Программная инженерия

Моделирование прецедентов

© Радченко Г.И., кафедра СП ЮУрГУ

Основы прецедентов

Что такое моделирование прецедентов?

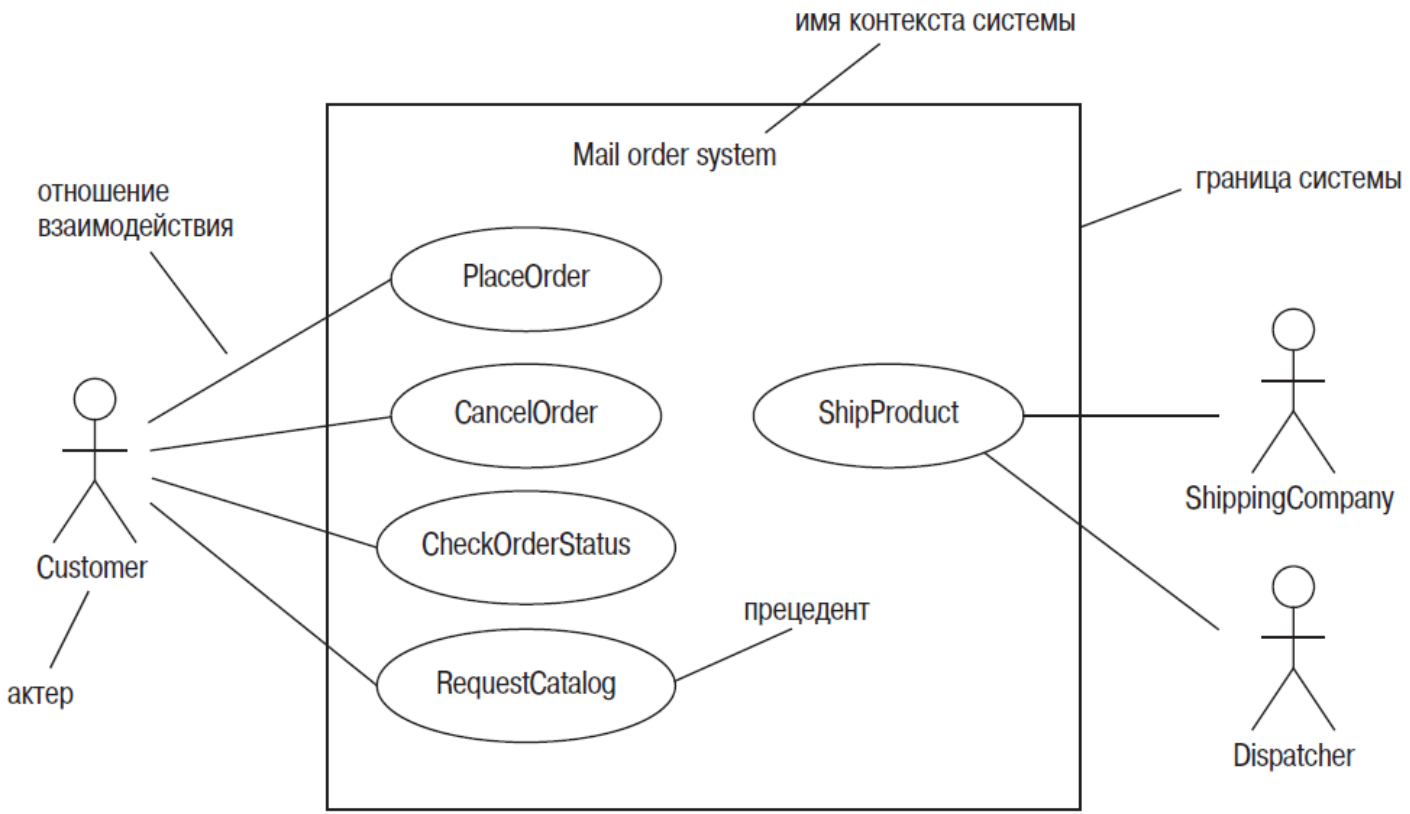
- **Моделирование прецедентов –** это форма выработки требований.
- **Процесс моделирования прецедентов:**
  1. Установить границы системы
  2. Выявить актеров
  3. Выявить прецеденты
  4. Повторить, пока не стабилизируются

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

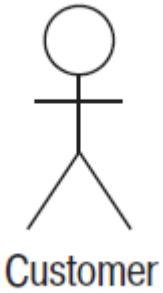


Границы системы

- **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

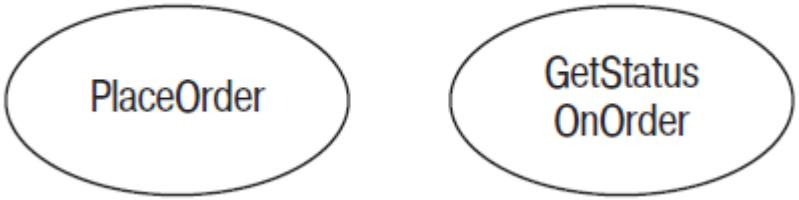
- Актеры всегда являются внешними по отношению к системе, следовательно, находятся вне вашего контроля.
- Актеры взаимодействуют *непосредственно с системой – так они по*могают в определении контекста системы.
- Актеры представляют роли, исполняемые людьми или сущностями по отношению к системе, а не 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.



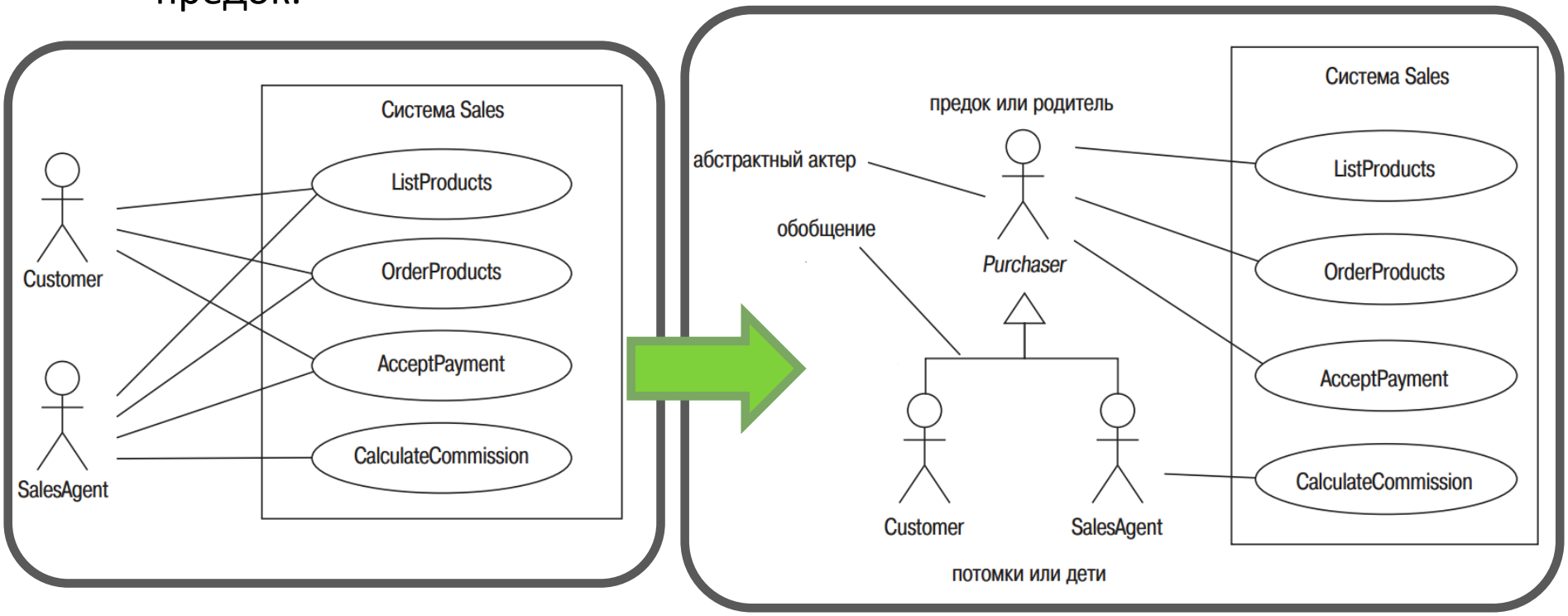
- Прецеденты *всегда инициируются актером;*
- Прецеденты *всегда описываются с точки зрения актеров.*

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.*

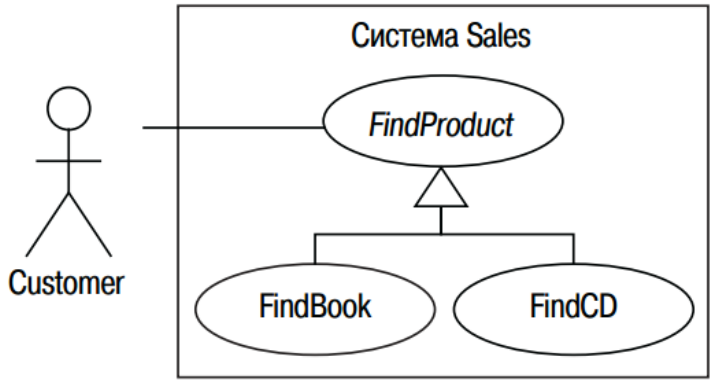
Обобщение актеров

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



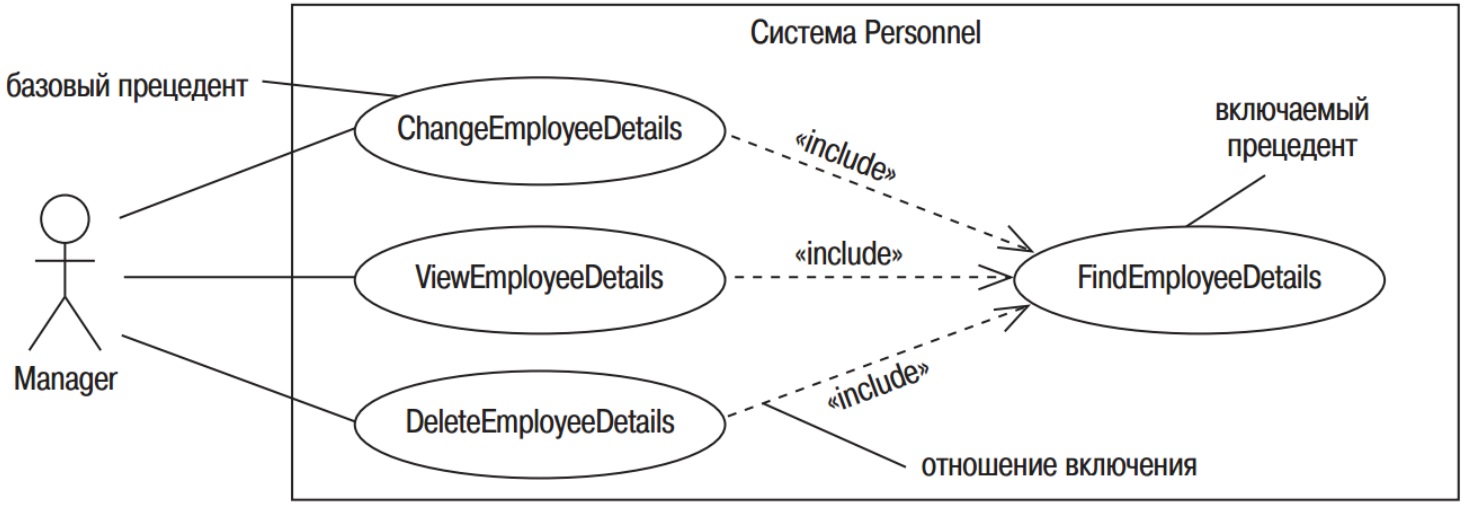
Обобщение прецедентов

- 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)

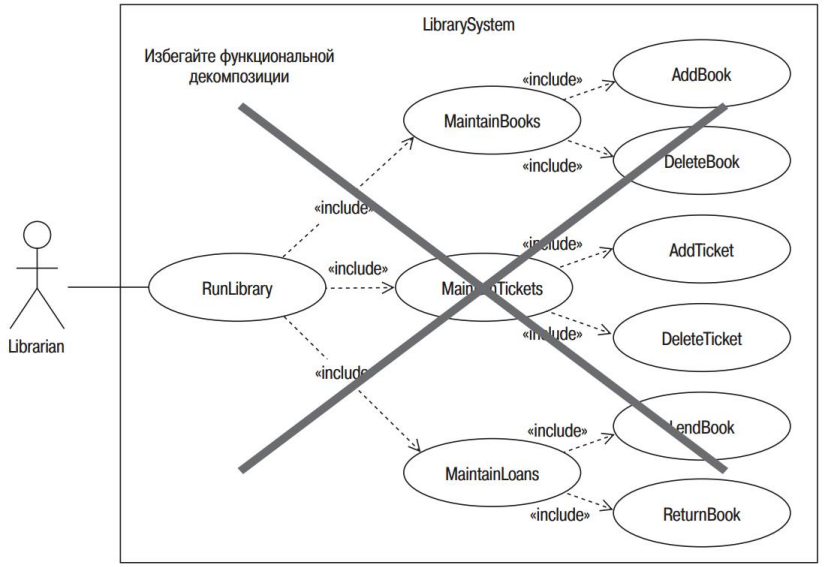
- Отношение «include» выносит шаги, общие для нескольких прецедентов, в отдельный прецедент, который потом включается в остальные.
- Включающий прецедент мы называем *базовым*, а тот прецедент, который включается, *включаемым*. Включаемый прецедент предоставляет поведение своему базовому прецеденту.



- 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.

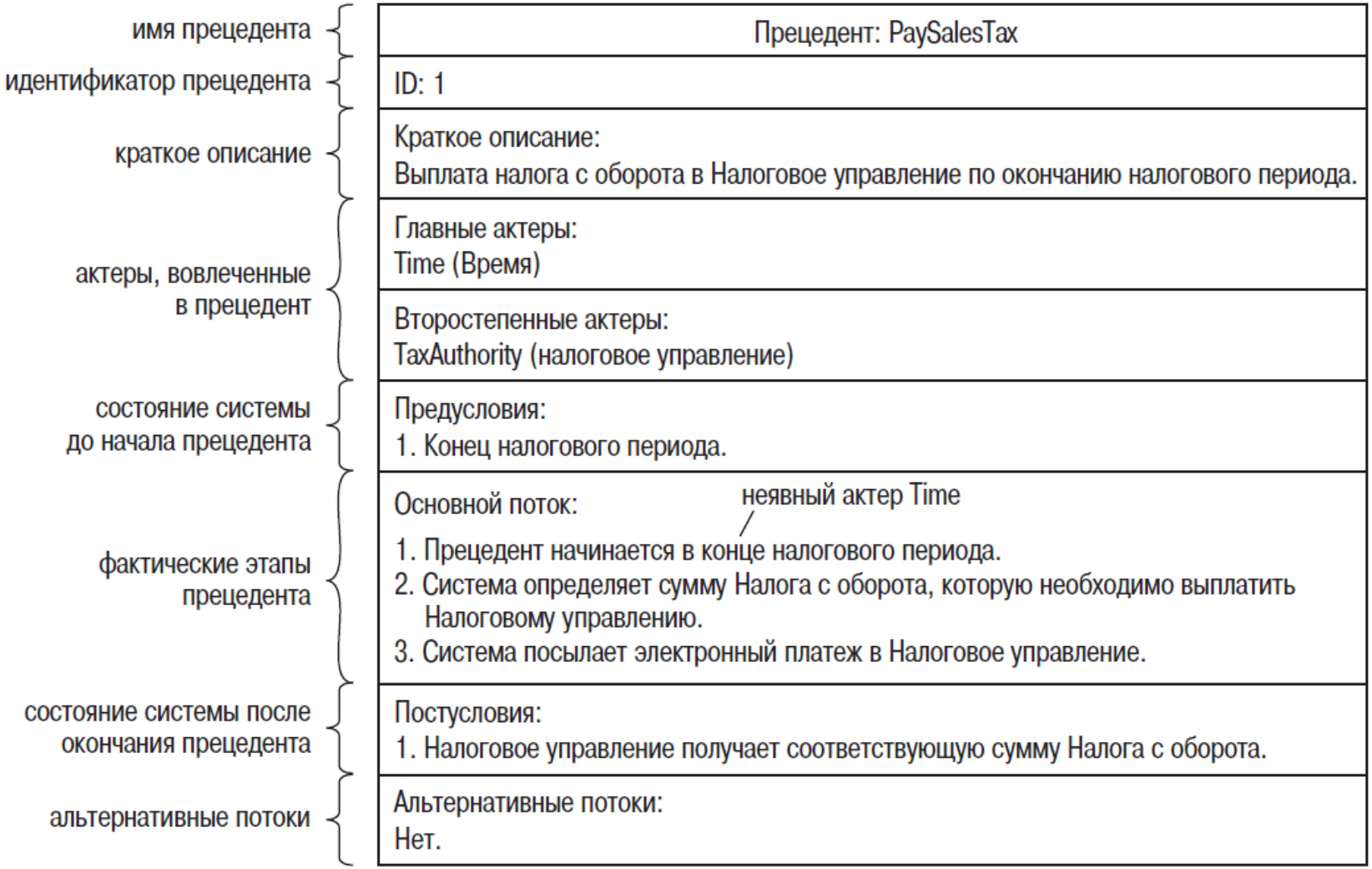


Детализация прецедентов

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



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»

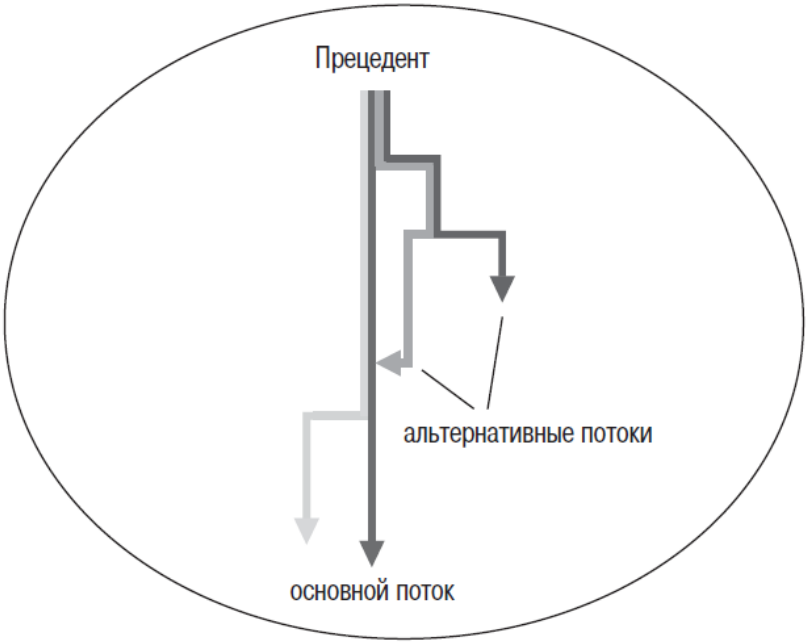
Основной поток:

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			
		$П_1$	$П_2$	$П_3$	$П_4$
Требование	Т1	X			
	Т2		X	X	
	Т3			X	
	Т4				X
	Т5	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.