Объектно-ориентированное проектирование ПС

Лекция 7

Способы декомпозиции системы

- ***Функциональная декомпозиция***– на основе потока данных с выделением обрабатывающих функций
- ***Объектная декомпозиция*** – на основе выделения сущностей, обладающих собственными наборами данных, состояниями и наборами операций

Способы декомпозиции системы

- В первом случае внимание концентрируется на ***порядке*** происходящих событий (действиях)
- Во втором – на ***агентах***, являющихся либо объектами, либо субъектами действий

Структурные единицы

- Основной структурной единицей при функциональной декомпозиции является ***процедура***, как программная реализация алгоритма
- Основной структурной единицей при объектно-ориентированной декомпозиции является ***объект*** как объединение данных и действий над ними

Начало проектирования

- Результаты анализа предметной области представляются в виде ***диаграммы прецедентов*** с ***описанием прецедентов***
- Следующий шаг – создание ***концептуальной модели разрабатываемой системы***, т.е. описание ее в терминах предметной области

Ключевые абстракции

- В концептуальной модели используются ***ключевые абстракции*** предметной области
- ***Ключевая абстракция*** - это класс или объект, который входит в словарь проблемной области

Выделение объектов

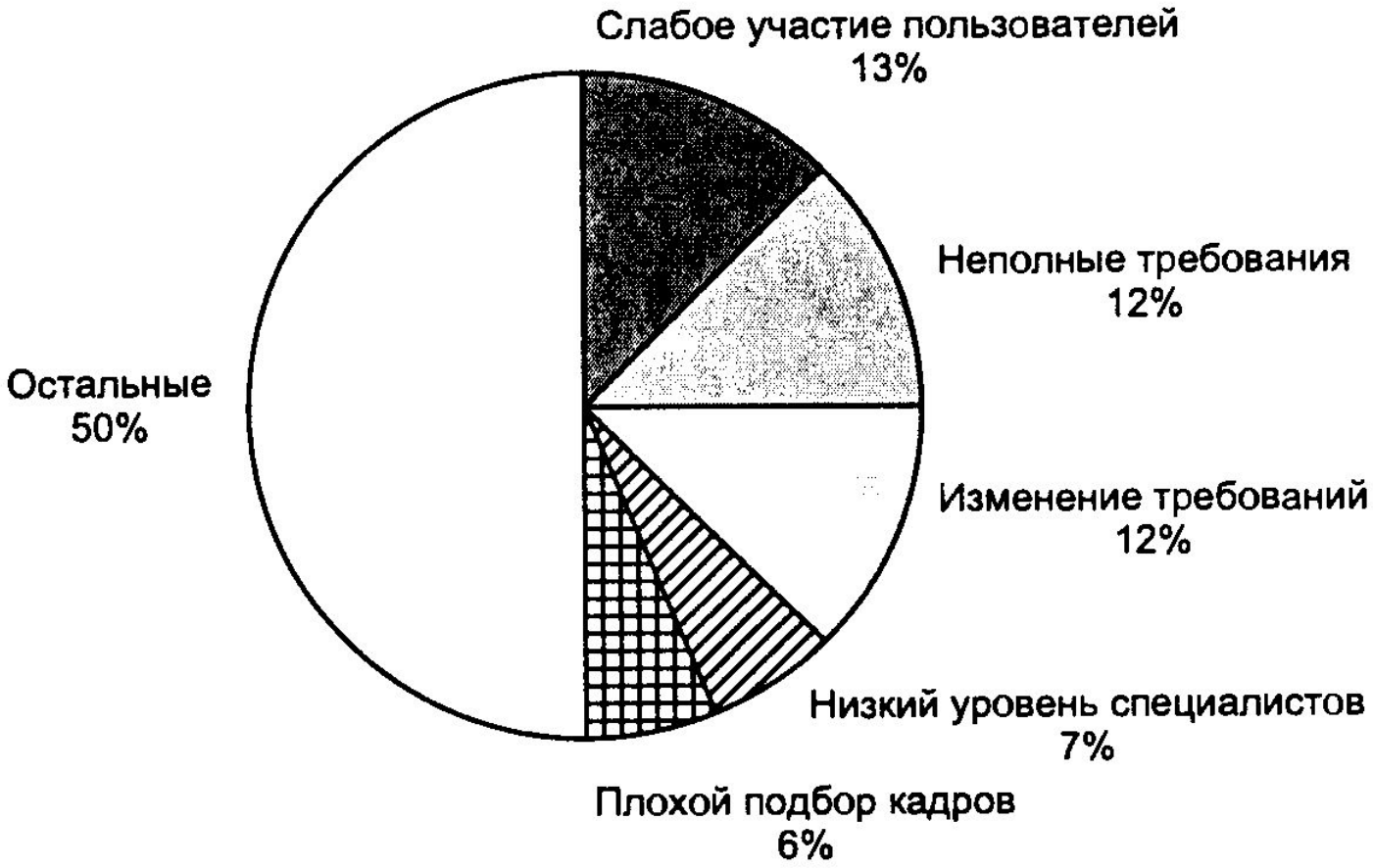
- Ключевые абстракции определяют границы системы: выделяют то, что входит в нее и важно для нас, а также устраняют все лишнее
- На более поздних этапах проектирования большинство ключевых абстракций будут отображены в ***программные классы***

ПРЕЦЕДЕНТЫ И ТРЕБОВАНИЯ

Требования

- Уточним понятие прецедента и его связь с понятием «требования к программной системе»
- ***Требования*** (requirements) – это возможности или условия, которым должна удовлетворять разрабатываемая система

Требования и риски



Формулировка требований

- Требования к системе могут быть представлены в виде списка кратких утверждений типа: «система должна обеспечивать выполнение такой-то операции»
- Такой подход связан, по крайней мере, с двумя недостатками:
  - нечеткостью формулировки требований,
  - неполнотой их списка

Формулировка требований

- Для описания функциональных требований к системе был предложен подход, основанный на использовании ***прецедентов*** (Ivar Jacobson, 1986)
- ***Прецедент (use case)*** – это описание способа использования системы с целью получения некоторого результата, значимого для исполнителя

Прецедент и сценарии

- Прецедент однозначно связан с результатом, но достижение одного и того же результата может происходить разными путями, в зависимости от тех или иных условий
- Конкретная последовательность действий или взаимодействий между системой и исполнителем в рамках одного прецедента называется ***сценарием*** (*scenario*)

Прецедент и сценарии

- Таким образом, прецедент – это набор различных сценариев использования системы для получения одного и того же значимого результата
- Сценарий, реализующийся с наибольшей вероятностью, называется *основным*, а остальные сценарии – *альтернативными*

Исполнитель

- Под ***исполнителем*** (actor) в предыдущих определениях понималась некоторая сущность, обладающая поведением (человек или организация, представленные ролью, или компьютерная система)

Это т пример описан в книге Крэга Лармана «Применение UML и шаблонов проектирования»

ПРИМЕР РАЗРАБОТКИ

Постановка задачи

- Разработать программное обеспечение для системы организации товарооборота и обработки платежей в магазинах розничной торговли

Модель разработки

- Разработка системы будет вестись в рамках модели **RUP** - адаптивного итеративного процесса с постепенным наращиванием функциональности ПС и уточнением требований посредством механизма обратной связи

Фазы процесса разработки

- **Начало** – анализ проблемы, формирование представлений о функциях системы и основных требованиях к ней
- **Развитие** –реализация базовой части системы и уточнение требований; осуществляется через последовательность итераций

Фазы процесса разработки

- **Конструирование** – разработка системы в полном объеме и окончательная формулировка требований; осуществляется через последовательность итераций, каждая из которых завершается созданием ***релиза***
- **Внедрение** – развертывание системы и бета-тестирование

Основные задачи:

- формирование представления о проекте
- формулирование исходных требований к системе
- оценка стоимости проекта
- идентификация основных рисков

ЭТАП НАЧАЛО

Анализ предметной области

- Предметная область – **ро́зничная торго́вля** (англ. *retail*)
- *Что делается* – производится продажа товаров конечному потребителю (частному лицу)
- *Как делается* – покупатель отбирает необходимые ему товары и производит оплату в кассе

Анализ предметной области

- *Где делается* – основным предприятием розничной торговли является магазин (дискаунтер, универмаг, универсам и т.д.)
- *Кто делает* – субъектами процесса розничной торговли являются продавец (менеджер торгового зала, кассир) и покупатель

Анализ предметной области

- *Когда делается* – каждый магазин имеет фиксированный график работы
- *Зачем делается* – розничная торговля обеспечивает удовлетворение потребностей населения в товарах различного назначения и получение торговой прибыли

Проблемы

- Низкая скорость выполнения операции оплаты покупок
- Ошибки кассиров при подсчете стоимости товаров и расчете с покупателями
- Сложность ведения учета проданных товаров
- Большие объемы работ по подготовке данных для системы анализа

Пути решения

- Создание компьютеризированной системы оплаты покупок Point-Of-Sale (POS-система)
- Интеграция этой системы с существующими компьютерными системами поддержки торговой деятельности – системой складского учета, системой анализа торговой деятельности, бухгалтерской системой

POS-терминал

- POS-система реализуется в виде набора *POS-терминалов*
- Каждый POS-терминал представляет собой программно-аппаратный комплекс, установленный на месте, где кассир осуществляет прием платежей от клиентов (АРМ кассира)

Аппаратная часть

- Аппаратная часть POS-терминала включает:
  - системный блок ПК,
  - фискальный регистратор,
  - POS-монитор кассира,
  - денежный ящик,
  - программируемую клавиатуру,
  - считыватель карт,
  - считыватель штрих-кодов

Фискальный регистратор

- Фискальный регистратор (ФР) – это устройство, обеспечивающее не корректируемую ежесуточную или ежесменную регистрацию наличных денежных расчетов и энергонезависимое долговременное хранение итоговой информации
- Он сохраняет результаты продаж в фискальной памяти в течение всего срока его эксплуатации, а также распечатывает чеки покупателя.

Аппаратная часть POS-терминала

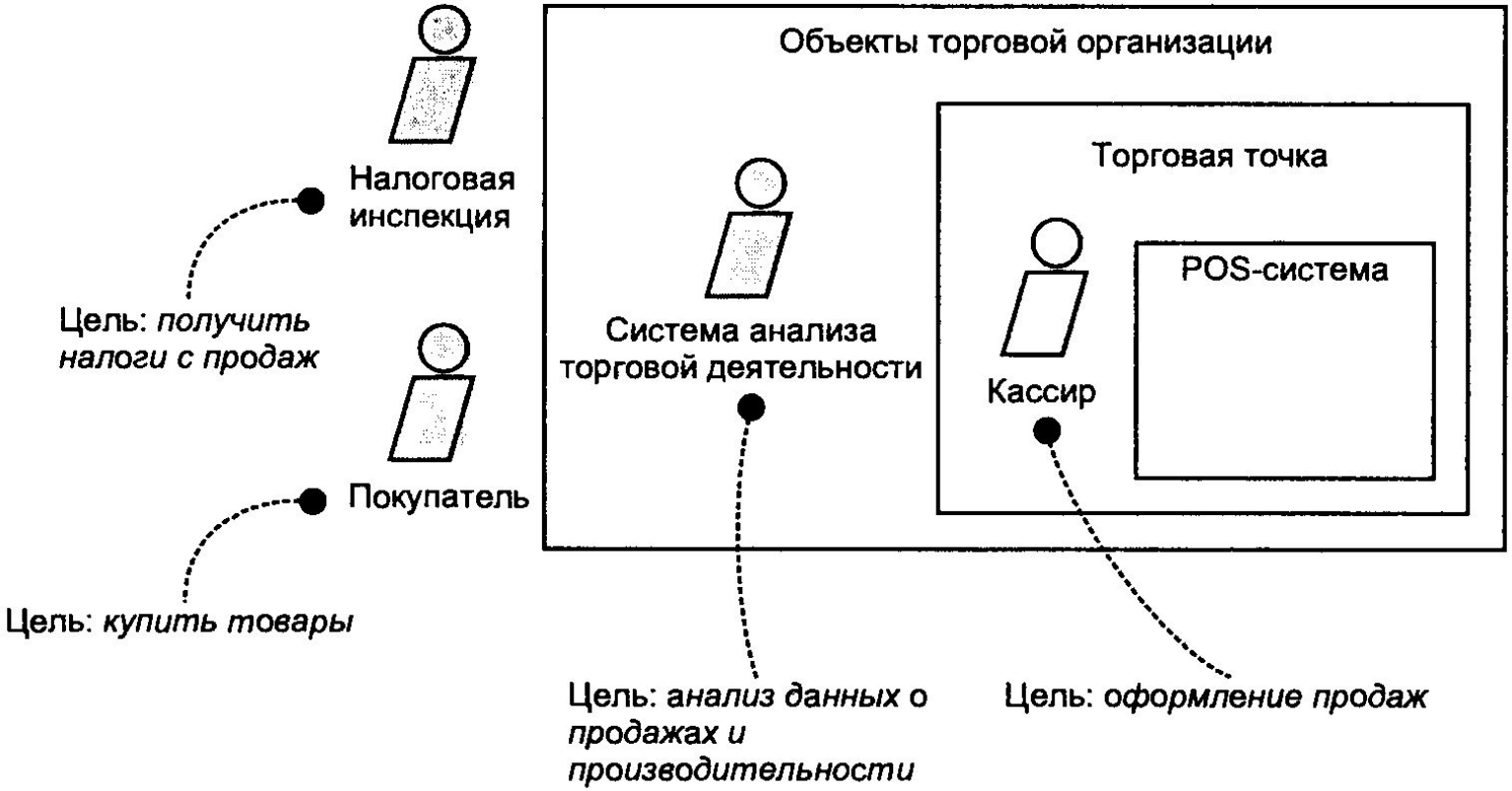
Интерфейс пользователя

- POS-терминал должен иметь интерфейс взаимодействия с пользователем для
  - поиска нужного товара и получения его характеристик;
  - формирования и печати чеков;
  - подсчета сдачи;
  - выполнения различных отчетов

Определение границ системы

- Границы системы проще всего определить установив *основных исполнителей*, потребности которых удовлетворяются данной системой
- Для этого надо ответить на вопросы:
  - Кто будет снабжать систему информацией?
  - Кто будет получать информацию от системы?
  - Кто будет осуществлять поддержку и обслуживание системы?
  - Использует ли система внешние ресурсы?

Границы системы



Основные исполнители

- Кассир
  - оформляет продажи,
  - выполняет возврат товара,
  - регистрирует выручку;
- Системный администратор
  - редактирует список пользователей,
  - управляет безопасностью;

Основные исполнители

- Менеджер
  - включает систему,
  - выключает систему
- Система анализа торговой деятельности
  - анализирует информацию о продажах и оценивает производительность

Прецеденты

- Следует иметь в виду, что прецеденты могут быть определены на разных уровнях детализации
- При анализе требований следует сосредоточить внимание на уровне ***элементарных бизнес-процессов***, т.е. задач достаточно высокого уровня
- Основной сценарий таких прецедентов содержит 5 – 10 шагов

Прецеденты для POS-системы

1. Включение системы
2. Регистрация в системе
3. Оформление покупки
4. Возврат товара
5. Регистрация выручки
6. Управление списком пользователей
7. Управление безопасностью
8. Анализ деятельности
9. Выключение системы

Ранжирование прецедентов

- Учитываются следующие факторы:
  - влияние на архитектуру (например, добавление новых классов);
  - наличие рискованных, срочных или сложных функций;
  - потребность в дополнительных исследованиях;
  - степень важности соответствующего бизнес-процесса

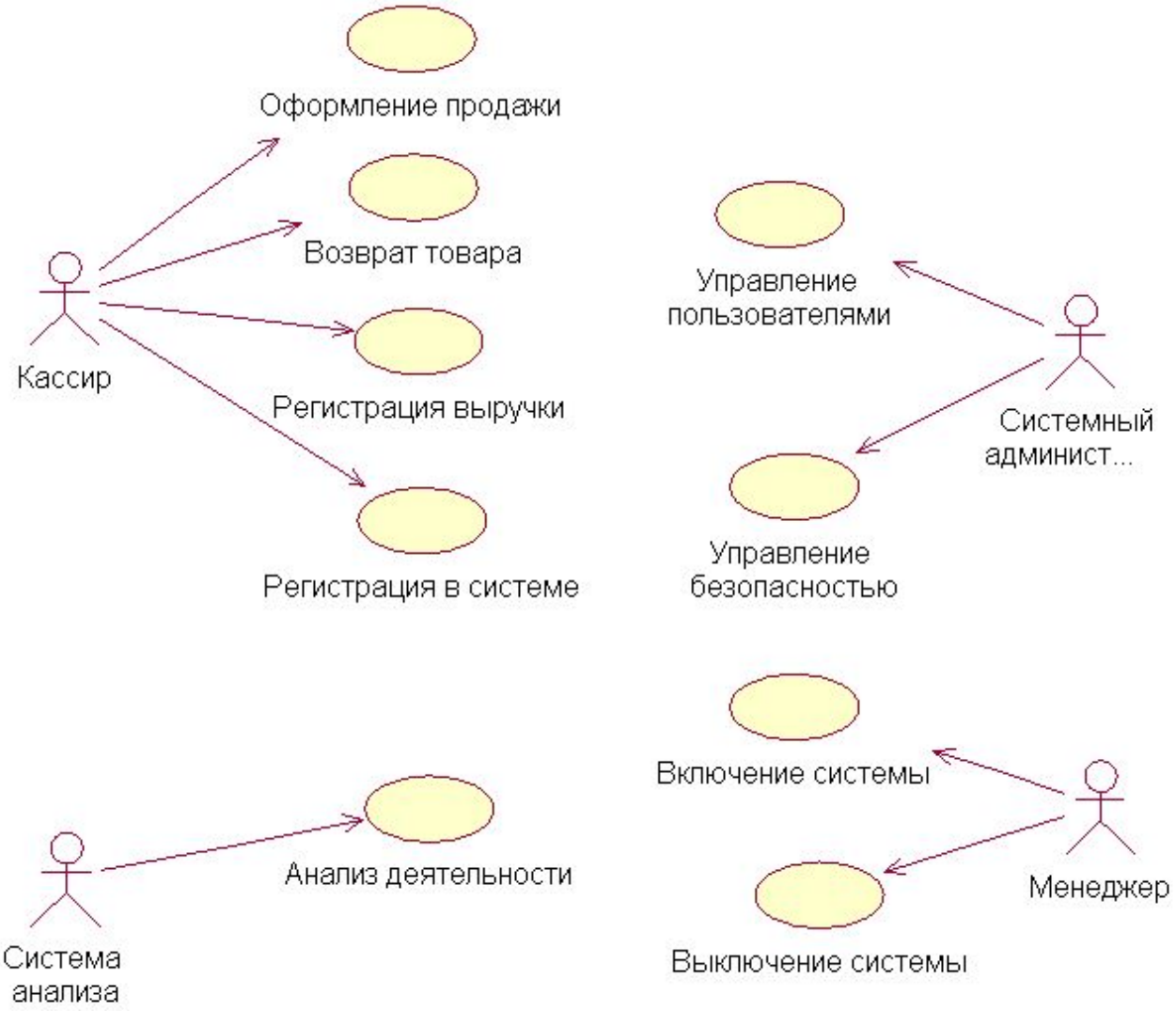
Ранжирование прецедентов

Ранг	Прецедент	Обоснование
Высокий	Оформление продажи	Основной бизнес-процесс
	Возврат товара	Влияет на архитектуру
Средний	Регистрация в системе	Влияет на безопасность
	Управление списком пользователей	Влияет на безопасность
	Управление безопасностью	Влияет на безопасность
	Анализ деятельности	Требует дополнительных исследований
Низкий	Включение системы	Влияние минимально
	Выключение системы	Влияние минимально
	Регистрация выручки	Влияние минимально

Функциональные требования

- Требования этой категории исследуются и формулируются в процессе разработки ***модели прецедентов (вариантов использования)***
- Как правило, одной задаче исполнителя соответствует один прецедент

Диаграмма прецедентов



Описание прецедентов

- Диаграмма прецедентов дает наглядное изображение системного контекста – границ системы, внешние по отношению к ней понятия и способы использования системы
- Однако, для формулирования и анализа требований необходимо детальное текстовое описание прецедентов

Описание прецедентов

- Текстовое описание прецедента может быть *развернутым* или *кратким*
- На начальном этапе развернутое описание дается лишь для *основных прецедентов* (10-20% от их общего числа)
- Пример развернутого описания для прецедента Оформление продажи

Диаграммы и описания прецедентов

- Важно иметь в виду, что главное в работе над прецедентами – это составление их описаний, основных и альтернативных сценариев, т.е. текстовых документов
- Диаграммы прецедентов играют важную, но второстепенную роль, помогая наглядно представить связи прецедентов с исполнителями, а также взаимосвязи прецедентов

Нефункциональные требования

- Определяются в дополнительной спецификации
- Приведем пример такой спецификации для POS-системы

Эргономичность

- Для достижения высокой скорости обслуживания покупателей при его высоком качестве необходимо:
  - обеспечить минимальное время отклика системы,
  - текст должен быть виден с расстояния 1 м,
  - не должно быть мерцания экрана,
  - предупреждающие сообщения должны сопровождаться звуковыми сигналами

Надежность

- При сбое в работе внешних систем (анализ деятельности) необходимо обеспечить возможность локальной обработки данных, их сохранение и последующую передачу
- Этот вопрос требует дальнейшей проработки

Производительность

- Покупатель хочет оформить покупку как можно быстрее
- Одна из основных причин задержки – низкая скорость авторизации
- Необходимо обеспечить выполнение авторизации менее, чем за 1 минуту в 90% случаев

Недостатки существующих решений

- Не обеспечивается автоматический переход из интерактивного в автономный режим при сбоях внешних систем;
- Отсутствие простой возможности интеграции с внешними системами;
- Отсутствие поддержки новых терминальных технологий

Итоги этапа Начало

- Выделены основные исполнители, задачи и прецеденты
- Выполнено ранжирование и описание прецедентов
- Сформулированы в черновом варианте требования к системе

Создается базовая архитектура системы
Производится разрешение высоких рисков
Определяется большинство требований (до 80% прецедентов получают развернутое описание)
Полностью разрабатывается некоторый фрагмент системы

ЭТАП РАЗВИТИЕ

Первая итерация

- Программная реализация базового сценария прецедента Оформление продажи
- Реализация прецедента Включение системы (необходим для предыдущего)
- Взаимодействие с внешними службами не реализуется

Словарь предметной области

- Для дальнейшей работы над системой требуется выделить основные сущности предметной области и зафиксировать их в словаре
- Register – реестр (терминал)
- Item – товар
- Store – магазин
- Sale – продажа
- Sales LineItem – элемент продажи

Словарь предметной области

- Cashier –кассир
- Customer – покупатель
- Manager – менеджер
- Payment – платеж
- Product Catalog – каталог товаров
- Product Specification – спецификация товара

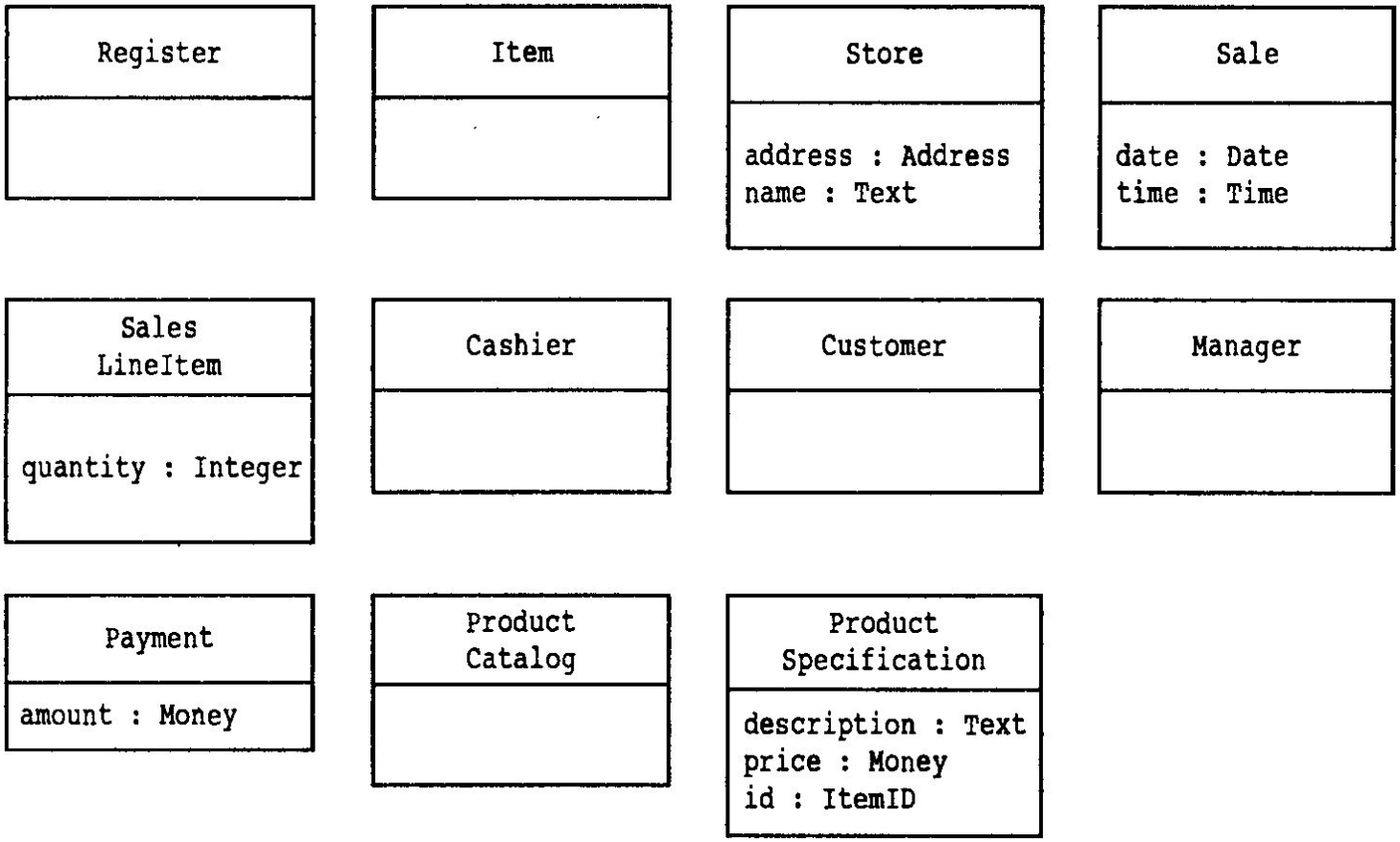
Концептуальная модель

- Выделенные таким образом сущности рассматриваются как *классы понятий* предметной области или ***концептуальные классы***
- Описание предметной области в терминах концептуальных классов называется ***концептуальной моделью*** предметной области

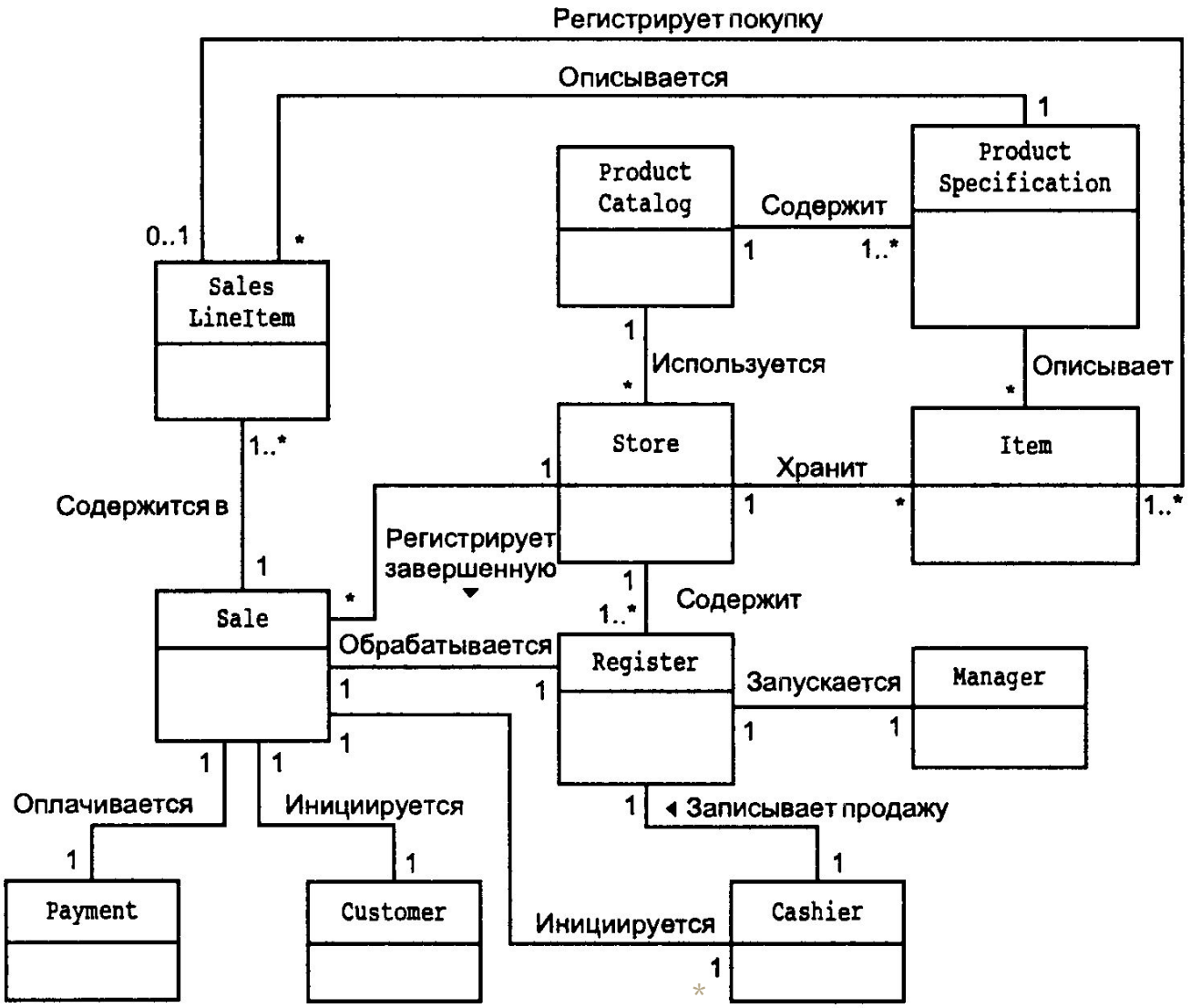
Концептуальная модель

- Отображает наиболее важные для цели моделирования классы понятий предметной области (концептуальные классы)
- Кроме того концептуальная модель может отображать
  - ассоциации между концептуальными классами,
  - атрибуты концептуальных классов

Классы и атрибуты



Концептуальная модель



Поведение системы

- Это описание действий, выполняемых системой, без детализации механизма их реализации
- Для визуального представления поведения системы используют *диаграмму последовательностей системы*

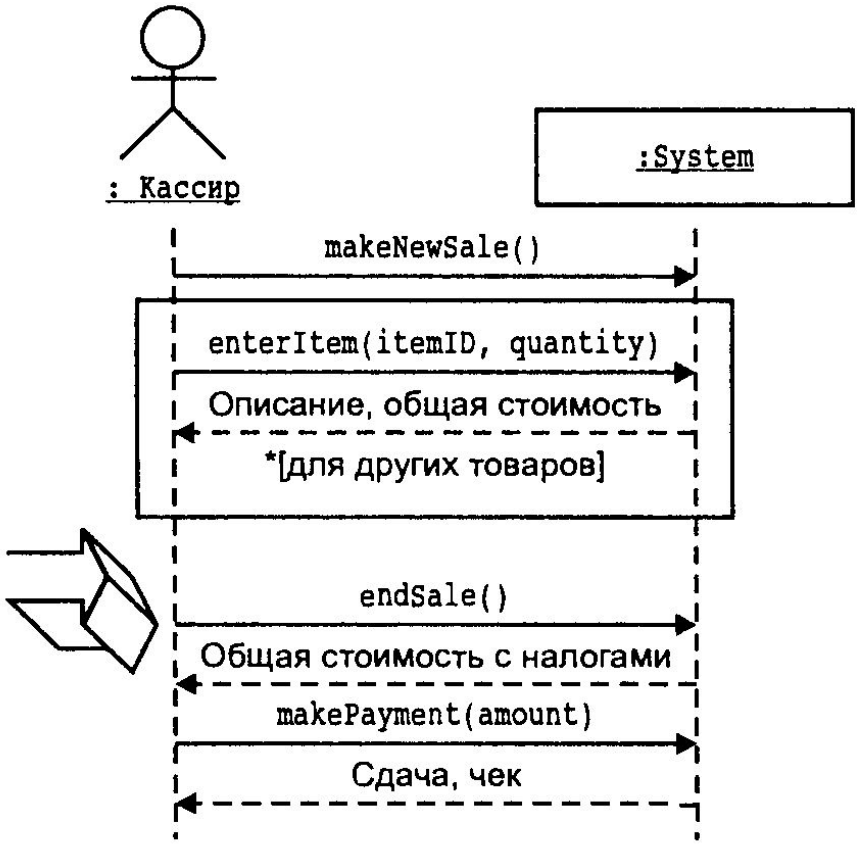
Диаграммы последовательностей

- Диаграммы последовательностей – это составная часть модели прецедентов, позволяющая визуализировать взаимодействие объектов в процессе реализации прецедента

Диаграмма последовательностей

Простой сценарий *Оформление продажи* с оплатой наличными.

1. Покупатель подходит к кассовому апперату POS-системы с выбранными товарами.
2. Кассир открывает новую продажу.
3. Кассир вводит идентификатор товара.
4. Система записывает идентификатор товара и выдает его описание, цену и общую стоимость.
Кассир повторяет действия, описанные в пп. 3-4, для каждого наименования товара.
5 . Система вычисляет общую стоимость покупки с налогом.
6. Кассир сообщает покупателю общую стоимость и предлагает оплатить покупку.
7. Покупатель оплачивает покупку, система обрабатывает платеж.
…



Системные операции

- Диаграмма последовательностей системы позволяет выделить набор ***системных операций***
- ***Операцией*** называется любое преобразование объекта или запрос к объекту
- Операция называется *системной*, если в качестве объекта выступает система в целом

Описание операций

- Операции требуют отдельного описания, если они достаточно сложны и их содержание не раскрыто в описании соответствующего прецедента

Структура описания операции

Операция	Имя операции и ее параметры
Ссылки (не обязательный раздел)	Прецеденты , в рамках которых может выполняться эта операция
Предусловия	Предположения о состоянии системы или объектов модели предметной области до начала выполнения операции. Эти предположения не проверяются при выполнении данной операции, а считаются истинными.
Постусловия	Состояние системы или объектов модели предметной области после выполнения операции.

Категории постусловий

- Создание или удаление экземпляра объекта
- Модификация атрибута экземпляра объекта
- Формирование или разрыв ассоциации

Системные операции POS

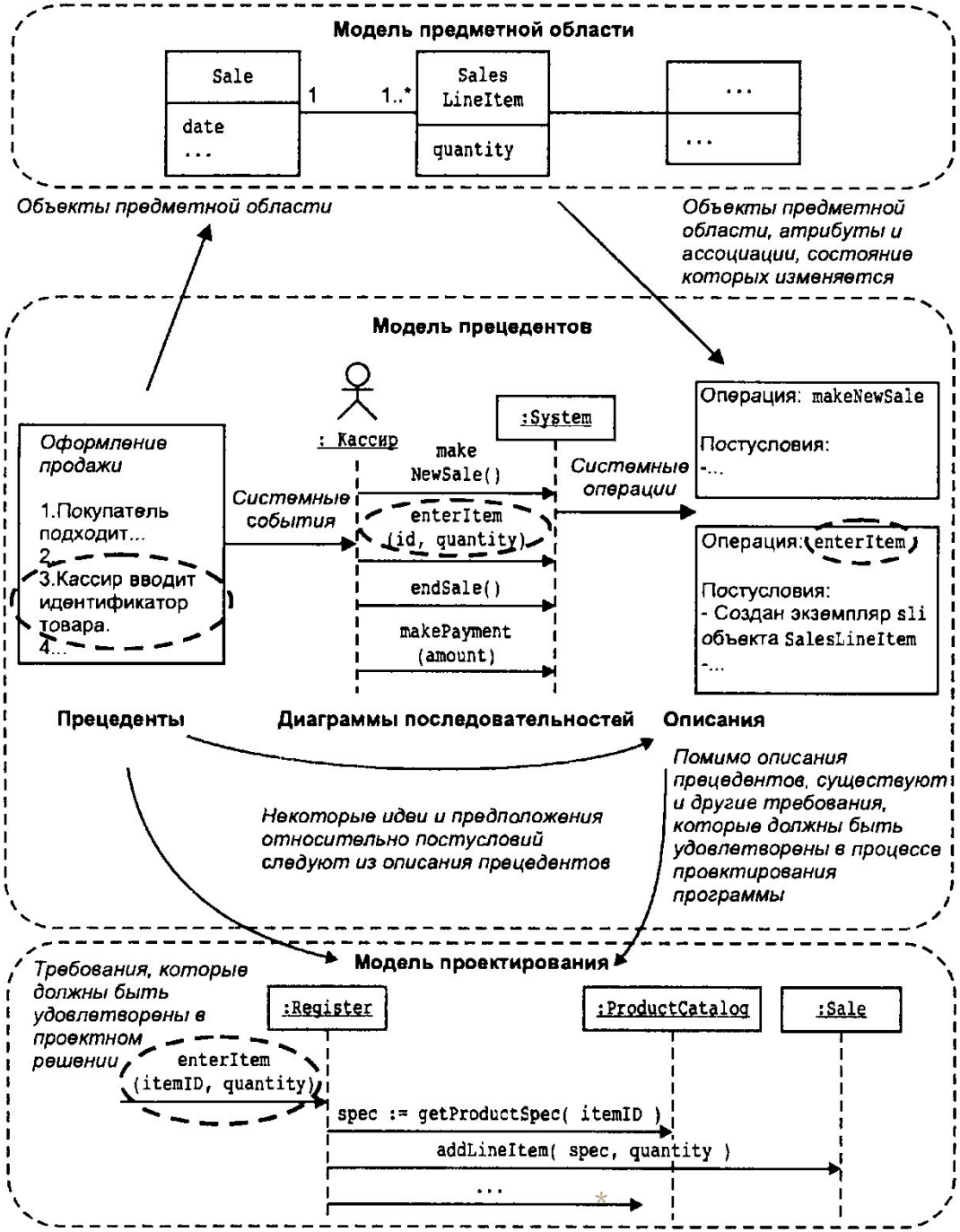
Операция	`makeNewSale()`
Ссылки	Прецеденты: `Оформление продажи`
Предусловия	Отсутствуют
Постусловия	- Создан экземпляр `s` объекта `Sale` (*создание экземпляра*) - Экземпляр `Sale` связан с объектом `Register` (*формирование ассоциации*) - Инициализированы атрибуты экземпляра `s`

Операция	`enterItem(itemID: ItemID, quantity: integer)`
Ссылки	Прецеденты: `Оформление продажи`
Предусловия	Инициирована продажа
Постусловия	- Создан экземпляр `sli` класса `SalesLineItem` (*создание экземпляра*) - Экземпляр `sli` связан с текущим экземпляром класса `Sale` (*формирование ассоциации*) - Атрибуту `sli.quantity` присвоено значение `quantity` (*модификация атрибута*) - Экземпляр `sli` связан с классом `ProductSpecification` на основе соответствия идентификатора товара `itemID` (*формирование ассоциации*)

Системные операции POS

Операция	`endSale()`
Ссылки	Прецеденты: `Оформление продажи`
Предусловия	Инициирована продажа
Постусловия	- Атрибут `Sale.iscomplete` принял значение `true` (*модификация атрибута*)

Операция	`makePayment(amount: Money)`
Ссылки	Прецеденты: `Оформление продажи`
Предусловия	Инициирована продажа
Постусловия	- Создан экземпляр `p` класса `Payment` (*создание экземпляра*) - Атрибут `p.amountTendered` принял значение `amount` (*модификация атрибута*) - Экземпляр `p` связан с текущим экземпляром класса `Sale` (*формирование ассоциации*) - Текущий экземпляр `Sale` связан с экземпляром класса `Store` для его добавления в журнал регистрации продаж (*формирование ассоциации*)

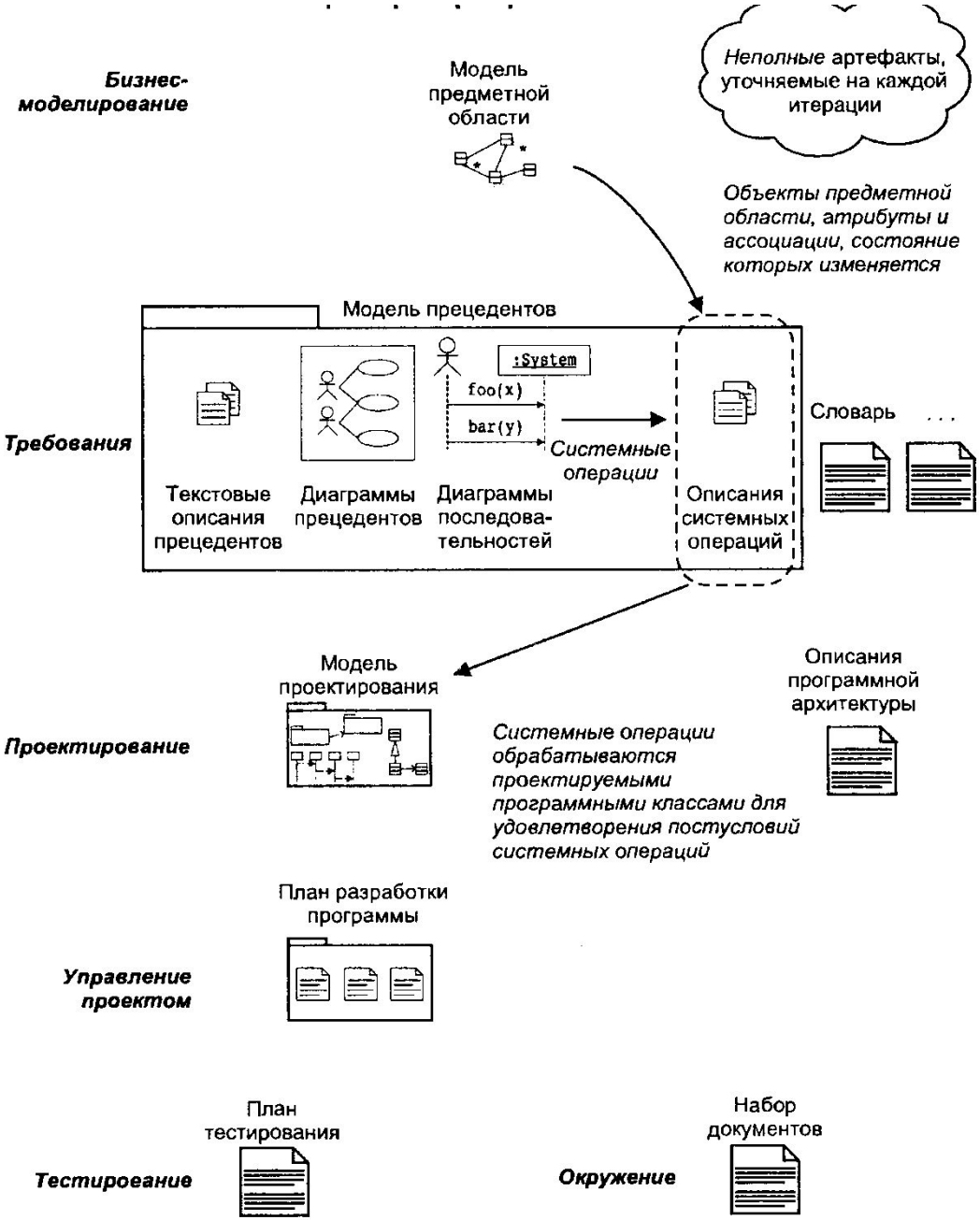

Модель предметной области
Sale
date
...
1
1..*
Sales LineItem
quantity
...
...
Объекты предметной области
Объекты предметной области, атрибуты и ассоциации, состояние которых изменяется
Модель прецедентов
Оформление продажи
1.Покупатель подходит...
2...
3.Кассир вводит идентификатор товара.
4...
: Кассир
:System
make NewSale()
enterItem (id, quantity)
endSale()
makePayment (amount)
Системные события
Системные операции
Операция: makeNewSale
Постусловия:
-...
Операция: enterItem
Постусловия:
- Создан экземпляр sli объекта SalesLineItem
-...
Прецеденты
Диаграммы последовательностей
Описания
Некоторые идеи и предположения относительно постусловий следуют из описания прецедентов
Помимо описания прецедентов, существуют и другие требования, которые должны быть удовлетворены в процессе проектирования программы
Модель проектирования
Требования, которые должны быть удовлетворены в проектном решении
:Register
:ProductCatalog
:Sale
enterItem (itemID, quantity)
spec := getProductSpec(itemID)
addLineItem(spec, quantity)
...
*

Модель проектирования

- Созданием концептуальной модели завершается анализ требований в рамках первой итерации
- На следующем этапе внимание фокусируется на разработке проектного решения, удовлетворяющего требованиям данной итерации, т.е. ***модели проектирования***

Концептуальные и программные классы

- Концептуальная модель содержит *концептуальные классы* с указанием их атрибутов
- Модель проектирования содержит *программные классы* с указанием их атрибутов и методов


Бизнес-моделирование
Модель предметной области
Неполные артефакты, уточняемые на каждой итерации
Объекты предметной области, атрибуты и ассоциации, состояние которых изменяется
Модель прецедентов
Требования
:System
foo(x)
bar(y)
Системные операции
Словарь
Текстовые описания прецедентов
Диаграммы прецедентов
Диаграммы последова-тельностей
Описания системных операций
Модель проектирования
Описания программной архитектуры
Проектирование
Системные операции обрабатываются проектируемыми программными классами для удовлетворения постусловий системных операций
План разработки программы
Управление проектом
План тестирования
Тестироеание
Набор документов
Окружение

Распределение обязанностей

- Основной задачей этапа проектирования является построение логики взаимодействия объектов, обеспечивающей выполнение системных требований
- Это достигается путем распределения *обязанностей* объектов

Знания и действия

- Обязанность определяется как контракт объекта и делятся на
  - *знания* (наличие информации об инкапсулированных данных, о связанных объектах)
  - *действия* (выполнение вычислений, создание экземпляра, инициирование действий других объектов или управление ими)

Реализация обязанностей

- Обязанности реализуются посредством методов программных классов
- Метод может реализовывать обязанность самостоятельно, либо во взаимодействии с методами других классов

Диаграммы взаимодействия

- Для визуализации распределения обязанностей между объектами используют *диаграммы взаимодействия* двух видов:
  - *диаграммы кооперации,*
  - *диаграммы последовательностей*
- В обоих случаях взаимодействие объектов представляется в виде обмена сообщениями

Шаблоны

- Распределение обязанностей подчиняется ряду принципов, обобщающих практический опыт проектирования программных систем
- Эти принципы формулируются в виде *шаблонов проектирования (**design patterns**)*

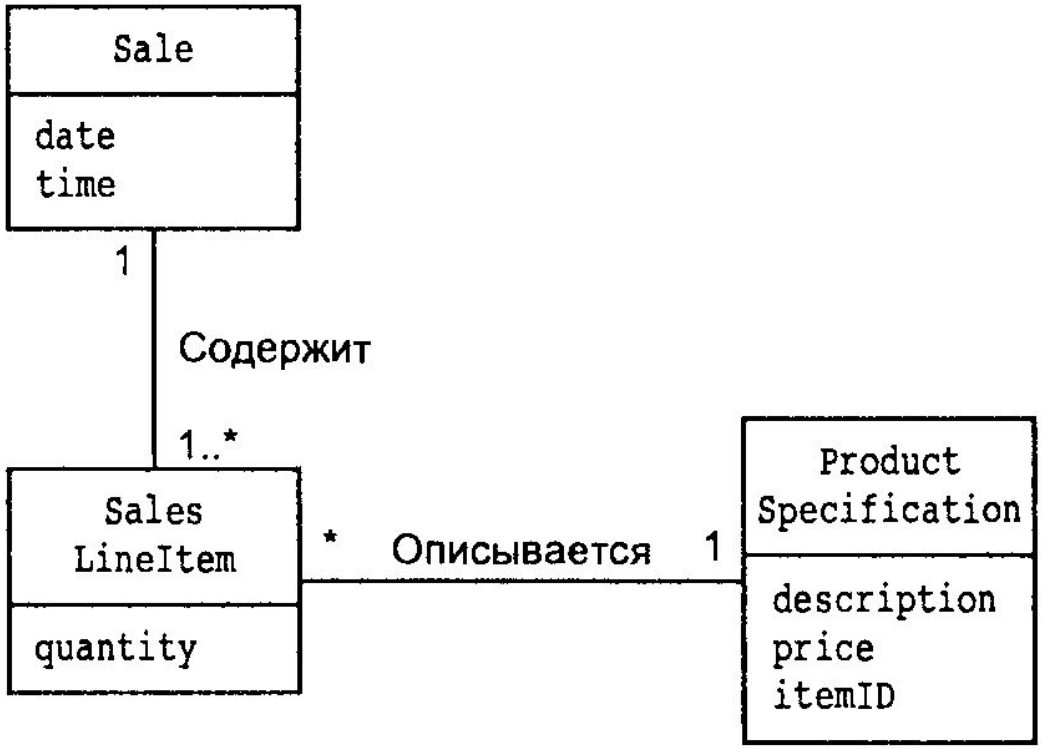
Шаблон Expert

- ***Проблема*** Каков наиболее общий принцип распределения обязанностей?
- ***Решение*** Назначить обязанность классу, владеющему информацией, необходимой для выполнения обязанности

Формулировка обязанности

- Вычислить общую сумму продажи
- Какая информация нужна для выполнения этой обязанности?
  - стоимость каждого вида товаров,
  - цену каждого вида товаров
- Какой класс должен выполнять эту обязанность?

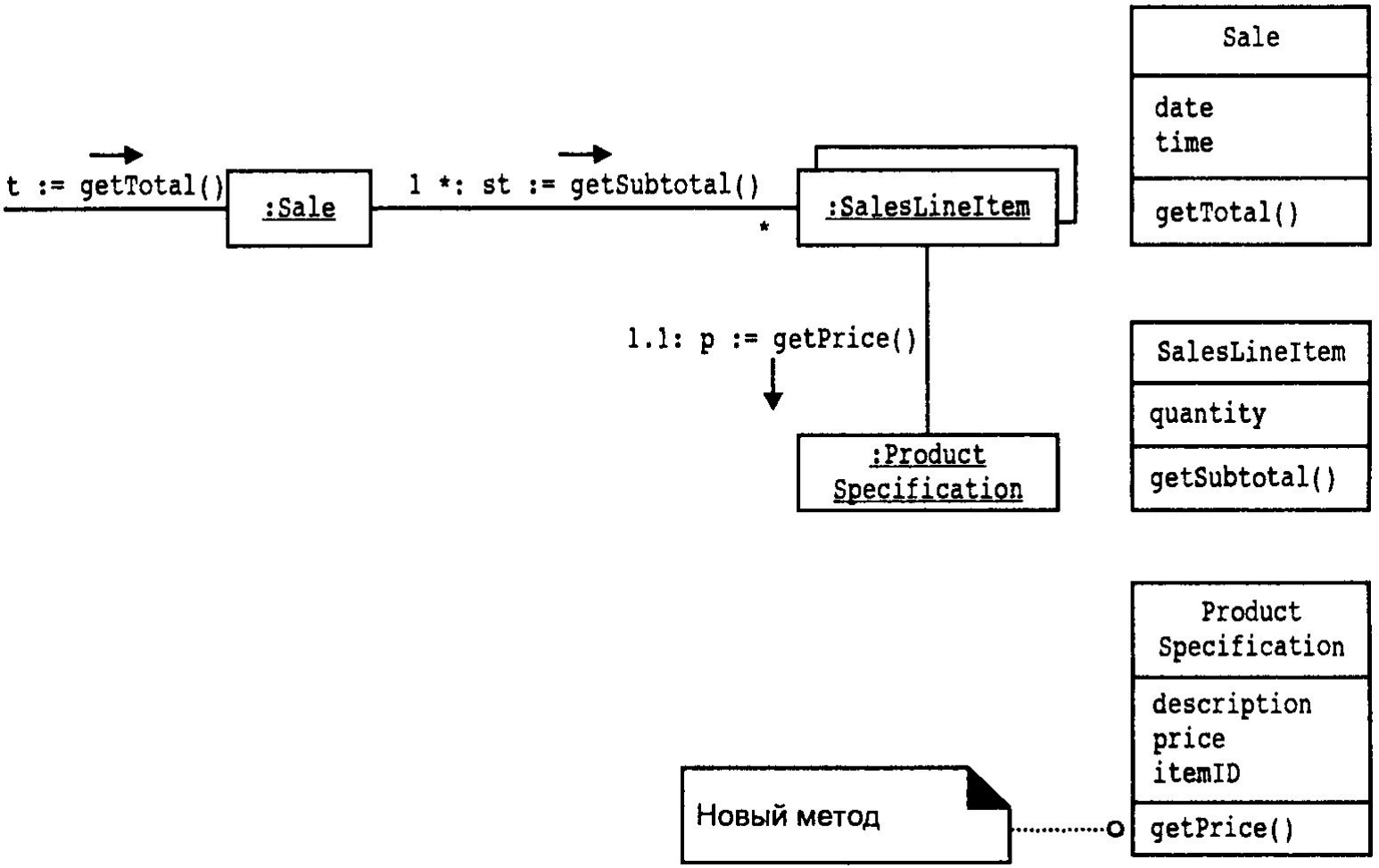
Вычисление общей стоимости



Распределение обязанностей

- Класс Sale – эксперт для вычисления общей суммы продажи
- Класс Sales LineItem– эксперт для вычисления промежуточной суммы элемента продажи
- Класс Product Specification – эксперт для определения цены товара

Диаграмма кооперации



Создание программных объектов

- Объекты программных классов должны быть созданы, чтобы их можно было использовать
- ***Проблема*** Какие классы должны отвечать за создание объектов классов Sale, Sales LineItem, Product Specification?

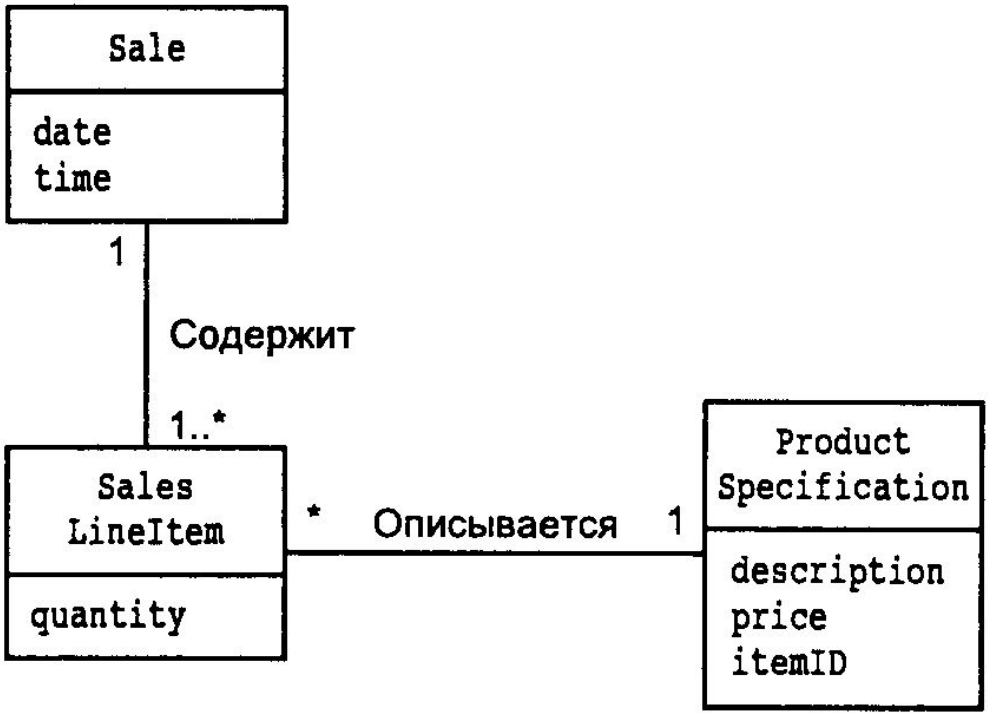
Шаблон Creator

- ***Решение*** Назначить классу B создавать объекты класса A, если выполняется одно из условий:
  - Класс B *агрегирует* объекты A
  - Класс B *содержит* объекты A
  - Класс B *записывает* объекты A
  - Класс B *активно использует* объекты A
  - Класс B *обладает данными для инициализации* объектов класса A

Шаблон Creator

- Под агрегацией классом B объектов класса A подразумевается, что последние являются *составляющими частями* объектов класса B
- Если же объект класса B выступает лишь в роли *контейнера* для объектов класса A, то говорят, класс B содержит объекты класса A

Выявление объекта-создателя



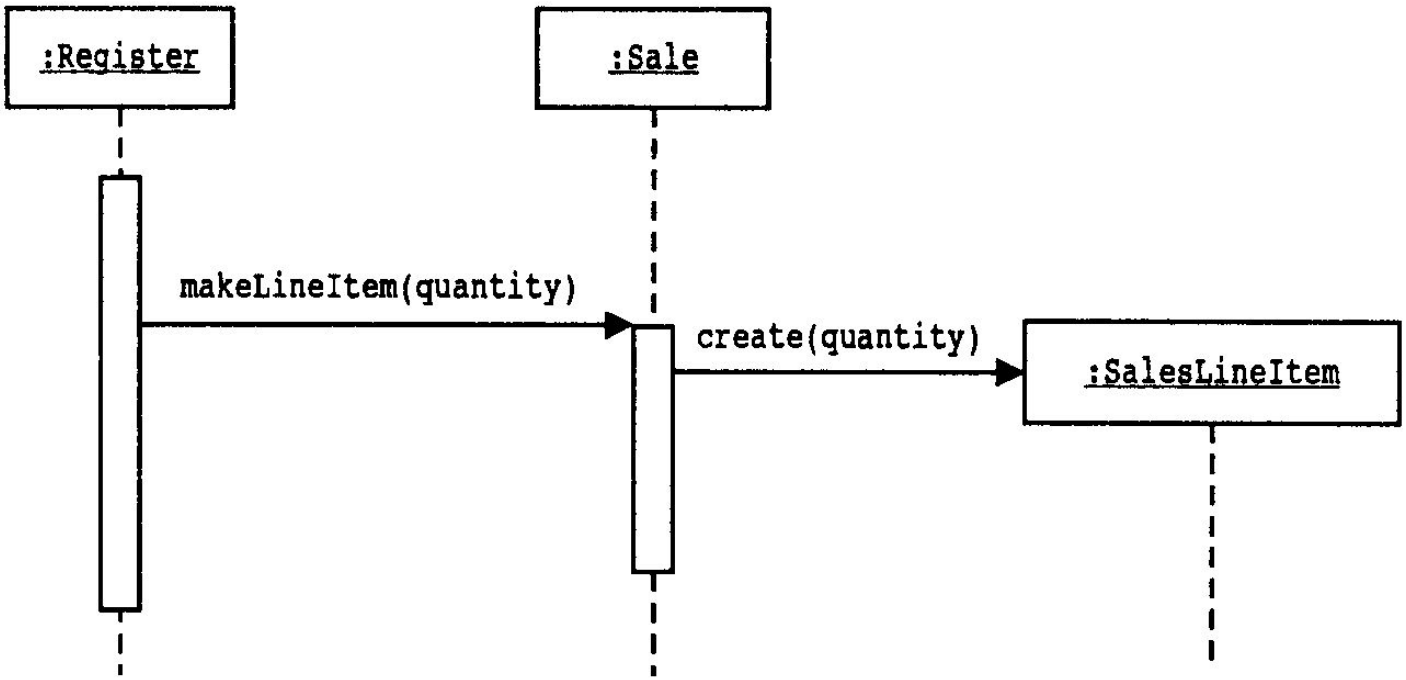
Шаблон Creator

- Определяет способ распределения обязанностей, связанный с процессом создания объектов
- Основное назначение – выявление объекта-создателя:
  - класс-контейнер
  - класс-регистратор
  - класс, владеющий информацией, необходимой при инициализации объекта

Создание объектов SalesLineItem

- Класс Sale агрегирует объекты класса SalesLineItem и поэтому является хорошим кандидатом на роль создателя объектов этого класса

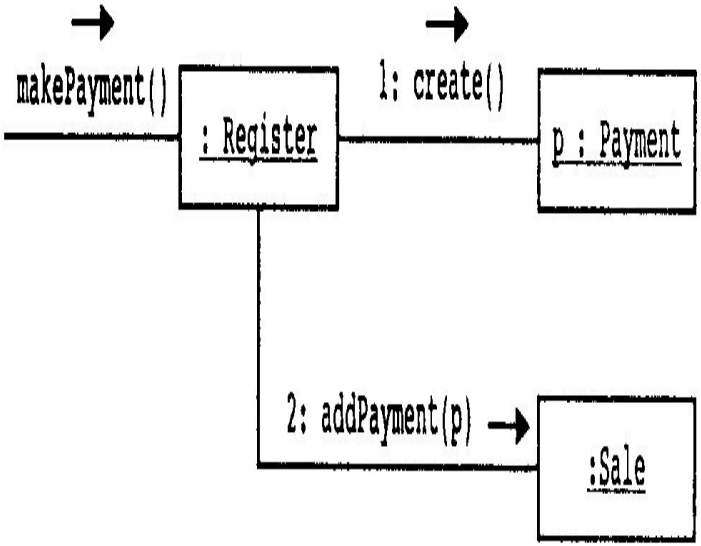
Диаграмма последовательностей

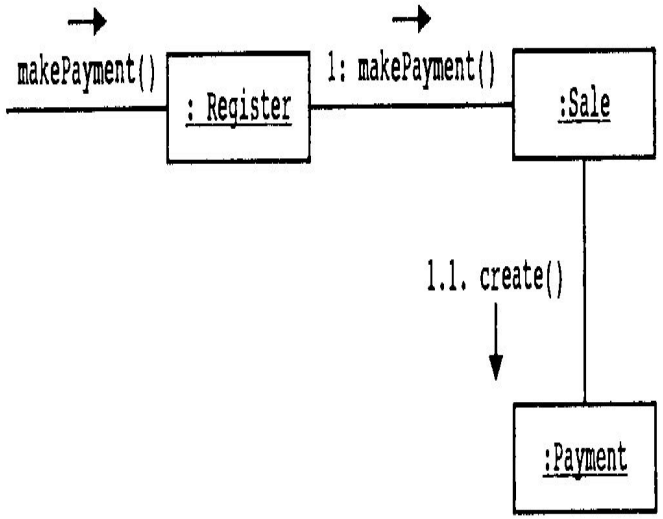


Обеспечение низкого сцепления

- Необходимо создать объект Payment и связать его с объектом Sale
- Возможны два альтернативных пути:
  - объект Payment создается объектом Register, который затем уведомляет об этом объект Sale;
  - объект Payment создается объектом Sale, который получает соответствующее указание от объекта Register

Два способа создания Payment





Шаблон Low Coupling

- Этот шаблон поддерживает независимость классов и слабое сцепление между ними
- В соответствии с данным шаблоном предпочтение следует отдать второму способу, т.к. при этом не возникает дополнительной связи между Register и Payment

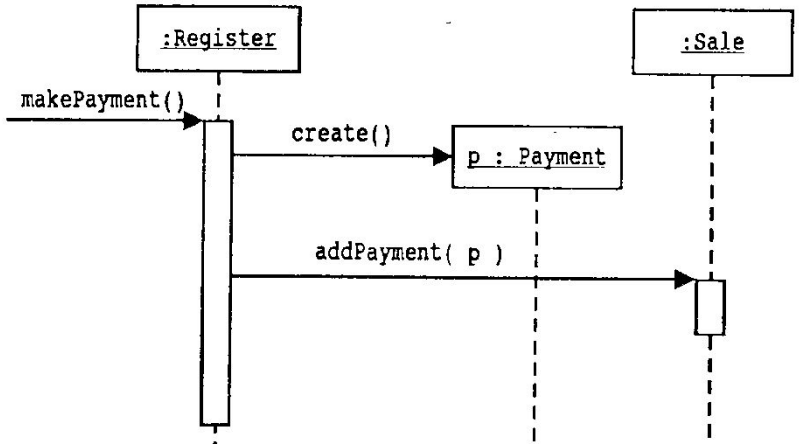
Шаблон Low Coupling

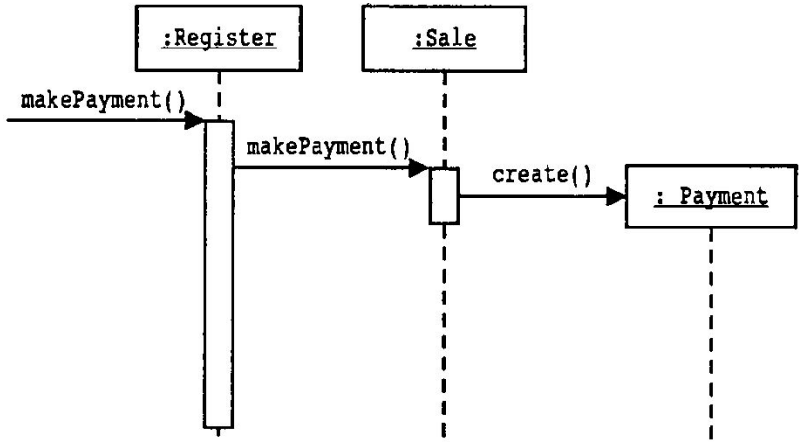
- Высокая степень сцепления объектов сама по себе не является проблемой
- Рекомендуется избегать ее в двух случаях:
  - для классов, являющихся достаточно общими по своей природе и многократно используемыми;
  - для неустойчивых и подверженными частому изменению элементов системы

Шаблон High Cohesion

- ***Проблема*** Как обеспечить управление сложностью?
- ***Решение*** Распределить обязанности способом, обеспечивающим высокую степень функциональной связности
- *Функциональная связность*– это мера взаимосвязи обязанностей класса
- Класс с низкой степенью связности выполняет много разнородных функций

Два способа создания Payment





Шаблон High Cohesion

- Классы с высокой степенью связности просты в понимании, поддержке и повторном использовании
- Сцепление и связность взаимозависимы: неправильное сцепление порождает слабую связность, и наоборот

Конец лекции