфейсы с потенциальными пользователями системы для оказания им помощи при формулировке целей и функций системы;

методы взаимодействия пользователей с системой для обеспечения скорости реакции системы.

Организация интерфейсов базируется на ключевых понятиях, связанных с конкретными экранами и форматами обмена данными, а также включает:

1. термины, образы и понятия, которые имеют значение для пользователя и домена;
2. модель организации, представления данных, функций и ролей, а также результаты их просмотра;
3. визуальные приемы отображения на экране элементов системы, наглядных для пользователя;
4. методы взаимодействия подсистем.

При архитектурном проектировании системы может применяться язык UML, который позволяет учитывать аспекты, свойственные действующим лицам, а также устанавливать форматы в меню и иконах интерфейсов [4.17].

Общая концепция объектного проектирования заключается в построении всех экранных форм и опробование их стиля на их разных вариантах. Выбор вариантов может вступить в противоречие с заданными характеристиками нефункциональных требований (например, обеспечение конфиденциальности, быстродействия и др.).

На основе модели *представления требований* и понятий компонентного или объектно-ориентированного проектирования проводится уточнение состава и содержания функций системы, методов их реализации и обеспечения их взаимодействия с помощью диаграмм потоков данных.

Взаимодействие объектов – это обмен сообщениями между элементами системы, подготовка ответа при выполнении операций, изменяющих свое состояние, и отправка ответа другим объектам.

Для уточнения поведения объектов используются диаграммы UML, отображающие различные аспекты взаимодействия объектов. Эти уточнения касаются интерфейсов и поведения объектов в сценариях, а также пересмотра моделей требований и состава объектов системы. Изменения начинаются с требований и поиска мест локации для внесения необходимых изменений в модель требований и их трассирование. Наряду с изменением требований к функциям системы могут изменяться нефункциональные требования, касающиеся ограничений на структуру системы и условий среды функционирования системы (отказоустойчивость и др.).

Модели требований для таких систем учитывают назначение и место требований в таких системах. Для этих целей разработаны национальные, корпоративные и ведомственные стандарты, которые фиксируют правила формирования нефункциональных требований, результатом которых могут быть сведения по обеспечению взаимодействия, защиты данных и др.

4.2.2. Общесистемный подход к проектированию архитектуры

Один из путей архитектурного проектирования – традиционный неформальный подход к определению архитектуры системы, составу ее компонентов, способов их представления и объединения во взаимосвязанную систему. Фактически создаваемая архитектура состоит из четырех уровней, которые включают:

1. Системные компоненты, устанавливающие интерфейс с оборудованием и аппаратурой;
2. Общесистемные компоненты, устанавливающие интерфейс с универсальными системами компьютеров;
3. Специфические бизнес-компоненты;
4. Прикладные программные системы.

1-й уровень – системные компоненты. Они осуществляют взаимодействие с периферийными устройствами компьютеров (принтеры, клавиатура, сканеры, манипуляторы и т.п.), используются при построении операционных систем.

2-й уровень – общесистемные компоненты. Они обеспечивают взаимодействие с универсальными сервисными системами среды работы прикладной системы, типа операционные системы, СУБД, системы баз знаний, системы управления сетями и т.п. Компоненты данного слоя используются во многих приложениях как необходимые составные компоненты.

3-й уровень – специфические компоненты определенной проблемной области, являются составляющими компонентами программных систем и предназначены для решения различных задач (например, бизнес-задач).

4-й уровень – прикладные программные системы, которые реализуют конкретные задачи отдельных групп потребителей информации из разных предметных областей (офисные системы, системы бухгалтерского учета и др.), могут использовать компоненты нижних уровней.

Компоненты любого из выделенных уровней используются, как правило, на своем уровне или более верхнем. Каждый уровень отражает соответствующий набор знаний, умений и навыков специалистов, создающих или использующих компоненты. Этот набор определяет соответствующее разделение специалистов программной инженерии (системщики, прикладники, программисты и др.).

При проектировании архитектуры программная система рассматривается как композиция компонент третьего уровня с доступом до компонентов первого и второго уровней. Т.е. архитектурное проектирование – это разработка компонентов третьего уровня, определение входных и выходных данных, слоев иерархии компонентов и их связей.

Результат проектирования – архитектура и инфраструктура, содержащая набор объектов, из которых можно формировать некоторый конкретный вид архитектурной схемы для конкретной среды выполнения системы. Заканчивается проектирование архитектуры системы описанием, в котором отображены зафиксированные проектные решения, логическая и физическая структура системы, а также способы взаимодействия объектов.

Объектный стиль проектирования заключается в декомпозиции проблемы на отдельные подсистемы (пакеты), определении функциональных и нефункциональных требований и модели предметной области. Определяются носители интересов (акторов), их возможные действия в пакете для получения результатов. Основу пакета составляет объектная модель, варианты использования, состав объектов и принципы их взаимодействия. Поведение объектов отражается диаграммами, которые задают последовательность взаимодействий объектов, правила перехода от состояния к состоянию (диаграммы состояний) и действия (диаграммы действий), а также поведение объектов кооперации (диаграммы кооперации). Объекты и соответствующие им диаграммы использования задают общую архитектурную схему системы, в рамках которой осуществляется реализация структуры и специфики поведения компонентов.

Архитектурная схема может быть: распределенная, клиент-серверная и многоуровневая.

Распределенная схема обеспечивает взаимодействие компонентов системы, расположенных на разных компьютерах через стандартные механизмы вызова RPC (Remote Procedure Calls), RMI (Remote Method Invocation), которые реализуются промежуточными средами (COM/DCOM, CORBA, Java и др.) [4.15, 4.16]. Взаимодействующие компоненты могут быть неоднородными, на разных языках программирования (C, C++, Паскаль, Java, Basic, Smalltalk и др.), которые допускаются в промежуточной среде системы CORBA (рис. 4.6). Для каждой пары языков взаимодействующих компонентов создаются интерфейсы типа L_i, L_n по количеству пар ЯП системы, допускающих взаимодействие между собой.

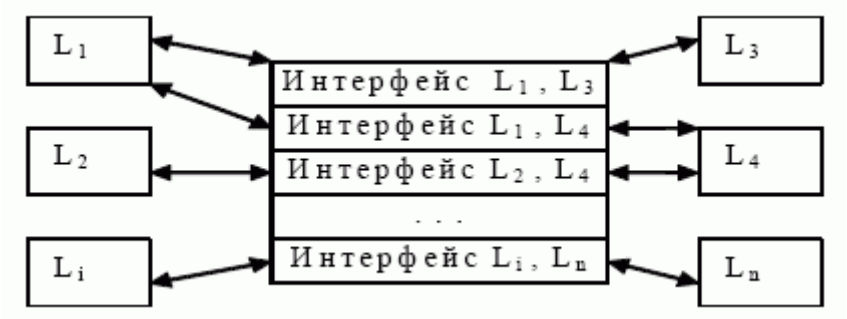

Рис. 4.6. Связь между языками L1, L2, …, Ln через интерфейсы

Схема клиент-сервер – *трехуровневая*.Главный вопрос этой схемы – доступ к ресурсам (аппаратуре, ПО и данным) и их разделение. При реализации архитектуры клиент-сервер сервер управляет ресурсами и предоставляет к ним доступ, а клиент их использует. Архитектура основана на распределенных объектах, которые инкапсулируют ресурс, и выдают услуги другим объектам.

Предоставляющие услуги объекты могут пользоваться тоже услугами других объектов. Функцию взаимодействия объектов выполняет *брокер объектных запросов* (*ORB*) через интерфейс клиент-сервер, он также предоставляет общесистемный сервис, услуги и различные ресурсы. Процесс разработки распределенных объектов начинается с формирования требований, проектирования объектов серверов, которые могут предоставлять услуги объектам клиента.

В качестве инструмента проектирования объектов применяется UML или унифицированный процесс RUP [4.17, 4.18]. Связи между объектами и их типами (операции и атрибуты) сервера и клиента задаются диаграммами классов. Взаимодействие объектов моделируется с помощью сценариев взаимодействия или *диаграмм последовательности*. Диаграммы состояний задают ограничения на операции к объектам сервера, преобразуются (генерируются) в интерфейсы (стабы), определяющие структуру и поведение объектов сервера (рис. 4.7).

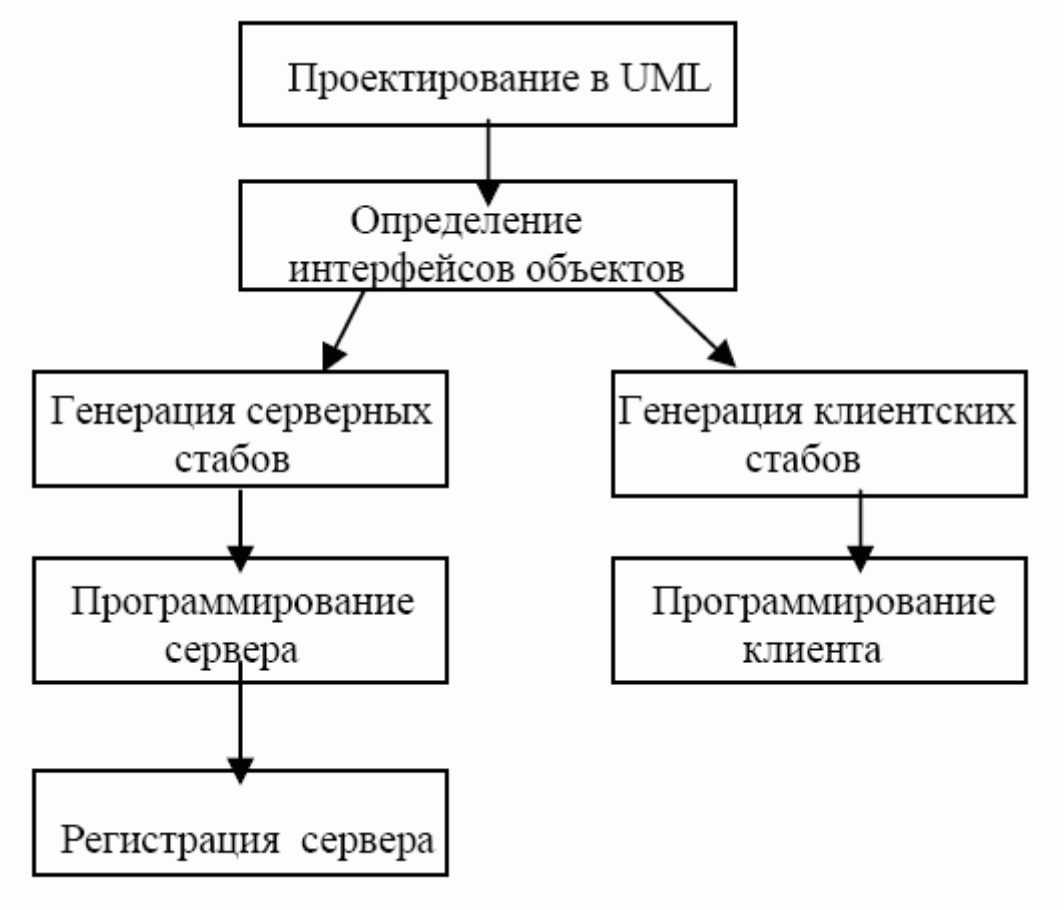

Рис. 4.7. Процесс разработки распределенных объектов

Стаб клиента используются в классах, экземплярами которых являются объекты клиента. При реализации объектов сервера используется стаб, тип которого наследуется от типа серверного стаба. Интерфейсы описываются в языке IDL и размещаются в промежуточном слое системы CORBA. Стабы предоставляют операции и соответствующие списки формальных параметров. При вызове клиент передает фактические параметры, которые соответствуют формальным параметрам. Объекты клиента и сервера – объекты стандартной модели архитектуры *OMA*.

Сущность стиля проектирования в рамках *унифицированного процесса* RUP состоит в предоставлении всех видов деятельности, выполняемых на моделях (анализа, проектирования, разработки и тестирования) процесса ЖЦ.

Модели охватывают все аспекты построения системы, структуру и поведение. В состав архитектуры системы входят модели процессов, содержащие статические и динамические объекты, их связи и интерфейсы между ними. В ней отображаются структура выделенных подсистем, справочников, словарей, а также результаты всех процессов.

Логическая структура проектируемой системы – это композиция объектов и готовых программных продуктов, выполняющих соответствующие функции системы. Композиция основывается на следующих положениях:

1. каждая подсистема должна отражать требования и способ их реализации (сценарий, прецедент, актер и т.п.);
2. изменяемые функции выделятся в подсистемы так, чтобы для них прогнозировались изменения требований и отдельные объекты, связанные с актером;

3. связь объектов осуществляется через интерфейс;
4. каждая подсистема должна выполнять минимум услуг или функций и иметь фиксированное множество параметров интерфейса.

Результаты архитектурного проектирования представляются нотациями в виде *диаграмм (сущность-связь*, переходы состояний, потоки данных и действий и т.п.) в *модели анализа* требований. Объекты диаграмм детализируют заданные функциональные требования к разработке системы и отображают процесс решения задач проекта.

Выделенные в *модели анализа* объекты объединяются в систему путем:

сборки объектов;

логического объединения объектов;

объединения по времени, т.е. сборка для заданного промежутка времени;

коммуникативного объединения объектов через общий источник данных;

процедурного объединения с помощью операторов вызова;

функционального объединения объектов;

Если во вновь создаваемой системе используется унаследованная система, то она снимает проблему дублирования и сокращает объем работ при проектировании архитектуры системы.

В сложных программных системах количество выделенных объектов может насчитывать сотни, их композиции не будут иметь выразительного представления, даже с учетом того, что объекты разных сценариев могут совпадать, поэтому в таком случае требуется дополнительный анализ для их отождествления.

Техническое проектирование состоит в отображении архитектуры системы в среду функционирования путем привязки элементов системы к особенностям платформы реализации: СУБД, ОС, оборудование и др. Перенос изготовленной ПС на другую платформу требует изменения параметров, настройки сервисов к новым условиям среды и адаптации используемых БД.

Для реализации таких свойств определяются объекты, которые взаимодействуют с сервисами системы, относительно которых декларируется переносимость. Любой определенный таким образом объект заменяется объектом, который не взаимодействует непосредственно с сервисом, а с некоторым абстрактным объектом-посредником, который осуществляет трансформацию абстрактного интерфейса в интерфейс конкретного сервиса системы. Объект-посредник при этом обладает свойством настраиваться на конкретную среду.

Вместе с тем любой аспект перехода на новую платформу может потребовать построения вспомогательных интерфейсных или управляющих объектов и корректировки существующих. Более того, может оказаться необходимость использования готовых подсистем, структура которых отличается от тех подсистем, которые были определены на основании анализа требований к системе. В этом случае вносятся соответствующие изменения в модель анализа требований и в архитектуру системы.

Контрольные вопросы и задания

1. Определите задачи анализа предметной области.
2. Сформулируйте задачи концептуального проектирования моделей ПрО.
3. Назовите продукты анализа домена в методе Шлеера и Меллора.
4. Назовите модели метода Шлеера и Меллора и их суть.
5. Какие еще модели ПрО существуют?
6. Перечислите ключевые факторы, влияющие на проектирование интерфейсов.
7. Назовите примеры нефункциональных требований, которые требуется учитывать на стадии проектирования архитектуры.
8. Какие уровни выделяются в архитектуре системы?
9. Какие известны способы объединения объектов в подсистемы?
10. Назовите приемы переноса системы в другую среду.