



часть третья

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ




UML

Унифицированный язык моделирования (UML) — это язык визуального моделирования для решения задач общего характера, который используется при определении, визуализации, конструировании и документировании программной системы.



UML


С помощью языка UML можно фиксировать решения, принятые при создании различных систем.





UML


UML можно использовать со всеми методами разработки, во всех предметных областях и на всех этапах жизненного цикла программы.





UML


UML состоит из четырех частей,
описывающих различные аспекты
системы:

- статические,
 - динамические,
 - организационные,
 - относящиеся к окружению.
- 



UML


Спецификация UML не определяет конкретный процесс разработки, однако использовать этот язык моделирования удобнее всего в итеративном процессе. Также его можно применять в большинстве существующих объектно-ориентированных процессов разработки.





UML


UML позволяет отображать и статическую структуру, и динамическое поведение системы. Система моделируется как группа дискретных объектов, которые взаимодействуют друг с другом таким образом, чтобы удовлетворить требования пользователя.





UML

Наиболее полного и разностороннего понимания системы можно достичь при моделировании с различных, но взаимосвязанных точек зрения.



UML

UML не является языком программирования.


С его помощью можно писать программы, но в нем отсутствуют свойственные большинству языков программирования синтаксические и семантические соглашения.

С помощью инструментальных средств, поддерживающих UML и содержащих генераторы кода, на основе созданной модели можно получить программный код на различных языках, и наоборот, по исходному коду уже существующих программ можно восстановить их UML-модели.



UML


UML является языком моделирования
общего назначения.





UML


UML — язык дискретного моделирования.
Он не предназначен для разработки
непрерывных систем, встречающихся в
физике и механике.





Способы применения UML


Существуют три режима использования UML разработчиками:

- режим эскиза,
 - режим проектирования,
 - режим языка программирования.
- 



Способы применения UML


В режиме *эскизирования* разработчики используют UML для обмена информацией о различных аспектах системы.





Способы применения UML


Сущность эскизирования, или эскизного моделирования, в избирательности.





Способы применения UML

Эскизы полезны также и в документации, при этом главную роль играет процесс передачи информации, а не полнота.




Способы применения UML

Язык *UML* как средство проектирования нацелен на полноту. В процессе прямой разработки идея состоит в том, что проект разрабатывается дизайнером, чья работа заключается в построении детальной модели для программиста, который будет выполнять кодирование. Такая модель должна быть достаточно полной в части заложенных проектных решений.



Способы применения UML


При разработке моделей требуется более сложный инструментарий, чем при составлении эскизов, так как необходимо поддерживать детальность, соответствующую требованиям поставленной задачи.





Способы применения UML


Граница между моделями и эскизами довольно размыта, но различия остаются в том, что эскизы сознательно выполняются неполными, подчеркивая важную информацию, в то время как модели нацелены на полноту, часто имея целью свести программирование к простым и до некоторой степени механическим действиям.





Способы применения UML

При использовании *UML* в качестве языка программирования разработчики рисуют диаграммы, которые компилируются прямо в исполняемый код, а UML становится исходным кодом.



Концептуальные области UML

Все концепции и модели языка UML можно отнести к четырем концептуальным областям:

- Статическая структура
- Элементы проектирования
- Элементы развертывания
- Динамическое поведение




Статическая структура

В первую очередь, любая точная модель должна определять полное множество объектов, то есть ключевые концепции приложения, их внутренние свойства и отношения между собой. Эта структура и есть статическое представление системы.



Статическая структура


Концепции приложения моделируются как классы, каждый из которых описывает тип дискретных объектов, содержащих определенную информацию и взаимодействующих между собой для реализации некоторого поведения. Информация, которую содержат объекты, моделируется как атрибуты, поведение — как операции.





Статическая структура


В процессе работы одни объекты имеют связи с другими. Такие отношения между объектами моделируются как ассоциации между соответствующими классами.





Статическая структура


Классы могут иметь интерфейсы, которые описывают поведение класса, видимое извне.





Статическая структура

Статическое представление системы изображается с помощью диаграмм классов.




Статическая структура

В языке UML существуют и другие элементы диаграмм: интерфейсы, типы данных, элементы Use Case и сигналы. Они носят общее название классификаторов и ведут себя в большинстве случаев как классы, но с некоторыми дополнениями и ограничениями для каждого типа классификатора.



Элементы проектирования

Модели UML создаются как для логического анализа, так и для проектирования, обеспечивающего реализацию системы.






Элементы проектирования

Компонент — это замещаемая часть системы, которая соответствует некоторому набору интерфейсов и обеспечивает их реализацию.


Компонент должен легко замещаться другими компонентами с тем же набором интерфейсов.





Элементы развертывания

Узел — это вычислительный ресурс периода прогона, который определяет местонахождение исполняемых компонентов и объектов.





Элементы развертывания

Артефакт — это физическая единица информации или описания поведения вычислительной системы.





Динамическое поведение


Существует три способа для моделирования поведения.





Динамическое поведение


Первый представляет историю жизни объекта и его взаимодействия с остальным миром.





Динамическое поведение


Второй описывает паттерны взаимодействия для набора соединенных объектов при реализации определенного поведения.





Динамическое поведение


Третий способ — это описание эволюции процесса исполнения программы в ходе осуществления ею разнообразной деятельности.





Динамическое поведение


Поведение отдельно взятого объекта описывается конечным автоматом. Конечные автоматы изображаются на диаграммах автоматов.





Динамическое поведение

В основе всех представлений поведения лежит набор элементов Use Case, каждый из которых описывает некий фрагмент функциональности системы с точки зрения актера — внешнего пользователя системы.



Представления UML


В языке UML нет четких границ между различными концепциями и конструкциями, но для удобства их можно разделить на несколько представлений.

Представление модели — это просто подмножество конструкций, которое представляет один из аспектов моделируемой системы.



Представление Use Case

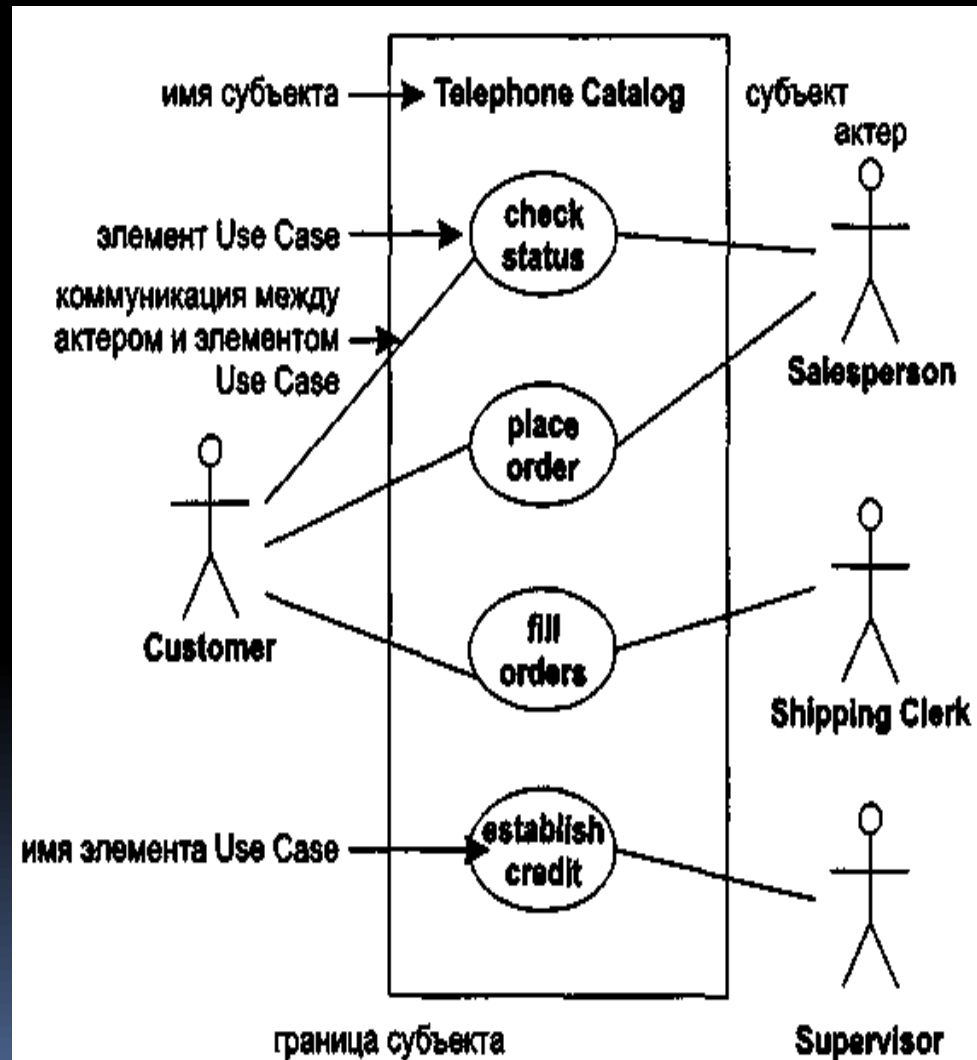
Представление Use Case описывает поведение подсистем, классов или всей системы с точки зрения пользователя.



Представление Use Case

При этом вся функциональность системы делится на транзакции с участием идеализированных пользователей системы, называемых *актерами* (actors). Элементы взаимодействия актера с системой называются *элементами Use Case* (или *прецедентами*, или *вариантами использования*).


Представление Use Case





Представление Use Case

Актер (actor) — это идеализированное представление роли, которую играет внешняя сущность (например, человек или процесс), вступающая во взаимодействие с системой, подсистемой или классом.



Актер определяет возможные виды взаимодействий между системой и некоторым классом ее пользователей.


Представление Use Case

Каждый актер может являться участником одного или нескольких элементов Use Case. Он взаимодействует с элементом Use Case (а следовательно, и с системой или классом, к которому относится данный элемент) посредством обмена сообщениями.



Представление Use Case


На диаграммах актер изображается в виде «проволочного» человечка, под которым указано его имя.





Представление Use Case


Элементом Use Case называется целостный блок видимой извне функциональности, предоставляемой классификатором (который называется субъектом) и выраженной в виде последовательностей сообщений между субъектом и одним или несколькими актерами данного блока.





Представление Use Case


Элемент Use Case описывает некоторый целостный фрагмент поведения системы, не вдаваясь при этом в особенности внутренней структуры субъекта.





Представление Use Case


Определение элемента Use Case содержит все свойственные ему виды поведения: основные последовательности, различные варианты стандартного поведения и возможные при этом исключительные ситуации с указанием ответной реакции на них.





Представление Use Case


Динамическое описание элемента Use Case осуществляется в языке UML с помощью диаграмм состояний, диаграмм последовательности, диаграмм коммуникации или неформальных текстовых описаний. Реализуются элементы Use Case в виде коопераций между классами системы.





Представление Use Case


На системном уровне элементы Use Case представляют собой некоторые видимые для внешних пользователей фрагменты поведения системы.





Представление Use Case


Элемент Use Case в ходе своего выполнения может получать дополнительную информацию от своих актеров.





Представление Use Case

Рассмотрение в терминах элементов Use Case можно применить ко всей системе в целом и распространить на более мелкие ее части — например, подсистемы и отдельные классы.




Представление Use Case

Элемент Use Case — это логическое описание определенной части функциональности системы. Он не является четкой конструкцией, которую можно напрямую реализовать в программном коде. Напротив, каждому элементу Use Case необходимо поставить в соответствие набор классов, на базе которого можно будет реализовать систему.



Представление Use Case


Одна из задач проектирования — найти для реализации элементов Use Case такие классы, которые удачно сочетали бы в себе нужные роли и не создавали при этом излишних сложностей.




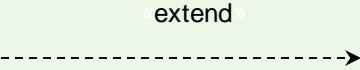
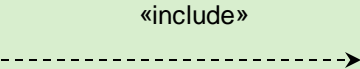
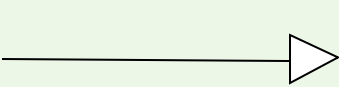


Представление Use Case

Помимо ассоциаций с актерами, элемент Use Case может иметь еще несколько видов отношений.



Представление Use Case

Отношение	Функция	Нотация
Ассоциация (Association)	Линия, по которой происходит обмен информацией между актером и элементом Use Case	
Расширить (Extend)	Включение добавочного поведения в исходный элемент Use Case «без ведома» последнего	
Включить (Include)	Включение добавочного поведения в исходный элемент Use Case, который явно описывает включение	
Обобщение элемента Use Case (Use case generalization)	Отношения между общим элементом Use Case и его более специфической разновидностью (второй наследует черты общего и добавляет к ним свои)	


Представление Use Case

На диаграммах элемент Use Case изображается в виде эллипса. Внутри эллипса или под ним указывается имя элемента. Сплошные линии соединяют элемент Use Case с его актерами.



Представление Use Case

Описание большого элемента Use Case
можно разбить на более простые
элементы.



Представление Use Case

Элемент Use Case может включать в себя черты поведения других элементов Use Case. Такое отношение носит название *отношения включения* (include).

Полученный этим путем элемент Use Case не является специализацией исходного и не может его заменять.

Представление Use Case

Элемент Use Case можно также определить как инкрементное расширение исходного элемента Use Case. Это называется *отношением расширения (extend)*. У исходного элемента Use Case может быть несколько расширяющих вариантов, которые вносят дополнения в его семантику. Все эти варианты могут применяться вместе.


Представление Use Case

Отношения включения и расширения изображаются на диаграммах в виде пунктирных стрелок. Над стрелками указывается ключевое для данного отношения слово («include» или «extend»). Стрелка «include» указывает на включаемый элемент Use Case, а стрелка «extend» — на расширяемый.



Представление Use Case

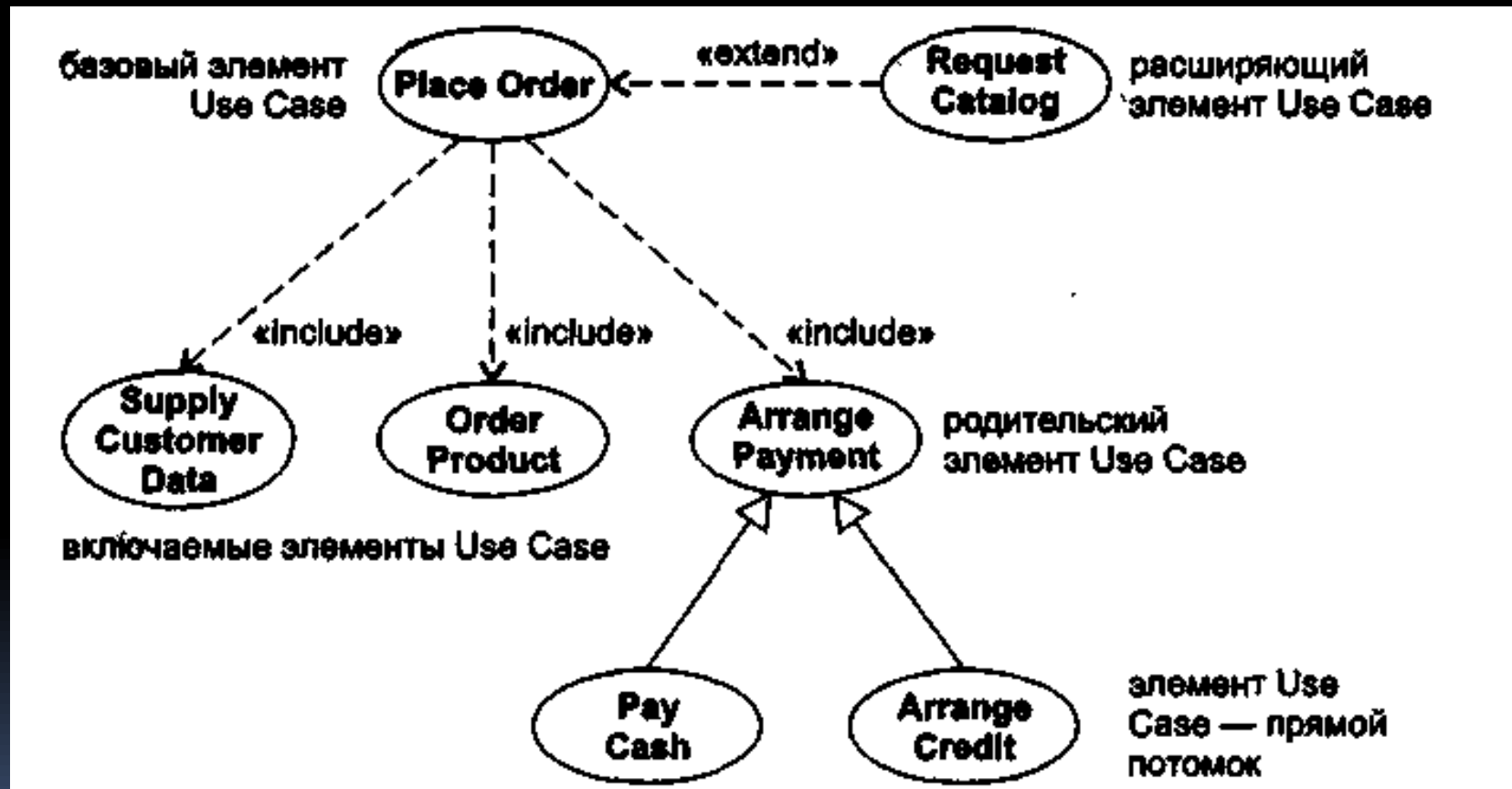
Элемент Use Case может иметь несколько потомков, любой из которых можно подставлять вместо родительского элемента Use Case. Этот механизм называется обобщением элементов Use Case.



Представление Use Case


Обобщение элементов Use Case изображается так же, как и любое другое обобщение — стрелкой с наконечником в виде большого полого треугольника, идущей от потомка к родителю.

Представление Use Case




Структурирование пользовательских историй







Поиск взаимоисключающих пользовательских историй

Типы взаимоисключающих требований:

- Ошибка в пользовательской истории
 - Разные взгляды на одну и ту же проблему
- 



Преобразование диаграммы вариантов использования

- Все элементы, которые не являются включениями или расширениями, должны стать главными узлами списка.
 - Элементы, являющиеся расширениями, должны стать дочерними элементами списка по отношению к тем, кого они расширяют.
- 



Вопросы?