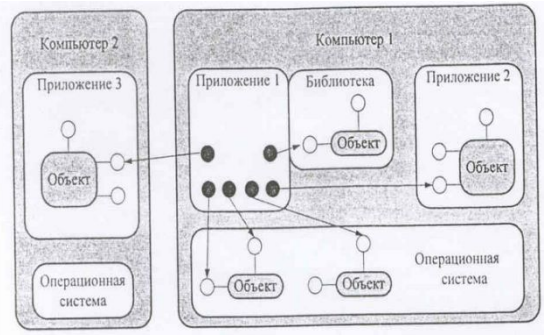
ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ ОСНОВНЫЕ ПОНЯТИЯ И ПОДХОДЫ

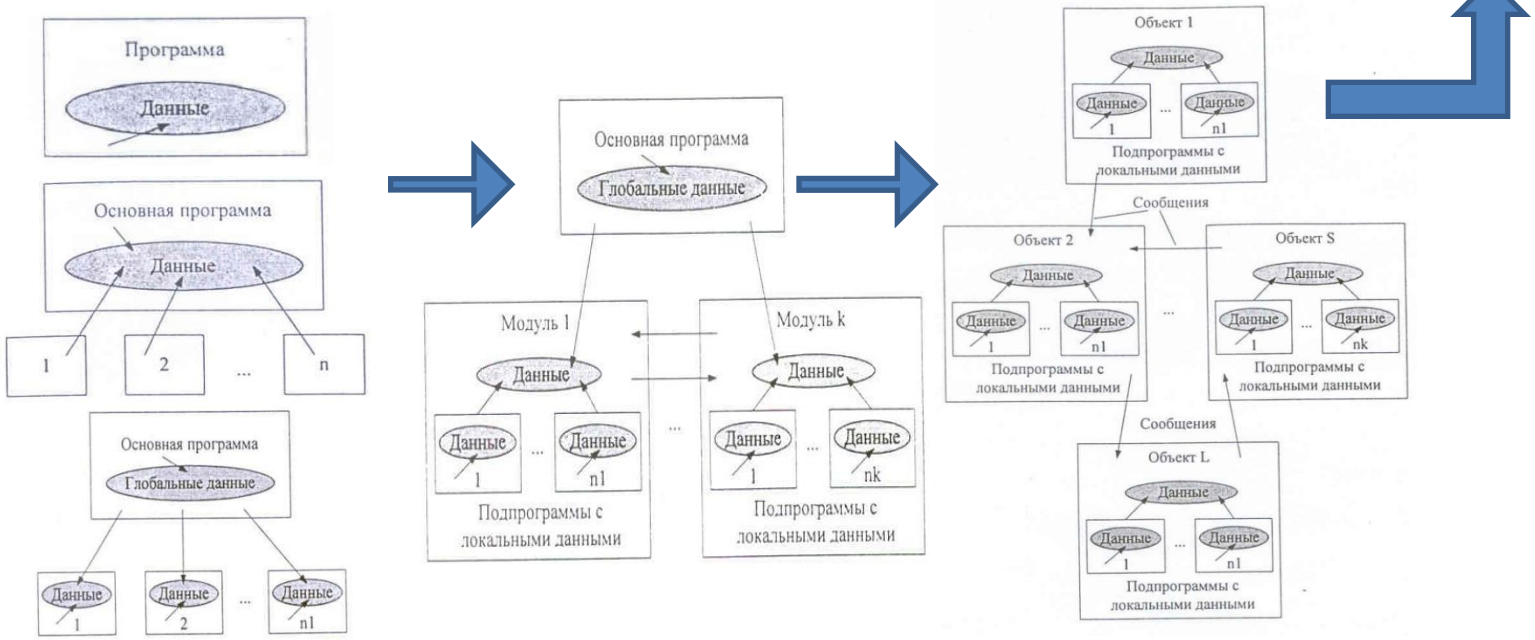
Методические материалы, инструкции, нормативы и стандарты, критерии оценки результатов

Исходные данные в стандартном представлении (документы, рабочие материалы, результаты предыдущей операции)

Результаты в стандартном представлении



Исполнители, программные и технические средства



Принципы работы со сложными системами

- Абстракция (abstraction) и уточнение (refinement).

-Модульность (modularity):

Примером разбиения на модули может служить структура пакетов и классов библиотеки JDK.

1. Классы, связанные с основными сущностями языка Java и виртуальной машины, собраны в пакете java.lang.
2. Вспомогательные широко применяемые в различных приложениях классы, такие, как коллекции, представления даты и пр., собраны в java.util.
3. Классы, используемые для реализации потокового ввода-вывода — в пакете java.io, и т.д.

Интерфейсом класса служат его общедоступные методы, а интерфейсом пакета — его общедоступные классы,

-Переиспользования.

Технологии Java

Независимость от платформы

- Исходные тексты хранятся в текстовом виде в файле .java
- Файл .java компилируется в файл .class
- Этот файл содержит байт-код (инструкции для выполнения интерпретатором)
- Байт-код интерпретируется во время выполнения

```
package ru.vsu.test;
import java.util.Date;
public class FirstProgram {
        private Date today;
        public Date getToday(){
                return today;
        }
        public void setToday(Date aToday){
                today = aToday;
        }
        public static void main (String[] args){
                FirstProgram fp = new FirstProgram();
                fp.setToday(new Date());
                System.out.println (fp.getToday());
        }
}
```

Варианты поставки

- J2ME (Micro Edition) – для мобильных устройств
- J2SE (Standard Edition) – разработка обычных приложений
- J2EE (Enterprise Edition) – разработка приложений многозвенной архитектуры

ЖИЗНЕННЫЙ ЦИКЛ И ПРОЦЕССЫ РАЗРАБОТКИ ПО

Жизненным циклом программного обеспечения называют период от момента появления идеи создания некоторого программного обеспечения до момента завершения его поддержки фирмой-разработчиком или фирмой, выполнявшей сопровождение.

Стандарты жизненного цикла

- IEEE — читается «ай-трипл-и», Institute of Electrical and Electronic Engineers, Институт инженеров по электротехнике и электронике;
- ISO — International Standards Organization, Международная организация по стандартизации;

ISO/IEC 12207 Standard for Information Technology — Software Life Cycle Processes

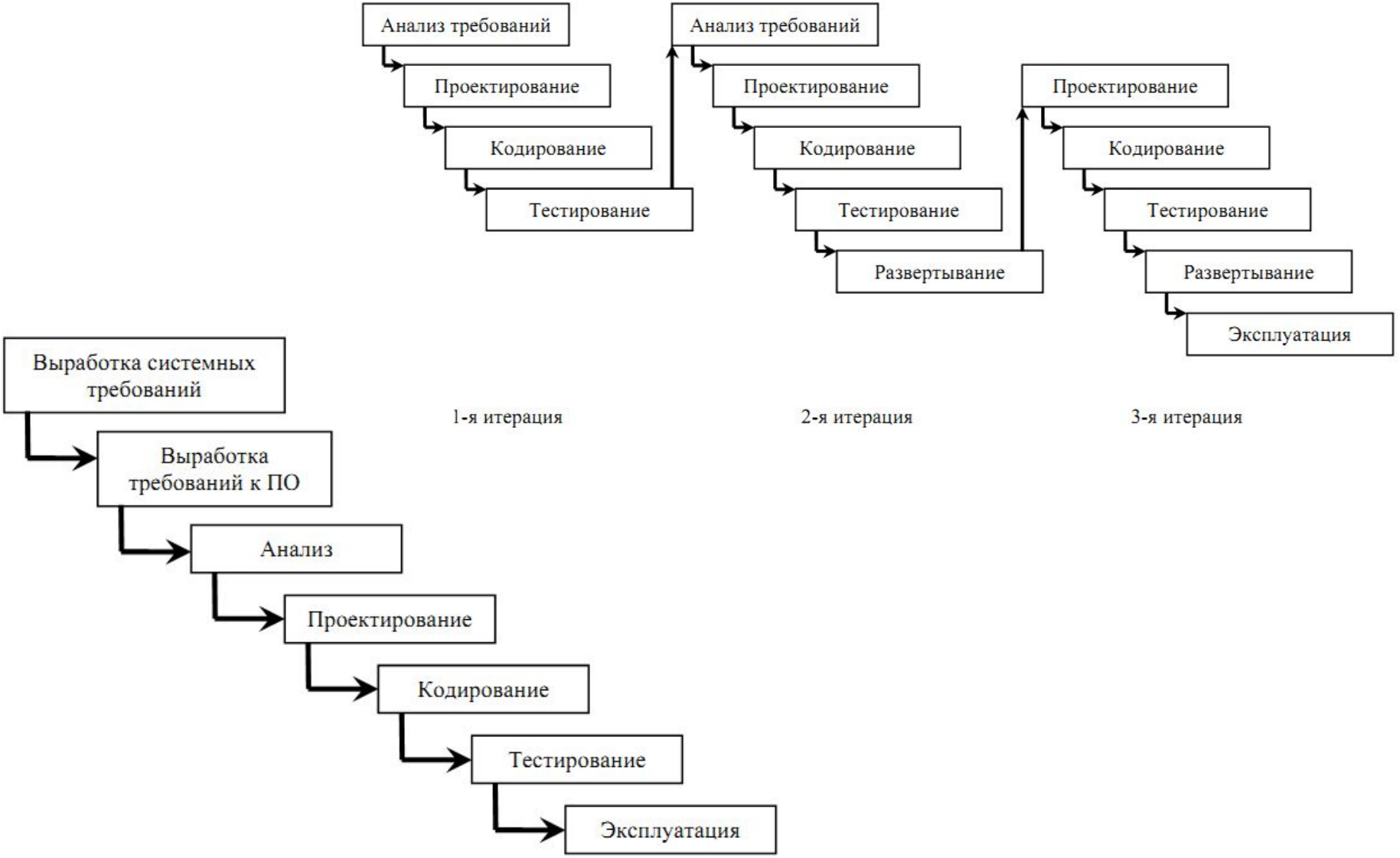
Основные процессы	Поддерживающие процессы	Организационные процессы	Адаптация
Приобретение ПО; Передача ПО (в использование); Разработка ПО; Эксплуатация ПО; Поддержка ПО	Документирование; Управление конфигурациями; Обеспечение качества; Верификация; Валидация; Совместные экспертизы; Аудит; Разрешение проблем	Управление проектом; Управление инфраструктурой; Усовершенствование процессов; Управление персоналом	Адаптация описываемых стандартом процессов под нужды конкретного проекта

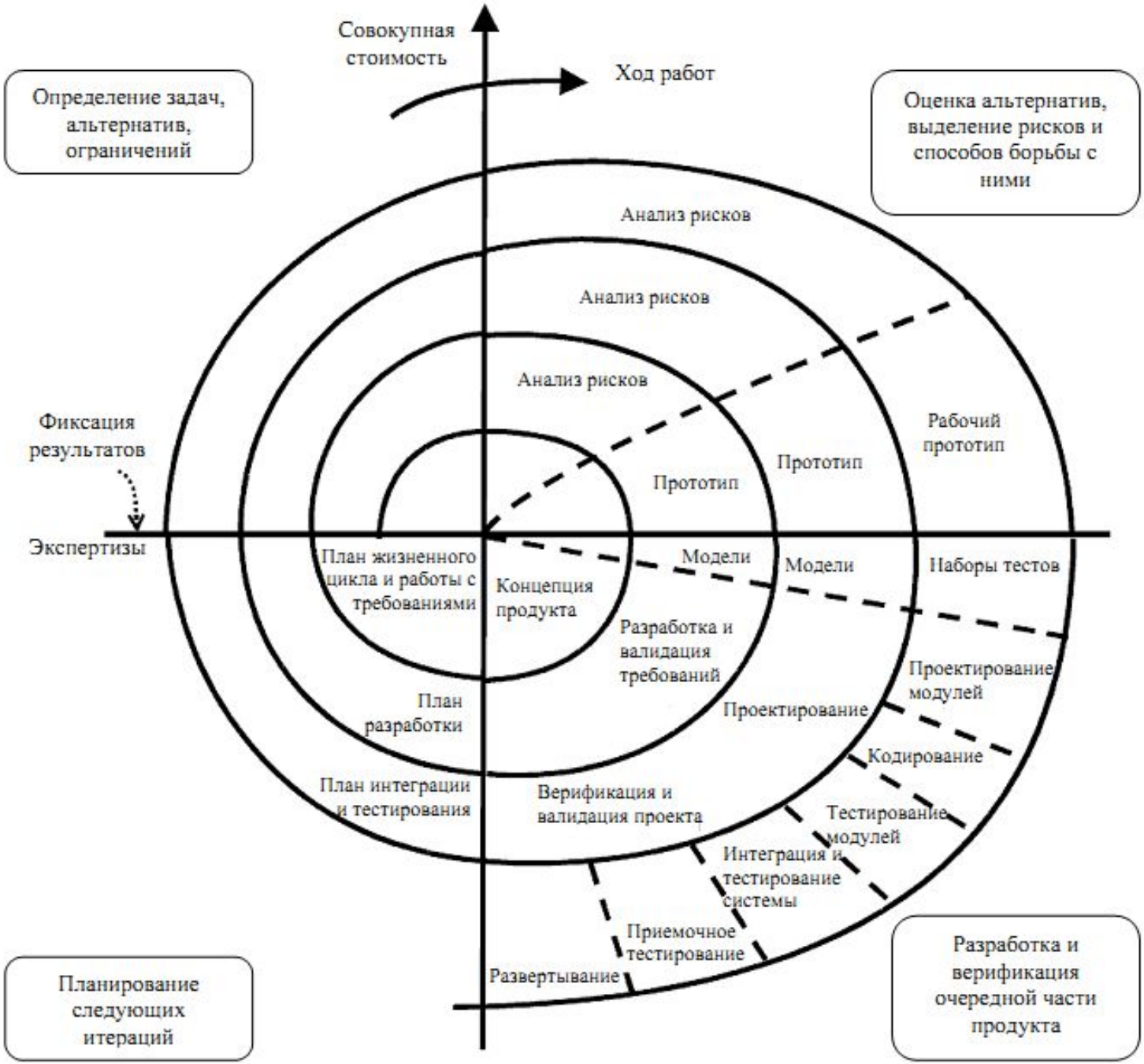
ISO/IEC 15288 Standard for Systems Engineering — System Life Cycle Processes (процессы жизненного цикла систем). Отличается от предыдущего нацеленностью на рассмотрение программно-аппаратных систем в целом.

ISO/IEC 15504 (SPICE) Standard for Information Technology — Software Process Assessment (оценка процессов разработки и поддержки ПО).

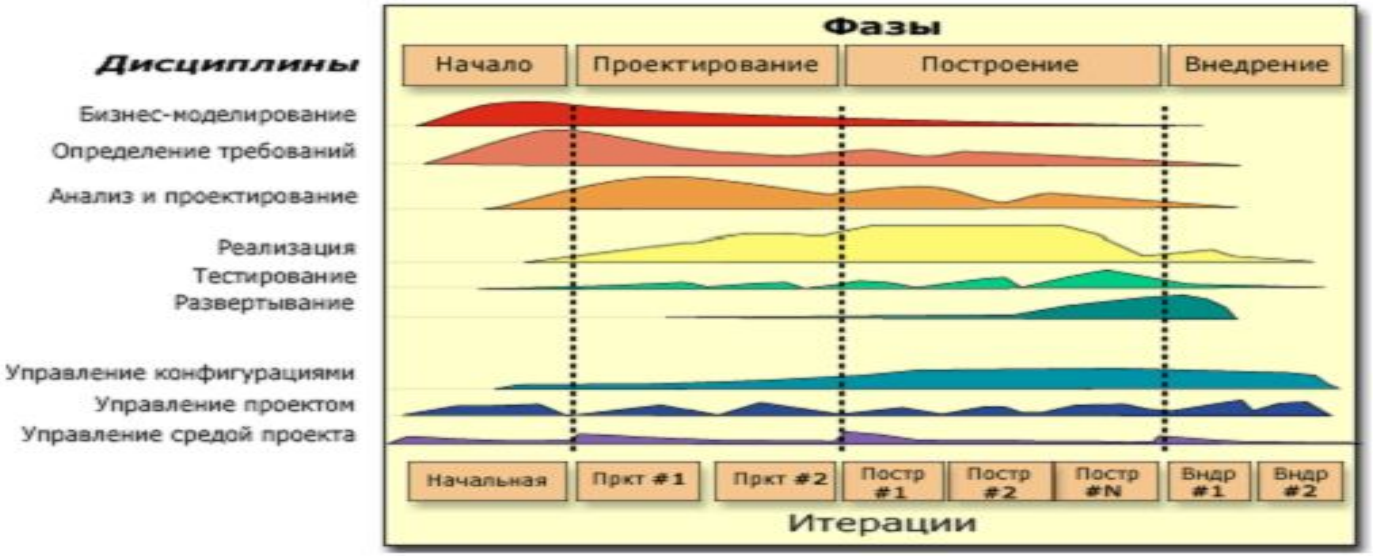
IEEE 1074-1997 — IEEE Standard for Developing Software Life Cycle Processes (стандарт на создание процессов жизненного цикла ПО).

Модели жизненного цикла




Совокупная стоимость
Ход работ
Определение задач, альтернатив, ограничений
Оценка альтернатив, выделение рисков и способов борьбы с ними
Анализ рисков
Анализ рисков
Анализ рисков
Фиксация результатов
Экспертизы
Прототип
Прототип
Рабочий прототип
План жизненного цикла и работы с требованиями
Концепция продукта
Модели
Модели
Наборы тестов
Разработка и валидация требований
План разработки
Проектирование
Проектирование модулей
Кодирование
План интеграции и тестирования
Верификация и валидация проекта
Тестирование модулей
Интеграция и тестирование системы
Приемочное тестирование
Развертывание
Планирование следующих итераций
Разработка и верификация очередной части продукта

УНИФИЦИРОВАННЫЙ ПРОЦЕС RATIONAL (RATIONAL UNIFIED PROCESS, RUP) И ЭКСТРЕМАЛЬНОЕ ПРОГРАММИРОВАНИЕ (EXTREME PROGRAMMING, XP)



ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1. В разделе "Наименование и область применения" указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.
2. В разделе "Основание для разработки" должны быть указаны: документ (документы), на основании которых ведется разработка; организация, утвердившая этот документ, и дата его утверждения; наименование и (или) условное обозначение темы разработки.
3. В разделе " Назначение разработки" должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.
4. Раздел "Технические требования к программе или программному изделию" должен содержать следующие подразделы:

- требования к функциональным характеристикам; состав выполняемых функций, организации входных и выходных данных, временные характеристики и т.п.
- требования к надёжности (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т.п.);
- условия эксплуатации (интерфейс, а также вид обслуживания, необходимое количество и квалификация персонала.)
- требования к составу и параметрам технических средств; состав технических средств с указанием их технических характеристик
- требования к информационной и программной совместимости; (решения, исходные коды, языки программирования)
- требования к упаковке; требования к транспортированию и хранению; специальные требования.

5. В разделе "Технико-экономические показатели" должны быть указаны: экономическая эффективность, предполагаемая годовая потребность, преимущества разработки по сравнению с аналогами.
6. В разделе "Стадии и этапы разработки" устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также, как правило, сроки разработки и определяют исполнителей.
7. В разделе "Порядок контроля и приёмки" должны быть указаны виды испытаний и общие требования к приёмке работы.

ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению»
ГОСТ 34.602-89: Техническое задание на создание системы
IEEE Std 830—1998 IEEE Recommended Practice for Software Requirements Specifications

ПРИМЕР

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Программа разрабатывается на основе учебного плана кафедры «Информационные технологии проектирования» от 5.09.2013.

3. НАЗНАЧЕНИЕ

Основным назначением программы является ознакомление пользователя с работой алгоритма Дейкстры .

4.ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1.Т р е б о в а н и я к ф у н к ц и о н а л ь н ы м х а р а к т е р и с т и к а м

4.1.1. Программа должна обеспечивать возможность выполнения следующих функций:

• ввод аналитического представления функции одной переменной и длительное хранение его в системе;

• ввод и изменение интервала определения функции;

• ввод и корректировку шага аргумента;

• построение таблицы значений функции на заданном интервале иди изображение графика функции на заданном интервале при условии, что на указанном интервале она не имеет точек разрыва.

4.1.2. Исходные данные:

• аналитическое задание функции;

• интервал определения функции;

• шаг изменения аргумента, определяющий количество точек на интервале.

4.2. Т р е б о в а н и я к н а д е ж н о с т и

4.2.1.Предусмотреть контроль вводимой информации.

4.2.2.Предусмотреть блокировку некорректных действий пользователя при работе с системой.

4.3. Т р е б о в а н и я к с о с т а в у и п а р а м е т р а м технических средств

4.3.1.Система должна работать на IBM совместимых персональных компьютерах.

4.3.2.Минимальная конфигурация:

• тип процессора.....Pentium и выше;

• объем оперативного запоминающего устройств .....32 Мб и более.

4.4. Требования к информационной и программной совместимости

Система должна работать под управлением операционной системы Windows'95 и выше.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1.Разрабатываемая система должна включать справочную информацию о работе системы и подсказки пользователю.

5.2.В состав сопровождающей документации должны входить:

• пояснительная записка;

 руководство пользователя.

№	Название этапа	Срок	Отчетность
1	Разработка ядра системы	1.1.2000 – 31.3.2000	Описание внутренних форматов, интерфейса и форматов данных базы. Реализация системы на уровне интерфейса
2	Разработка методов и алгоритмов и их реализация для задачи коммивояжера	1.4.2000 – 30.6.2000	Описание методов и алгоритмов. Программные модули, реализующие методы
3	Разработка методов и алгоритмов и их реализация для задачи построения минимального связывающего дерева и задачи поиска кратчайшего пути в графе	1.7.2000 – 30.9.2000	Описание методов и алгоритмов. Программные модули, реализующие методы
4	Тестирование программного продукта и составление программной документации	1.10.2000 – 31.12.2000	Тесты. Документация. Программный продукт

СХЕМА ЗАХМАНА ИЛИ АРХИТЕКТУРНАЯ СХЕМА ПРЕДПРИЯТИЯ

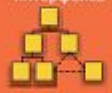
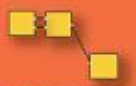
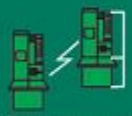
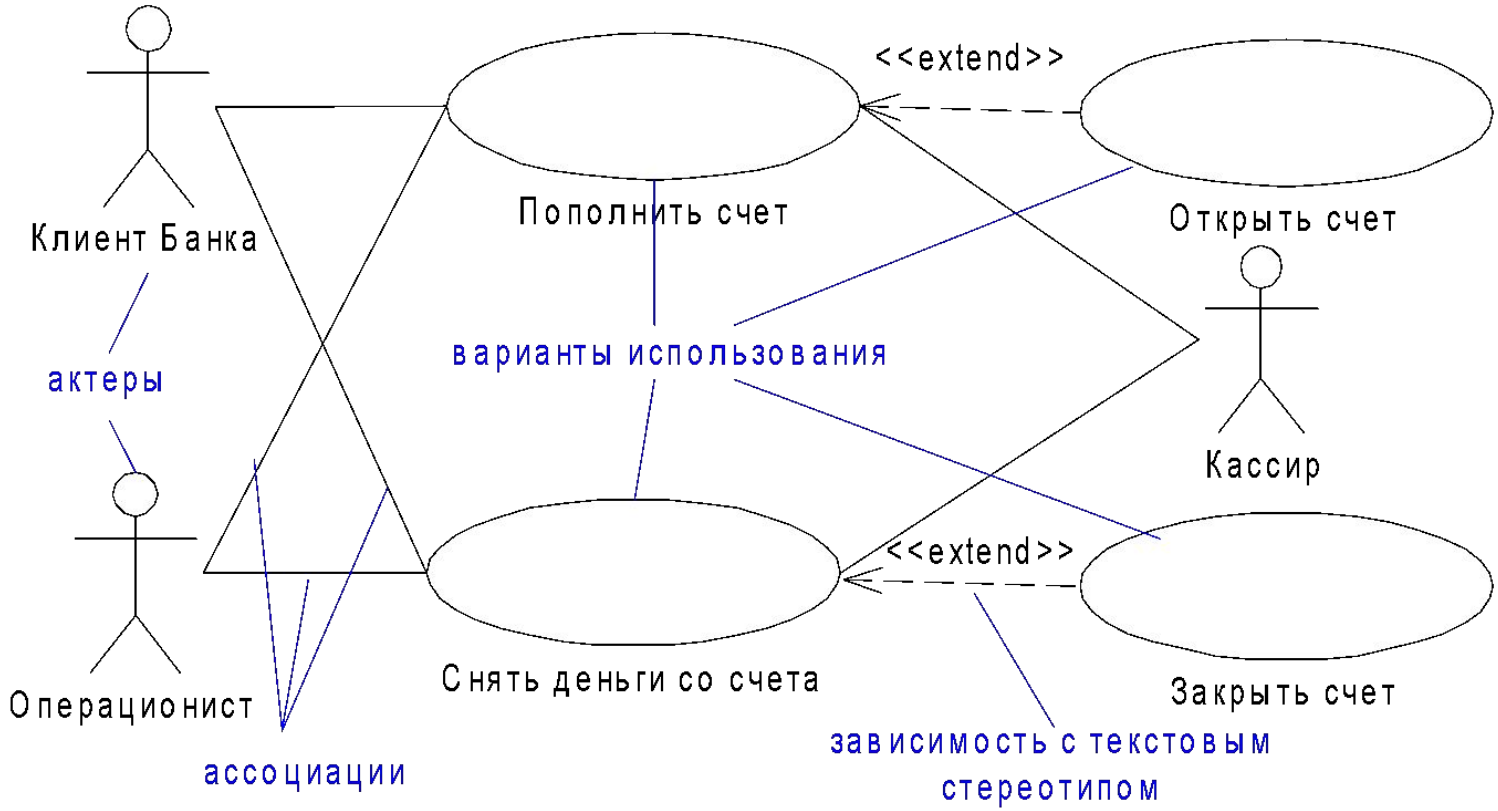
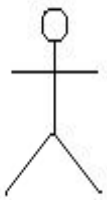
	Мотивация	Люди	Данные	Функции	Место	Время
Контекст	Цели и стратегия бизнеса 	Важные для бизнеса организации 	Вещи, значимые для бизнеса 	Основные бизнес-процессы 	География бизнеса 	События и периоды, важные для бизнеса 
Модель бизнеса	Бизнес план, частные цели и стратегии 	Модели потоков работ 	Семантические модели  Бизнес-сущности и их связи	Модели бизнес-процессов 	Система логистики 	Базовый график работ 
Системная модель	Модель бизнес-правил 	Архитектура пользовательского интерфейса 	Концептуальная модель данных 	Архитектура приложений 	Архитектура распределенной системы 	Структура обработки событий 
Технологическая модель	Модель правил обработки событий 	Архитектура представления 	Физическая модель данных 	Архитектура программно-аппаратной системы 	Технологическая архитектура 	Структура циклов управления 
Детальное представление	Спецификации правил работы системы 	Спецификации ролей и прав доступа 	Спецификации форматов данных 	Код программных компонентов 	Спецификации архитектуры сети 	Спецификации обработки событий и прерываний 
Работающая организация	Стратегия и тактика	Структура организации	Данные	Выполняемые функции	Географическое расположение и сети	Планы

Диаграмма вариантов использования *(use case diagram)*

- диаграмма, на которой изображаются варианты использования проектируемой системы, заключенные в границу системы, и внешние актеры, а также определенные отношения между актерами и вариантами использования



Основные обозначения на диаграмме вариантов использования



actor

communication association

generalization

«extend»

extend

use case

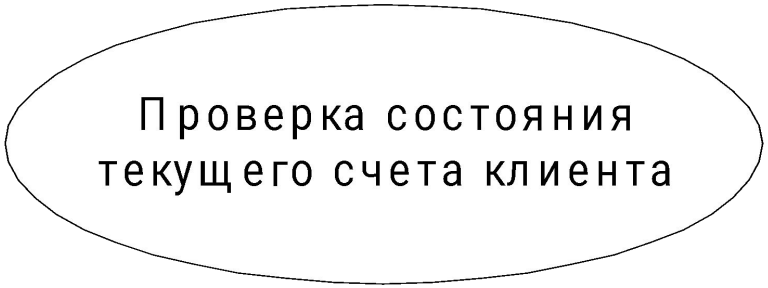
«include»

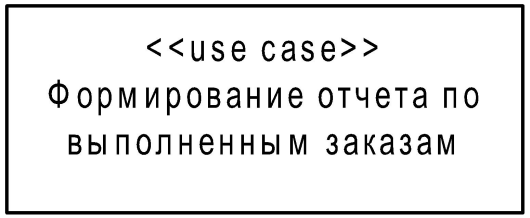
include

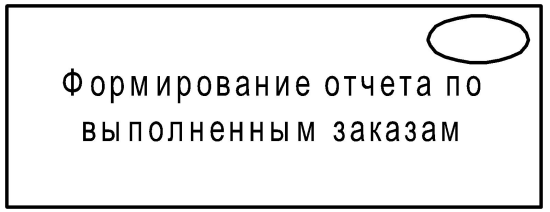
system boundary

Вариант использования (use case)

- – представляет собой общую спецификацию совокупности выполняемых системой действий с целью предоставления некоторого наблюдаемого результата, который имеет значение для одного или нескольких актеров
- Отвечает на вопрос «Что должна выполнять система?», не отвечая на вопрос «Как она должна выполнять это?»
- Имена – отглагольное существительное или глагол в неопределенной форме

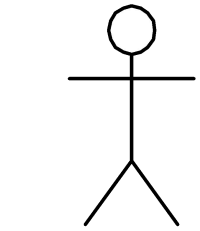




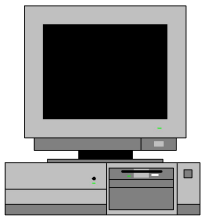


Актер (actor)

- – любая внешняя по отношению к проектируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач
- *Примеры актеров*: кассир, клиент банка, банковский служащий, президент, продавец магазина, менеджер отдела продаж, пассажир авиарейса, водитель автомобиля, администратор гостиницы, сотовый телефон



Клиент банка



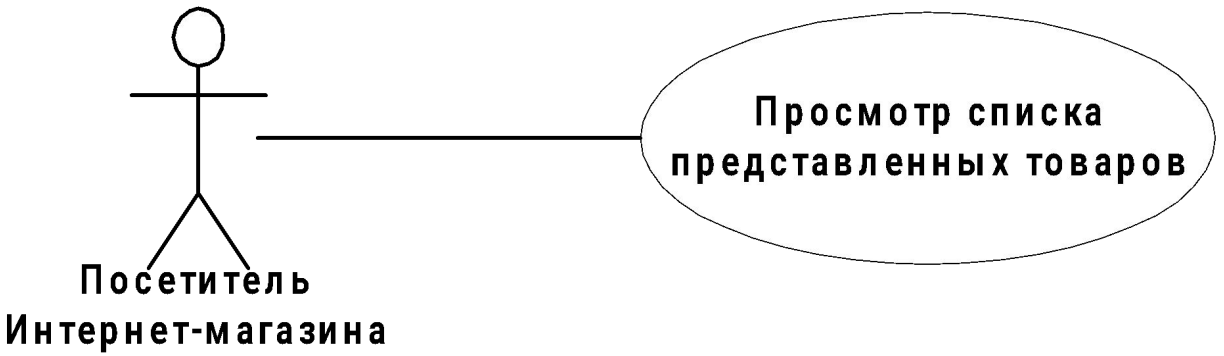
Удаленный пользователь

Вопросы для идентификации актеров в системе

- Какие организации или лица будут использовать систему
- Кто будет получать пользу от использования системы
- Кто будет использовать информацию от системы
- Будет ли использовать система внешние ресурсы
- Может ли один пользователь играть несколько ролей при взаимодействии с системой
- Могут ли различные пользователи играть одну роль при взаимодействии с системой
- Будет ли система взаимодействовать с законодательными, исполнительными, налоговыми или другими органами

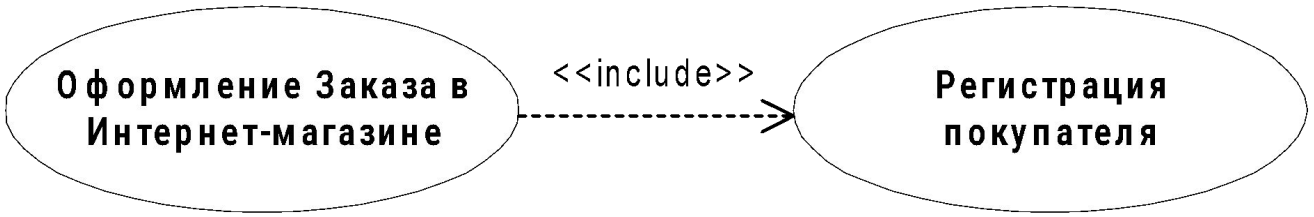
Отношение ассоциации

- *Ассоциация* (association) является одним из фундаментальных понятий в языке UML 2.x и может использоваться на различных канонических диаграммах при построении визуальных моделей
- Применительно к диаграммам вариантов использования отношение ассоциации может служить только для обозначения взаимодействия актера с вариантом использования.



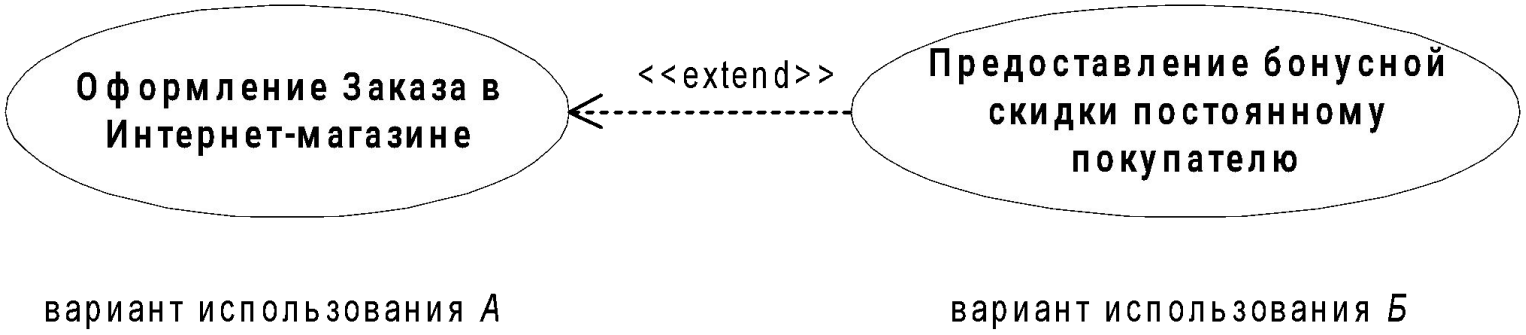
Отношение включения

- Отношение *зависимости (dependency)* определяется как форма взаимосвязи между двумя элементами модели, предназначенная для спецификации того обстоятельства, что изменение одного элемента модели приводит к изменению некоторого другого элемента
- Отношение *включения (include)* специфицирует тот факт, что некоторый вариант использования содержит поведение, определенное в другом варианте использования

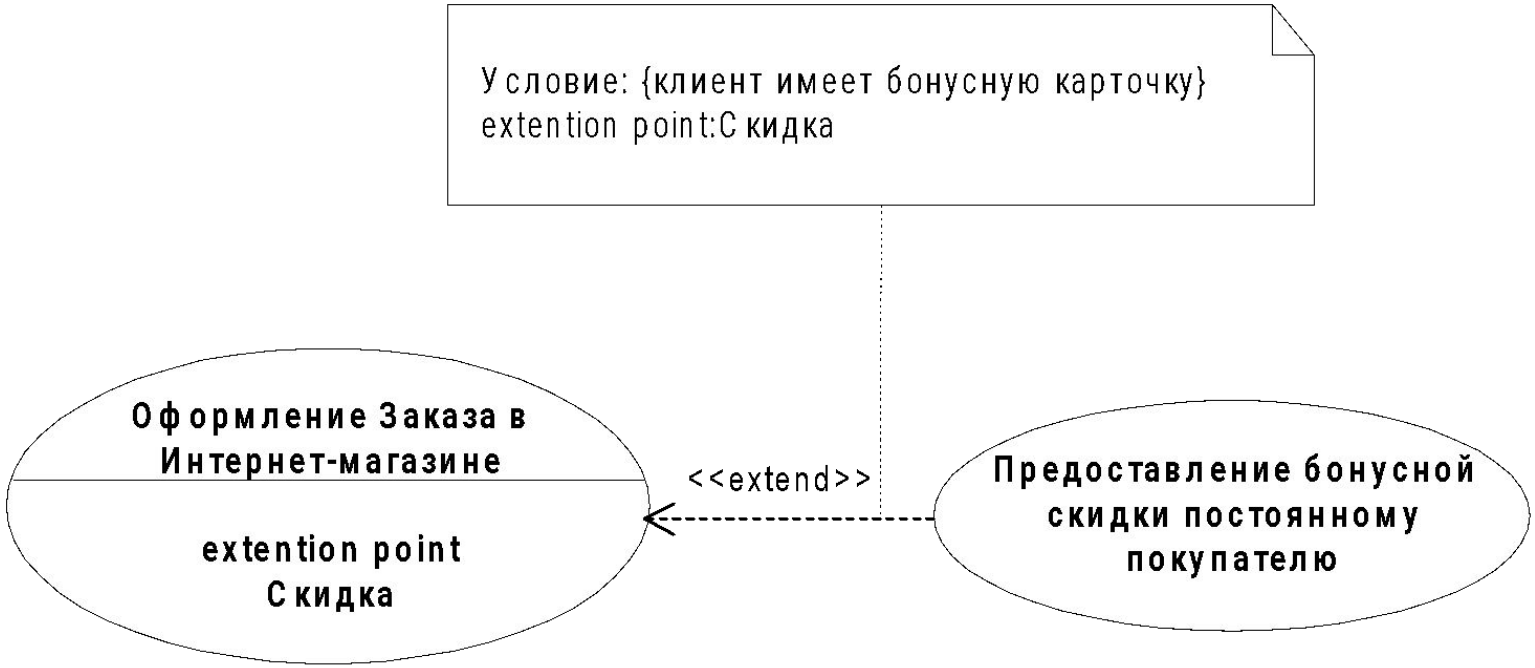


Отношение расширения

- Отношение *расширения (extend)* определяет взаимосвязь одного варианта использования с некоторым другим вариантом использования, функциональность или поведение которого задействуется первым не всегда, а только при выполнении некоторых дополнительных условий.

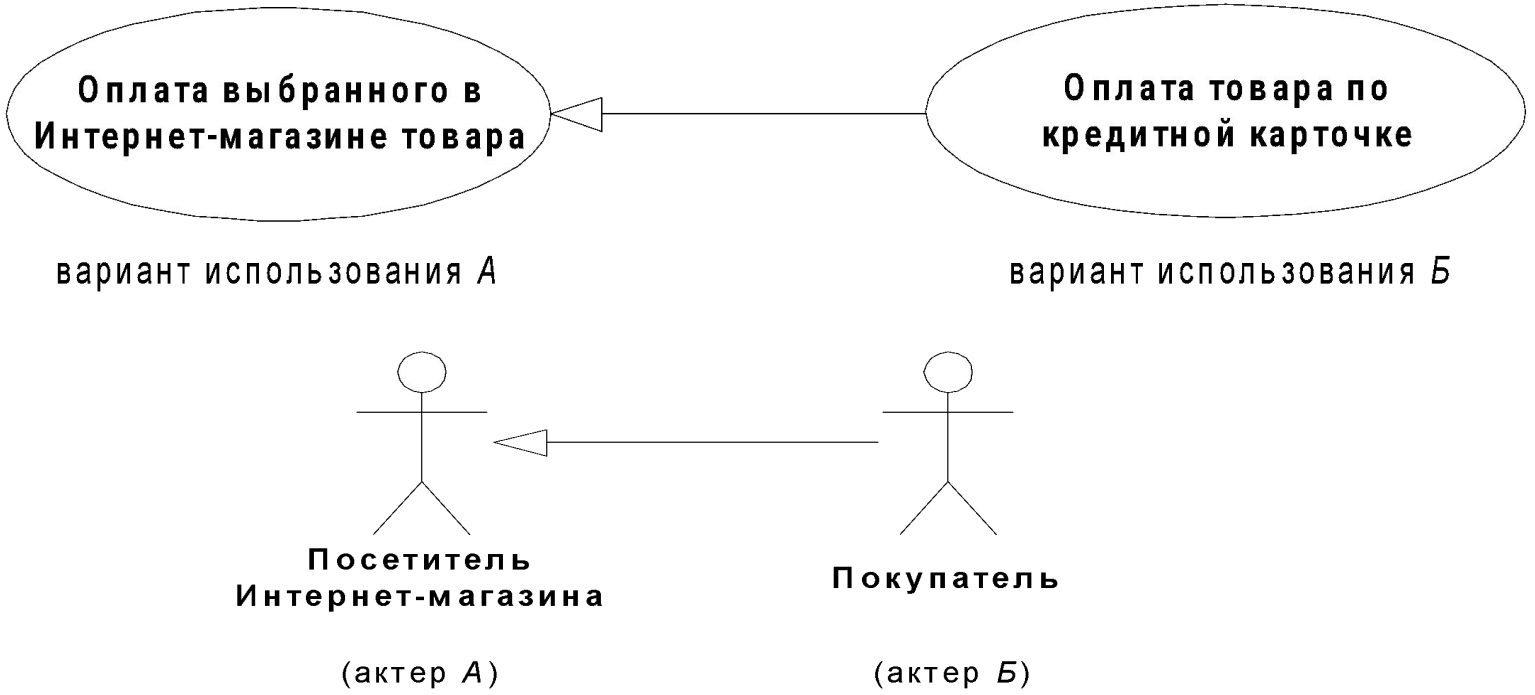


Изображение отношения расширения с условием выполнения

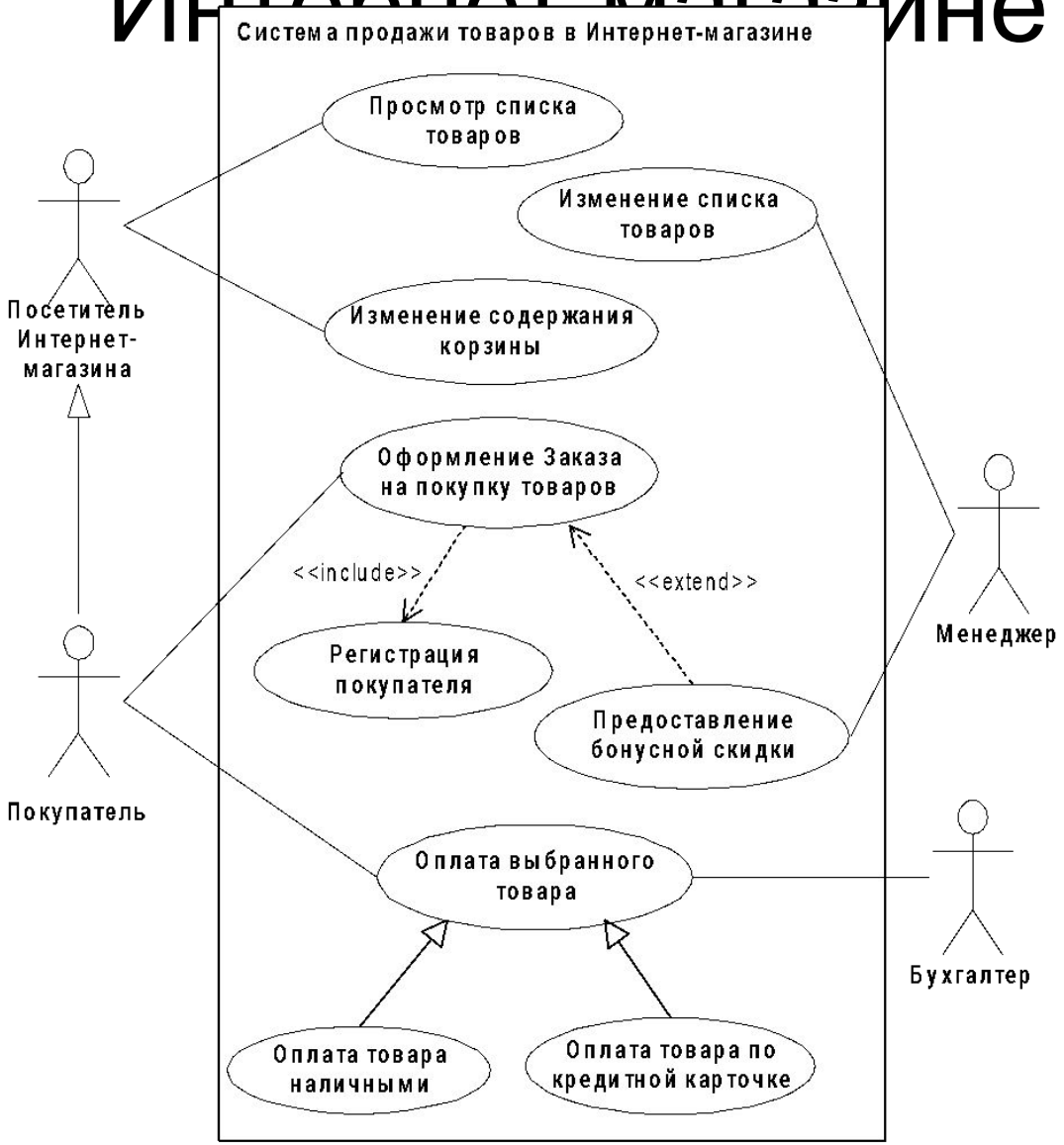


Отношение обобщения

- *Отношение обобщения (generalization relationship)* предназначено для спецификации того факта, что один элемент модели является специальным или частным случаем другого элемента модели



Пример диаграммы ВИ для системы продажи товаров в Интернет-магазине



Спецификация ВИ с помощью текстовых сценариев

- *Сценарий (scenario)* – специально написанный текст, который описывает поведение

Актер	Цель № 1	Успех	Исключение № 1	Примечания
			Исключение № 2	
			Исключение № 3	
	Цель № 2	Успех	Исключение № 1	
			Исключение № 2	
			Исключение № 3	

Типичные ошибки при разработке диаграмм вариантов использования

- Превращение диаграммы вариантов использования в диаграмму деятельности за счет желания отразить все функциональные действия
- Инициатором действий является разрабатываемая система
- Спецификация атрибутов и операций классов до того, как сформулированы все варианты использования
- Задание слишком кратких имен вариантам использования
- Описание вариантов использования в терминологии, непонятной пользователям системы или заказчику
- Отсутствие описаний альтернативных последовательностей действий
- Тратится слишком много времени на решение вопросов о том, какие стереотипы и ассоциации использовать на диаграмме