Построение концептуальной модели

При выполнении анализа прецедентов устанавливается картина взаимодействий в системе. Помимо этого, прецеденты помогают также определить словарь системы. Этот словарь обуславливается предметной областью и определяет основные объекты системы. В концептуальной модели перечисляются объекты, необходимые вам для адекватного описания системы.

Как стратегия борьбы со сложностью программных систем применяется декомпозиция системы на составляющие. В рамках ***структурного подхода*** к проектированию задача разбивается на *процессы или функции*, а при ***объектно-ориентированном*** ***подходе*** - на *понятия (концептуальные классы)*.

**Концептуальный класс** – это представление идеи или объекта из предметной области. Основной задачей на стадии анализа является идентификация различных понятий из предметной области и представление результатов в концептуальной модели.

**Концептуальная модель (модель предметной области)** – это визуальное представление концептуальных классов и отношений между ними. Необходимо заметить, что концептуальная модель не является описанием программных элементов, в ней нет элементов вроде окон, БД (если только мы не проектируем графический интерфейс или СУБД), нет обязанностей или методов.

Модель предметной области определяет наиболее важные типы объектов контекста системы. Объекты предметной области представляют собой «предметы», которые существуют, или события, которые происходят в той среде, в которой работает система.

Многие из объектов предметной области, или, используя более точную терминологию, классов предметной области можно определить из технического задания или в ходе опроса специалистов по проблемной области. Классы предметной области можно разбить на три типовых категории:

* бизнес-объекты, которые описывают сущности, используемые в бизнесе, такие как заявки, счета и контракты;
* объекты и понятия реального мира, которые система должна отслеживать, такие как вражеские самолеты, ракеты и траектории;
* события, которые произойдут или произошли, например прибытие самолета, отлет самолета или перерыв на обед.

Модель предметной области описывается диаграммами UML, главным образом диаграммами классов.

Эти диаграммы дают клиентам, пользователям, рецензентам и другим разработчикам информацию о классах предметной области и их ассоциативных связях.

Цель моделирования предметной области состоит в том, чтобы понять и описать наиболее важные классы контекста предметной области. Небольшие предметные области обычно содержат от 10 до 50 основных классов. В более обширной предметной области классов может быть гораздо больше.

Оставшиеся сотни кандидатов в классы, которые аналитики выявят внутри предметной области, сохраняются в глоссарии в виде определений понятий. В противном случае модель предметной области станет слишком большой и потребует значительно больших усилий, чем запланировано для этой стадии процесса.

Иногда, например, для особенно малых предметных областей, относящихся к бизнесу, нет необходимости создавать объектную модель предметной области. Вполне достаточно будет глоссария понятий.

Глоссарий и модель предметной области помогают пользователям, клиентам, разработчикам и другим заинтересованным лицам использовать общий словарь. Для того чтобы обмениваться знаниями, необходима общая терминология. Там, где процветает беспорядок, разрабатывать программы трудно, если не невозможно. Поэтому для построения программной системы любого размера современные инженеры должны проделать процедуру, обратную той, что произошла при постройке Вавилонской башни, — «объединить» языки всех участников разработки так, чтобы они понимали друг друга.

В заключение — предостережение, касающееся места моделирования предметной области в процессе моделирования. Очень легко начать моделирование внутренней структуры системы, а не ее контекста. Например, некоторые объекты предметной области могут быть непосредственно представлены в системе, и аналитики предметной области могут раз за разом попадать в ловушку описания деталей с учетом этого представления. В таких случаях очень важно иметь в виду, что цель моделирования предметной области состоит в том, чтобы разобраться в контексте системы и с помощью этого понять требования к системе — ведь они порождаются этим контекстом. Другими словами, моделирование предметной области должно способствовать пониманию задачи, которую система должна решать в контексте предметной области. С внутрисистемными методами решения этой задачи мы будем иметь дело в ходе рабочих процессов анализа, проектирования и реализации.

Как мы увидим, модель предметной области на самом деле является частным случаем более полной бизнес-модели. Таким образом, создание бизнес-модели — это серьезная альтернатива разработке модели предметной области.

На языке UML модель предметной области представляется в виде набора *диаграмм классов*, на которых не определены никакие операции. Модель предметной области может отображать следующее:

* Объекты предметной области (концептуальные классы);
* Ассоциации между концептуальными классами;
* Атрибуты концептуальных классов.

Идентификация понятий

При идентификации понятий следует иметь в виду то, что ***лучше излишне детализировать концептуальную модель, чем недоопределить её***. Не стоит исключать из словаря понятия лишь на том основании, что из анализа требований не следует необходимость его запоминания или понятие не имеет атрибутов, или понятие играет “поведенческую” роль.

К определению понятий-кандидатов есть несколько подходов, приведем два из них:

1. *Поиск понятий по списку категорий.*

При таком подходе все вещи, обладающие каким-то свойством или группой свойств, образуют одну категорию. Мы относим какое-либо понятие к категории на основе наличия или отсутствия признака образующего категорию.

Таблица 6 – Список категорий концептуальных классов

|  |  |
| --- | --- |
| **Категория** | **Примеры**  **(из области торговли, резервирования авиабилетов).** |
| Физические или материальные объекты | POST, Airplane (Самолет) |
| Спецификации, элементы дизайна или описания объектов | ProductSpecification,  Flight Description (описание полета) |
| Места | Store (магазин), Airport |
| Транзакции | Sale (продажа), Payment (платеж), Reservation (Резервирование) |
| Элементы транзакций | SalesLineItem (элемент продажи) |
| Роли людей | Cashier (кассир), Pilot |
| Контейнеры других объектов | Store (магазин), Bin (бункер), Airplane |
| Содержимое контейнеров | Item (товар), Passenger (пассажир) |
| Другие компьютеры или электромеханические системы, внешние по отношению к данной системе | CreditCardAuthorisation System  AirTrafficControl (Система управления движением) |
| Абстрактные понятия | Hunger (голод) |
| Организации | SalesDepartment, ObjectAirline (авиалиния) |
| События | Sale (Продажа), Robberty (кража), Meeting (встреча), Flight (Полет), Crash (крушение), Landing (приземление) |
| Правила и политика | RefundPolicy (правила возврата товара) |
| Каталоги | ProductCatalog (Каталог товаров) |
| Записи финансовой, трудовой, юридической и другой деятельности | Receipt (Чек), Ledger (Гросс бух), EmploymentContract (Трудовой контракт) |
| Финансовые инструменты и службы | LineOfCredit (Кредитная линия), Stock (акция) |
| Руководства, книги | EmployeeManual (Должностная инструкция) |

1. *Выделение понятий из текстовых описаний.*

Этот метод заключается в простом выделении существительных из текстовых описаний предметной области и выборе их в качестве кандидатов-понятий или атрибутов.

Применять этот метод надо очень аккуратно, поскольку между существительными и понятиями нет взаимно-однозначного соответствия и слова естественного языка могут иметь синонимы и омонимы. Тем не менее, это информация к размышлению. При применении этого метода удобно пользоваться развернутыми описаниями прецедентов (пример: Покупка товара).

|  |  |
| --- | --- |
| **Действия исполнителя** | **Отклик системы** |
| 1. ----//---- |  |
| 1. Кассир вводит универсальный код для каждого товара. Если выбрано несколько единиц одного и того же товара, кассир может ввести количество единиц. | 1. Определяет цену товара и добавляет информацию о товаре для выполнения транзакции. Выводятся описание товара и его цена. |

На данной итерации необходимо включать в концептуальную модель только те понятия, которые имеют отношение к реализуемому прецеденту (в нашем случае, *Покупка товара*).

Пользуясь описанными выше двумя методами, определим список понятий кандидатов для системы розничной торговли для первого цикла разработки. На этой итерации мы реализуем прецедент *Покупка товара*, поэтому и список кандидатов должен соответствовать требованиям и упрощениям для этого прецедента.

Вот эти понятия-кандидаты: *Товар, Магазин, Продажа, Платеж, Каталог\_продуктов, Описание\_продукта, Элемент\_продажи, Кассир, Покупатель, Менеджер*.

Типичной ошибкой при создании КМ является отнесение некоторого объекта к **атрибутам**, в то время как его необходимо отнести к **понятиям**. Следует придерживаться такого правила:

*Если некоторый объект в реальном мире не является числом или текстом, это, скорее всего, концептуальный класс, а не атрибут.*

**Пример.** *Резервирование авиабилетов*.

Для понятия Flight (Полёт) место назначения надо рассматривать не как атрибут destination, а как отдельное понятие Airport.

Практически невозможно обойтись без понятий, представляющих собой *спецификации* или *описания* других понятий. Удобно обозначать: «*XSpecification* *описывает X»*. Спецификации идентифицируются одними из первых, поскольку являются очень типичными.

Концептуальный класс спецификации или описания вводится в следующих случаях:

* Существует необходимость описания элементов или служб независимо от существования экземпляров этих объектов;
* Удаление экземпляров описываемых классов приводит к потере важной информации;
* Если при наличии понятия описания устраняется дублирование информации.

Добавление ассоциаций

**Ассоциация** – это связь между понятиями, отражающее некоторое значимое и полезное соотношение между ними.

Ассоциация обозначается проведенной между понятиями линий, с которой связано определённое имя. Обычно ассоциация является двунаправленной, то есть от одного объекта возможен логический переход к другому. Такой переход не определяет взаимосвязей между программными сущностями. На концах линий могут содержаться выражения, определяющую количественную связь между экземплярами понятий.

Стрелка направления чтения. Она не несёт смысловой нагрузки, лишь указывает направление чтения. Часто опускается. Если ее нет, то читать надо слева направо.

*Имя ассоциации*

*Кратность*

Записывает текущую

**Кассир**

**Продажа**

1

1

Каждый конец ассоциации называется **ролью.** Роль может иметь ***имя***, оно может быть подписано к концу ассоциации.

Works-for

**Сотрудник**

**Компания**

employee

***Кратность******роли*** определяет, сколько экземпляров типа **А** может быть ассоциировано с одним экземпляром типа **В** в определённый момент.

1

\*

**Магазин**

**Товар**

Хранит

Кратность

Следует заметить, что на стадии анализа ассоциация не является описанием потоков данных, экземпляров переменных или взаимодействия объектов программной системы. Она представляет собой лишь описание значимого отношения, имеющего чисто аналитический смысл, взятый из реального мира. Может быть так, что определенные на стадии анализа ассоциации не понадобятся на стадии проектирования, а может быть, потом будут добавлены еще какие-то ассоциации.

Рекомендации по поиску ассоциаций:

1. Необходимо уделить внимание тем ассоциациям, для которых данные о связи должны сохраняться в течение некоторого периода времени (***важным*** ***ассоциациям***).
2. Списком большое количество ассоциаций приводит к ошибкам в концептуальной модели. Как следствие, необходимо избегать избыточных ассоциаций (характерно для 3х и более объектов, участвующих в одном процессе).
3. Идентификация ассоциаций не должна занимать много времени, гораздо важнее все-таки выявить концептуальные классы.

Некоторые важные ассоциации можно выявить, используя *список стандартных ассоциаций:*

Категории ассоциаций, помеченные \*, являются ***категориями ассоциаций с высоким приоритетом***; ассоциации из этих категории очень полезно включать в концептуальную модель.

Ассоциация зависит от требований. Любое изменение требований влечет их добавление или удаление в КМ.

При использовании только *важных* ассоциаций будет сгенерирована минимальная “информационная модель'', содержащая лишь те знания предметной области, которые определяются принятыми текущими требованиями. При таком подходе нельзя получить полное представление о предметной области (например, другому человеку).

Часто КМ используется как средство, с помощью которого делается попытка разобраться в предметной области. В таком случае удаление некоторых не очень важных ассоциаций может привести к созданию модели, которая не отражает ключевых идей и связей.

Например, ассоциация *Покупатель инициирует Продажу*. Она не является необходимой, но ее отсутствие оставит за пределами рассмотрения важный аспект предметной области. Таким образом, можно сформулировать следующий основной критерий оценки модели: ***«Позволяют ли все важные и второстепенные ассоциации разобраться в понятиях предметной области и их отношениях?»***

Таблица 7 - Список стандартных ассоциаций

|  |  |
| --- | --- |
| **Категория** | **Примеры** |
| \*А является физической частью В |  |
| \*А является логической частью В | SalesLineItem – Sale (Элемент продажи - Продажа) |
| \*А физически содержится в/на В | Item – Self (товар – полка) |
| \*А логически содержится в В | ItemDescription - Catalog (Описание\_товара – Каталог\_товаров) |
| А является описанием В | ItemDescription – Item |
| А является элементом транзакции или отчета В | SalesLineItem – Sale |
| А известен/зарегистрирован/записан/включен в отчет/содержится в В | Sale – Store (Продажа – Магазин) |
| А является организационной единицей В | Department – Store |
| А использует или управляет В |  |
| А взаимодействует с В | Customer - Cahier |
| А связан транзакцией с В | Customer - Payment |
| А является транзакцией, которая связана с другой транзакцией В | Payment - Sale |
| А является собственностью В | Store - Item |

В примере **Internet-магазин** при анализе прецедентов можно выделить основные объекты, часть из которых представлена на рисунке 1.

Создание концептуальной модели важно по нескольким причинам. Во-первых, концептуальная модель описывает стоящую перед вами задачу независимо от конкретных способов ее реализации. Даже наоборот: модель описывает систему на абстрактном уровне. Подобная независимость дает возможность использовать созданную концептуальную модель для решения многих других задач в той же предметной области.

Во-вторых, концептуальная модель закладывает основу для создания объектной модели, которая потом превратится в готовую систему. В конечной реализации могут появиться некоторые новые классы или исчезнуть введенные вначале, но концептуальная модель дает вам скелет, основу для создания системы.



Рисунок 1

И, наконец, хорошо построенная концептуальная модель дает готовый словарь для вашей задачи, использование которого помогает найти общий язык всем сотрудникам, вовлеченным в реализацию проекта, и тем самым облегчает понимание задачи.

Добавление атрибутов

**Атрибут**- это абстрактное свойство объекта.

В КМ включаются только те атрибуты, для которых определены соответствующие требования (например, прецеденты), или необходимо хранить определенную информацию.

|  |
| --- |
| **Продажа** |
| date  time |

Атрибуты помещаются во второй раздел условного обозначения класса UML. Можно указывать также тип атрибута.

Тип атрибута не должен быть сложным понятием предметной области (Sale, Airport). Как правило, это примитивный тип данных. К стандартным типам атрибутов можно отнести Boolean, Date, Number, String (Text), Time Social, Address, Color, Geometrics (Point, Rectangle), Phone Number, Security Number (номер страхового полиса), Universal Product Code (UPC), ZIP или Postal Code (почтовый индекс), перечисляемые типы.

Требование того, что атрибуты предметной области должны описываться только простыми типами данных, не предполагает того, что на языках программирования свойства объекта должны иметь только примитивные, простые типы. Ассоциации между концептуальными классами предметной области зачастую реализуются как свойства, указывающие на другие сложные программные объекты. Однако это далеко не единственная возможность реализации ассоциации, и соответствующее решение нужно отложить на более поздний срок.

После объединения концептуальных классов, ассоциаций и атрибутов, будет создана модель предметной области рассматриваемой задачи. Не существует такого понятия, как «единственно верная модель». Модели, созданные различными разработчиками, наверняка будут различаться. Каждая модель является полезной в большей или меньшей степени, в зависимости от того, насколько точно она позволяет выполнить требования, предъявляемые к системе.

Ключевые понятия

Предположим, вы описали прецеденты, создали диаграммы взаимодействия и даже начали построение концептуальной модели. Что делать дальше?

Прецеденты имеют три основных предназначения. Первое связано с функциями, которые они выполняют. Прецеденты говорят, как должна функционировать система: кто будет ее использовать, что будет делать с ней пользователь и какой результат он ожидает получить Анализ прецедентов позволяет досконально понять создаваемую систему.

Во-вторых список прецедентов дает в то же время список задач, выполняемых в ходе создания системы. Вы можете начать с установления приоритетов прецедентов и оценки времени, необходимого для реализации каждого из них. Затем можно спроектировать временило последовательность разработки прецедентов. Завершение реализации прецедента можно сделать вехой в работе над системой. А иногда временные ограничения могут вынудить вас форсировать разработку одних прецедентов в ущерб другим.

И, наконец, прецеденты помогут вам создать концептуальную модель, которая станет основой вашей системы. (И если вы все сделаете правильно, то эту модель сможете использовать еще не раз!)

Когда вы чувствуете, что построение концептуальной модели и анализ прецедентов близок к завершению, можете начинать моделировать различные составляющие программы. Однако не моделируйте их произвольно. Прежде всего, следует приступить к тем частям программы, которые кажутся неоднозначными или рискованными. Моделирование может помочь вам глубже разобраться в задаче и, кроме того, подсказать, где скрыты опасности и как их устранить.

*Несколько советов по эффективному объектно-ориентированному анализу.*

1. Избегайте остановок в процессе анализа. Такие остановки могут возникнуть при попытке провести полный анализ сразу. При этом вы не сдвинетесь с места из-за того, что вам необходимо полное понимание задачи, а очень часто такое понимание приходит только после начала проектирования или даже реализации.
2. Делайте итерации. Везде. Прежде всего создайте предварительный список прецедентов. Определите приоритеты их реализации и на каждой итерации последовательно, в соответствии с приоритетами, выполняйте анализ прецедентов. Каждая итерация должна содержать некоторое количество анализа, проектирования и реализации. Время, посвящаемое реализации, будет увеличиваться в процессе работы над проектом: в начале потребуется совсем немного времени, но чем дальше, — тем больше.
3. Привлеките к анализу экспертов по предметной области, в которой вам придется работать, даже если единственным доступным экспертом окажется сам заказчик. Пока вы сами не станете экспертом в этой предметной области, вам будет необходима помощь извне.
4. Сделайте ваш анализ независимым от конкретной реализации. Не позволяйте реализации влиять на ход анализа.