

# ПРОГРАММНАЯ ИНЖЕНЕРИЯ

## ПОИСК И ВЫЯВЛЕНИЕ ТРЕБОВАНИЙ.

# ПОИСК ТРЕБОВАНИЙ

# ПОИСК ТРЕБОВАНИЙ

- ◎ непосредственные пользователи системы
- ◎ другие заинтересованные стороны (например, руководители, специалисты обслуживания, установщики)
- ◎ другие системы, с которыми взаимодействует данная система
- ◎ аппаратные устройства, с которыми взаимодействует данная система
- ◎ правовые и регулирующие ограничения
- ◎ технические ограничения
- ◎ коммерческие цели

# ВЫЯВЛЕНИЕ ТРЕБОВАНИЙ

Естественный язык формируют 3 фильтра:

- ⊙ Пропуск – информация отфильтровывается
- ⊙ Искажение – информация изменяется взаимосвязанными механизмами вымысла и представления
- ⊙ Обобщение – информация обобщается в правила, убеждения и понятия об истинности и ложности

# ПРИМЕР «ПРОПУСКА»

- ◎ **Пример:** «Они используют систему для получения книг на время» – *пропуск*.
- ◎ **Вопрос:** Кто именно использует систему для получения книг на время?
- ◎ **Ответ:** читатели, другие библиотеки и библиотекари.

# ПРИМЕР «ИСКАЖЕНИЯ»

- ◎ **Пример:** «Тот, кто имеет книгу на руках, не может взять другую книгу до тех пор, пока не вернет предыдущую, срок возврата которой истек» – *искажение*.
- ◎ **Вопрос:** Существуют ли такие обстоятельства, при которых кто-либо мог бы взять новую книгу до того, как будут возвращены все имеющиеся на руках книги, срок возврата которых истек?
- ◎ **Ответ:** Фактически существует два обстоятельства, при которых право читателя на получение книг может быть восстановлено. Во первых, все имеющиеся на руках книги, срок возврата которых истек, возвращены; во-вторых, за все невозвращенные книги, срок возврата которых истек, внесена плата.

# ПРИМЕР «ОБОБЩЕНИЯ»

- ◎ **Пример:** «Для получения книг у всех должен быть формуляр» – *обобщение*.
- ◎ **Вопрос:** Есть ли пользователи системы, которым не обязательно иметь формуляр?
- ◎ **Ответ:** Некоторые пользователи системы, например другие библиотеки, могут не иметь формуляра или имеют специальный формуляр с другими сроками и условиями возврата книг.

# КВАНТОР ОБЩНОСТИ

◎ При встрече с квантором общности всегда можно найти пропуск, обобщение и искажение

- **все**

- **каждый**

- **всегда**

- **никогда**

- **никто**

- **несколько**



# ИНТЕРВЬЮ

- ◎ Не заблуждайтесь, вам может *казаться* что вы все очень хорошо понимаете;
- ◎ Задавайте контекстно-свободные вопросы;
- ◎ Не занимайтесь телепатией. Телепатия – это **заблуждение** по поводу того, что вам известны чьи-то чувства.
- ◎ Лучший способ записи информации во время интервью – бумага и ручка!
- ◎ Анкеты не заменяют интервью, но могут быть полезным дополнением.

# ДОКУМЕНТАЦИЯ ТРЕБОВАНИЙ: UML-ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

# ЧТО ТАКОЕ МОДЕЛИРОВАНИЕ ПРЕЦЕДЕНТОВ?

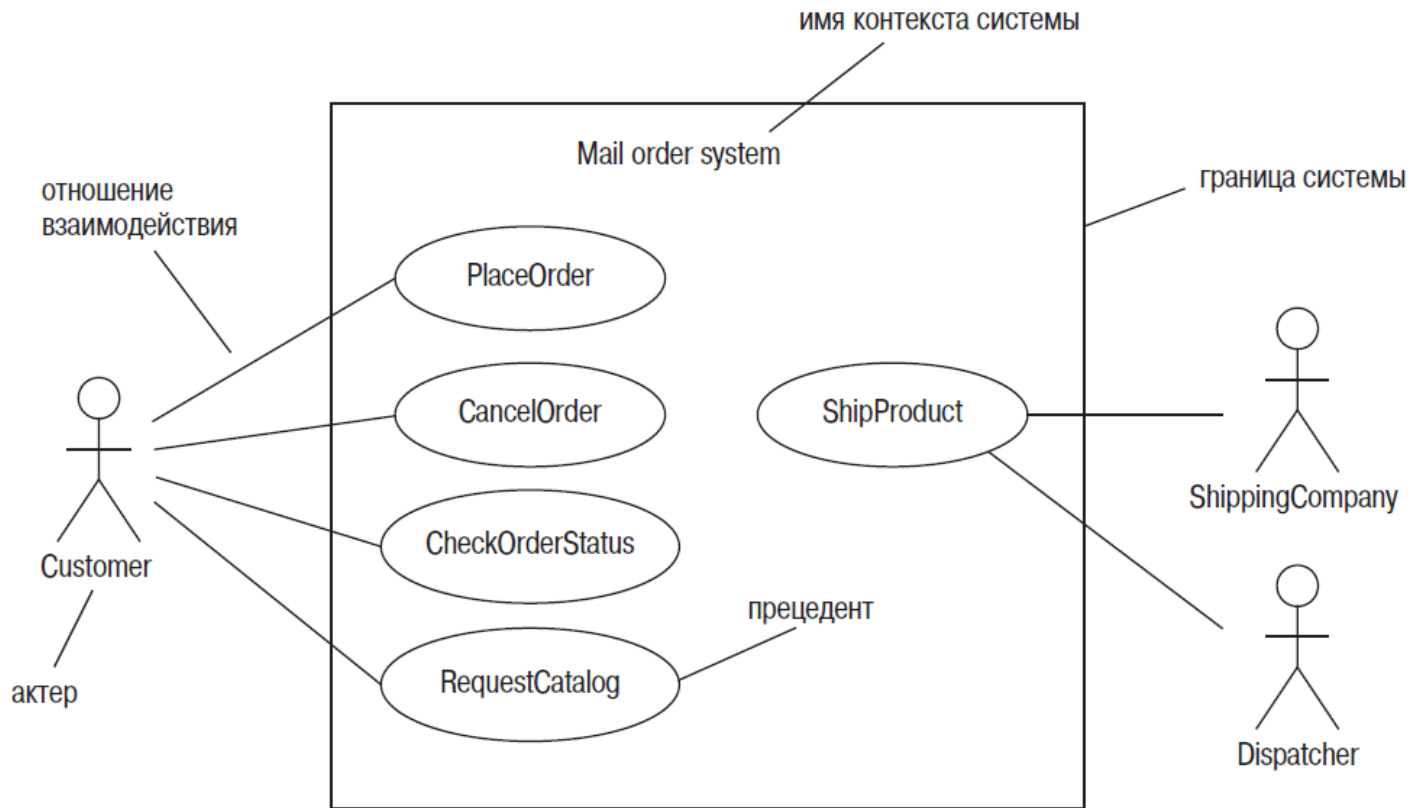
- ◎ Моделирование прецедентов – это форма выработки требований.
- ◎ Процесс моделирования прецедентов:
  - ◎ Установить границы системы
  - ◎ Выявить актеров
  - ◎ Выявить прецеденты
  - ◎ Повторить, пока не стабилизируются

# МОДЕЛЬ ПРЕЦЕДЕНТОВ

Состоит из 4-х компонентов:

1. **Граница системы** – прямоугольник, очерчивающий прецеденты для обозначения границы моделируемой системы
2. **Актеры** – роли, выполняемые людьми или сущностями, использующими систему
3. **Прецеденты** - то, что актеры могут делать с системой
4. **Отношения** – отношения между актерами и прецедентами

# ПРИМЕР ДИАГРАММЫ ПРЕЦЕДЕНТОВ

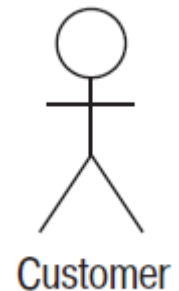


# ГРАНИЦЫ СИСТЕМЫ

- ◎ **Границы системы (контекст системы в терминах UML)** отделяет систему от всего остального мира.
- ◎ Из-за неясности границ системы возникают серьезные проблемы при проектировании.
- ◎ Контекст изображается в виде прямоугольника с именем системы. *Актеры размещаются вне границ блока, а прецеденты – внутри*

# АКТЕРЫ

- ◎ **Актеры** – это роли, исполняемые сущностями, непосредственно взаимодействующими с системой.
- ◎ Одна и та же роль может исполняться многими разными сущностями одновременно либо последовательно во времени.
- ◎ Например, одну роль `Customer` (покупатель) может исполнять несколько реальных человек (сущностей). Некоторые из них могут исполнять роль `System Administrator`.



# ИДЕНТИФИКАЦИЯ АКТЕРОВ

- ◎ Актеры всегда являются внешними по отношению к системе, следовательно, находятся вне вашего контроля.
- ◎ Актеры взаимодействуют *непосредственно с системой* – так они помогают в определении контекста системы.
- ◎ Актеры представляют роли, исполняемые людьми или сущностями по отношению к системе, а не конкретных людей или сущностей.
- ◎ У каждого актера должно быть короткое, осмысленное с прикладной точки зрения имя.





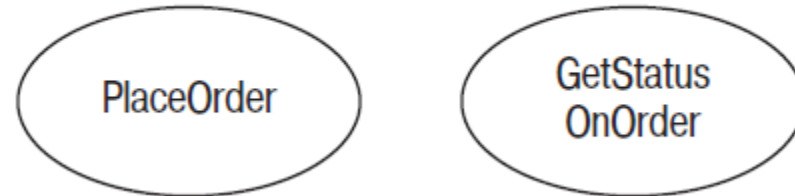
# ИДЕНТИФИКАЦИЯ АКТЕРОВ

- ◎ Чтобы выявить актеров, спросите: «Кто или что использует или взаимодействует с системой?»:
  - ◎ Кто устанавливает систему?
  - ◎ Кто или что запускает и выключает систему?
  - ◎ Кто обслуживает систему?
  - ◎ Какие системы взаимодействуют с данной системой?
  - ◎ Кто или что получает и предоставляет информацию системе?
  - ◎ Происходит ли что-нибудь в точно установленное время?



# ПРЕЦЕДЕНТЫ

- ◎ **Прецедент** – это описание последовательности действий, включая альтернативные и ошибочные последовательности, которые система, подсистема или класс могут осуществлять, взаимодействуя с внешними актерами.
- ◎ **Прецедент** – это что-то, что должна делать система по желанию актера. Это «вариант использования» системы конкретным актером.



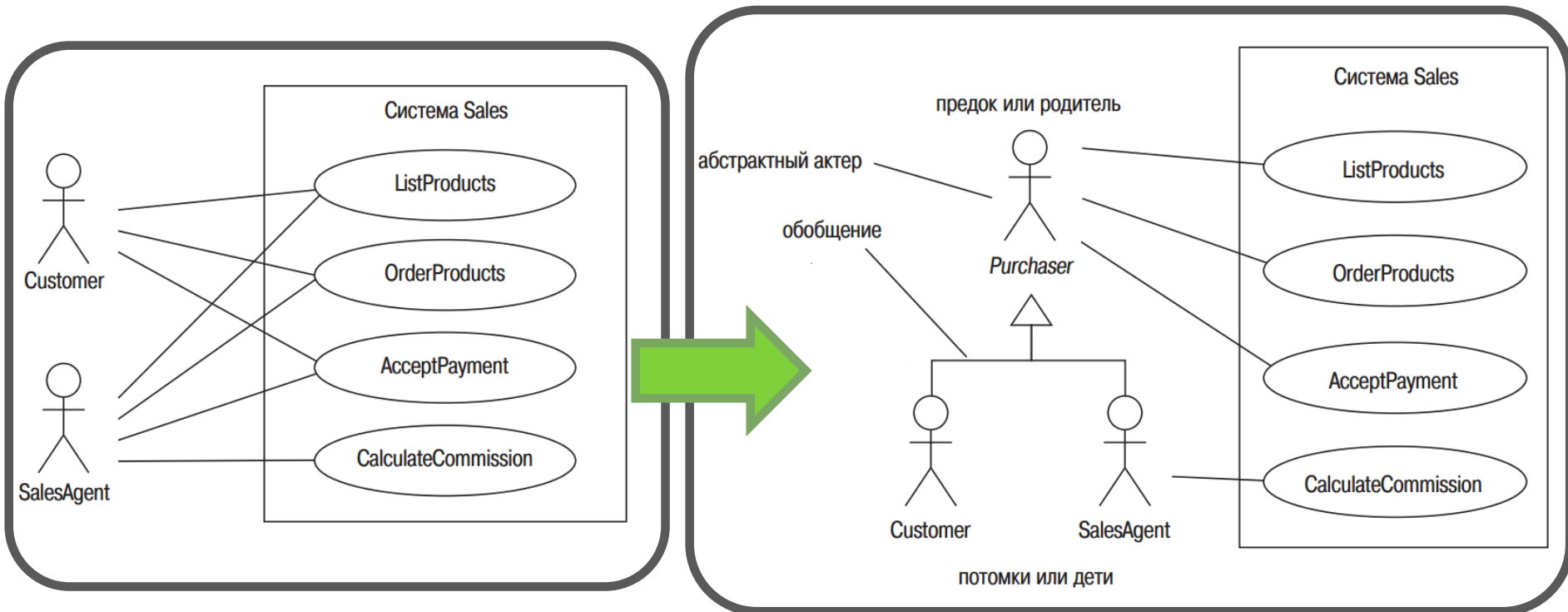
- ◎ Прецеденты всегда инициируются актером;
- ◎ Прецеденты всегда описываются с точки зрения актеров.

# ИДЕНТИФИКАЦИЯ ПРЕЦЕДЕНТОВ

- ⊙ Чтобы найти прецедент, надо спросить: «Как каждый из актеров использует систему?» и «Что система делает для каждого актера?»
- ⊙ Во время идентификации прецедентов могут обнаружиться некоторые новые актеры.
- ⊙ Каждому прецеденту должно быть присвоено короткое описательное имя, представляющее собой глагольную группу (в конце концов, прецедент означает «*выполнить*» что-нибудь!).
- ⊙ Во время идентификации прецедентов могут обнаружиться некоторые новые актеры. Это нормально. Иногда приходится очень тщательно анализировать функциональность системы, чтобы выявить всех актеров, причем *правильно выявить*.

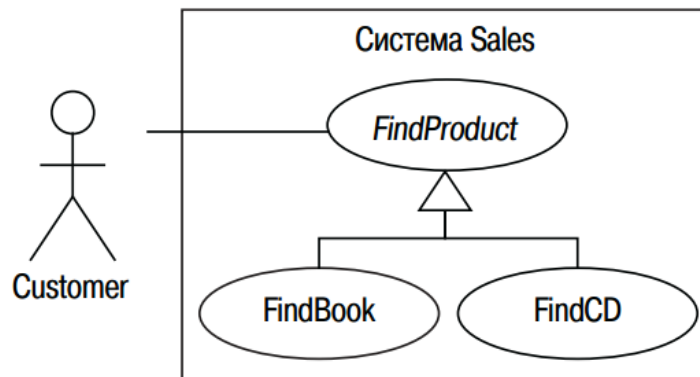
# ОБОБЩЕНИЕ АКТЕРОВ

- ◎ Обобщение актеров выносит поведение, общее для двух или более актеров, в актера-родителя.
- ◎ Актер-потомок может использоваться везде, где ожидается актер-предок.



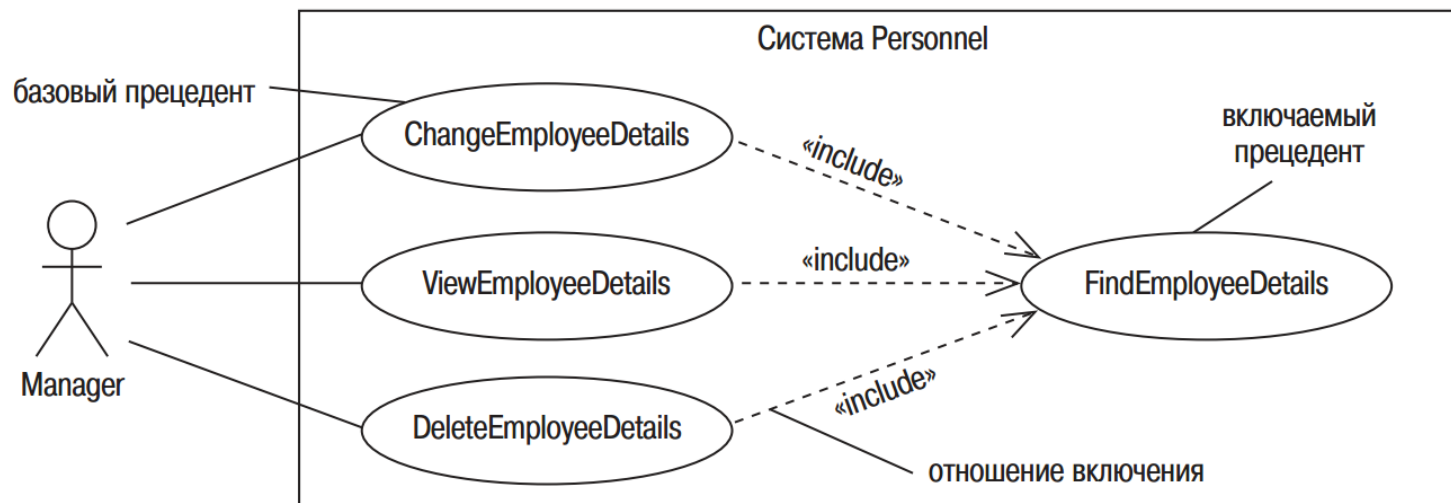
# ОБОБЩЕНИЕ ПРЕЦЕДЕНТОВ

- ◎ Обобщение прецедентов выносит поведение, общее для одного или более прецедентов, в родительский прецедент.
- ◎ В обобщении прецедентов дочерние прецеденты представляют более специализированные формы их родителей. Дочерний прецедент автоматически наследует все возможности своего родителя. Потомки могут:
  - ◎ наследовать возможности родительского прецедента;
  - ◎ вводить новые возможности;
  - ◎ переопределять (менять) унаследованные возможности.



# ОТНОШЕНИЕ «ВКЛЮЧЕНИЕ» (INCLUDE)

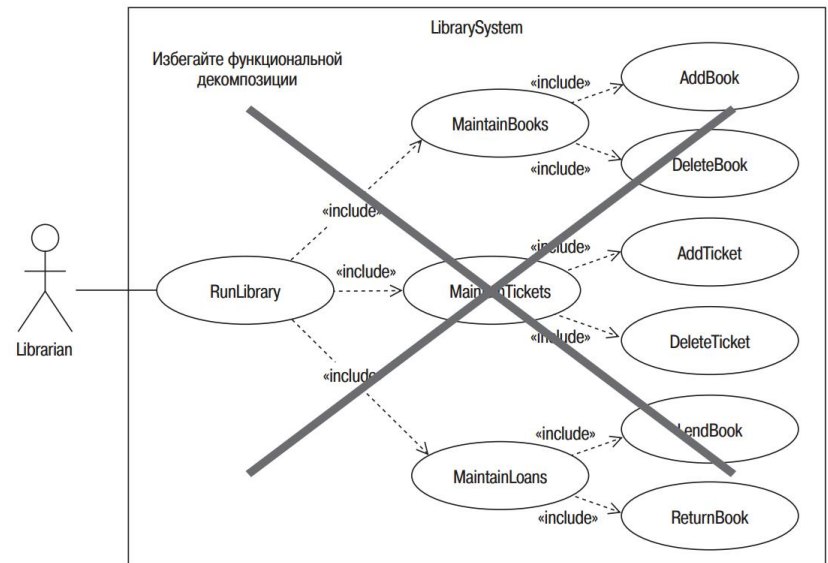
- ① Отношение «include» выносит шаги, общие для нескольких прецедентов, в отдельный прецедент, который потом включается в остальные.
- ① Включающий прецедент мы называем *базовым*, а тот прецедент, который включается, *включаемым*. Включаемый прецедент предоставляет поведение своему базовому прецеденту.



- ① Включаемые прецеденты могут быть как полными, так и неполными. Если включаемый прецедент неполный, он просто содержит часть потока событий, которая имеет смысл только тогда, когда включена в соответствующий базовый прецедент.

# СОВЕТЫ ПО ПРЕЦЕДЕНТАМ

- ☉ Делайте прецеденты короткими и простыми. Есть хорошее правило: основной поток прецедента должен помещаться на одной странице.
- ☉ Помните, прецеденты создаются для того, чтобы понять, чего актеры ждут от системы, а не как она должна это осуществлять
- ☉ Избегайте функциональной декомпозиции – она не подходит для модели прецедентов.
- ☉ Применение функциональной декомпозиции говорит о том, что аналитик неправильно продумал систему. Обычно это свидетельствует о том, что он обучен более традиционным методам процедурного программирования и пока что не уловил принципа ОО программирования.



# ДЕТАЛИЗАЦИЯ ПРЕЦЕДЕНТОВ



# ДЕТАЛИЗАЦИЯ ПРЕЦЕДЕНТА

- ◎ Детализация прецедента – это точное определение каждого прецедента.
- ◎ Детализацию всех прецедентов можно производить параллельно, постепенно повышая уровень детализации.
- ◎ Итог: прецедент с спецификацией.

# ПРИМЕР СПЕЦИФИКАЦИИ ПРЕЦЕДЕНТА

имя прецедента	<p>Прецедент: PaySalesTax</p>		
идентификатор прецедента		ID: 1	
краткое описание		Краткое описание: Выплата налога с оборота в Налоговое управление по окончании налогового периода.	
		Главные актеры: Time (Время)	
актеры, вовлеченные в прецедент		Второстепенные актеры: TaxAuthority (налоговое управление)	
		Предусловия: 1. Конец налогового периода.	
состояние системы до начала прецедента		<p>Основной поток:                    неявный актер Time</p> <ol style="list-style-type: none"> <li>1. Прецедент начинается в конце налогового периода.</li> <li>2. Система определяет сумму Налога с оборота, которую необходимо выплатить Налоговому управлению.</li> <li>3. Система посылает электронный платеж в Налоговое управление.</li> </ol>	
фактические этапы прецедента			
состояние системы после окончания прецедента		Постусловия: 1. Налоговое управление получает соответствующую сумму Налога с оборота.	
альтернативные потоки		Альтернативные потоки: Нет.	

# ОСНОВНОЙ ПОТОК

- ◎ Основной поток описывает «идеальный» ход развития событий в прецеденте.
- ◎ Альтернативные потоки могут перехватывать ошибки, ответвления и прерывания основного потока.
- ◎ Основной поток *всегда начинается с действий главного актера*, направленных на инициацию прецедента.

# ЗАПИСЬ ОСНОВНОГО ПОТОКА

- ◎ Удачным способом начала потока можно считать следующую форму записи:

Прецедент начинается, когда <актер> <действие>.

- ◎ Каждый этап потока прецедента должен быть выражен в следующей форме:

<номер> <ктолибо> <совершает некоторое действие>.

- ◎ Использование пассивного залога для описания этапа является неверным: «Вводятся данные покупателя»

# КЛЮЧЕВЫЕ СЛОВА В ОПИСАНИИ ОСНОВНОГО ПОТОКА

- ◎ Альтернативные потоки могут быть заменены с помощью ключевого слова «Если»

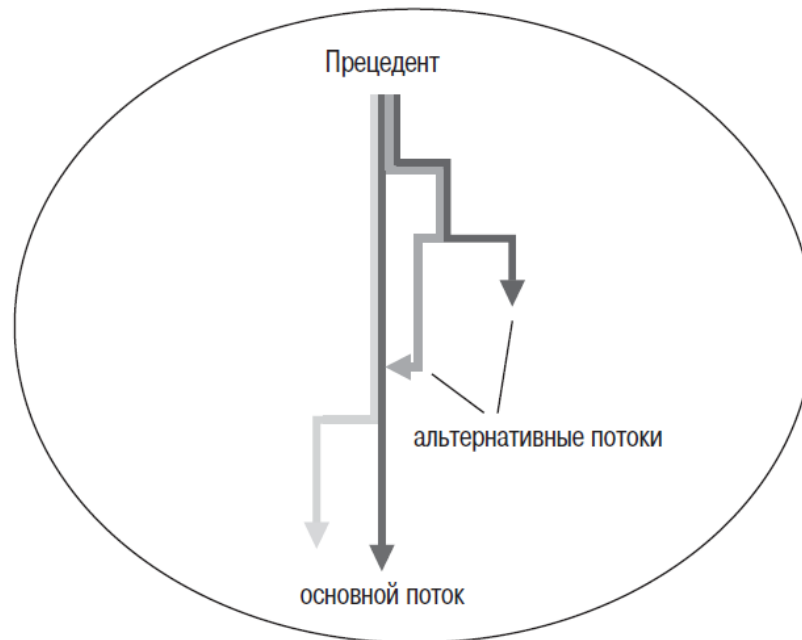
Основной поток:

1. Прецедент начинается, когда Покупатель выбирает товарную позицию в корзине.
2. Если Покупатель выбирает «удалить позицию».
  - 2.1. Система удаляет позицию из корзины.
3. Если Покупатель вводит новое количество.
  - 3.1. Система обновляет количество товаров в корзине.

- ◎ Аналогично можно использовать ключевые слова «Пока ...» или «Для всех элементов...» для описания циклов или повторений

# АЛЬТЕРНАТИВНЫЕ ПОТОКИ

- © Могут не возвращаться в основной поток после выполнения, т.к. обрабатывают ошибки, исключительные ситуации и т.п.



# ДОКУМЕНТАЦИЯ АЛЬТЕРНАТИВНЫХ ПОТОКОВ

Альтернативные потоки:  
InvalidEmailAddress  
InvalidPassword  
Cancel

Обычно  
альтернативные  
потоки  
документируются  
отдельно.

Для них создается  
аналогичная  
спецификация,  
только  
используется ID 2-  
го уровня

Альтернативный поток: CreateNewCustomerAccount:InvalidEmailAddress
ID: 5.1
Краткое описание: Система сообщает Покупателю, что он ввел недействительный адрес электронной почты.
Главные актеры: Покупатель
Второстепенные актеры: Нет.
Предусловия: 1. Покупатель ввел недействительный адрес электронной почты.
Альтернативные потоки: 1. Альтернативный поток начинается после шага 2.2 основного потока. 2. Система сообщает Покупателю, что он ввел недействительный адрес электронной почты.
Постусловия: Нет.

# ОТОБРАЖЕНИЕ ТРЕБОВАНИЙ



# ОТОБРАЖЕНИЕ ТРЕБОВАНИЙ

- При отображении требований устанавливаются взаимосвязи между моделью требований и моделью прецедентов.
- Может проводиться как с помощью специального ПО (RequisitePro, DOORS) или вручную.

		Прецедент			
		П <sub>1</sub>	П <sub>2</sub>	П <sub>3</sub>	П <sub>4</sub>
Требование	T1	X			
	T2		X	X	
	T3			X	
	T4				X
	T5	X			

# ПРИМЕНЕНИЕ МОДЕЛИ ПРЕЦЕДЕНТОВ

- ◎ Применяются, когда:
  - преобладают функциональные требования;
  - много типов пользователей (много актеров);
  - в системе много интерфейсов (много актеров).
- ◎ Не стоит применять, если:
  - преобладают нефункциональные требования;
  - в системе мало пользователей;
  - в системе мало интерфейсов.
- ◎ Например, сложно описать модель прецедентов для встроенных систем, систем со сложными алгоритмами.