Программная инженерия

Самочадин Александр Викторович

Samochadin_av@spbstu.ru

Курс: Программная инженерия

Лекции: 1 час в неделю

Лабораторные: 1 час в неделю

Курсовое проектирование: консультации

Зачет

Экзамен

Программная инженерия: активности

Программная инженерия

Соммервилл, Иан Инженерия программного обеспечения, 6-е издание.: Пер. с англ. – М.: Издательский дом

"Вильямс", 2002. – 624 с.

Дана широкая панорама тем инженерии ПО, охватывающих все этапы и технологии разработки программных систем. В книге представлен весь спектр процессов, ведущих к созданию программного обеспечения, начиная от начальной разработки системных требований и далее через проектирование, непосредственное программирование и аттестацию до модернизации программных систем.

Разработка требований

Вигерс Карл, Битти Джой Разработка требований к программному обеспечению. 3-е изд., дополненное / Пер. с англ. — М. : Издательство «Русская редакция» ; СПб. : БХВ-Петербург, 2014. — 736 стр.

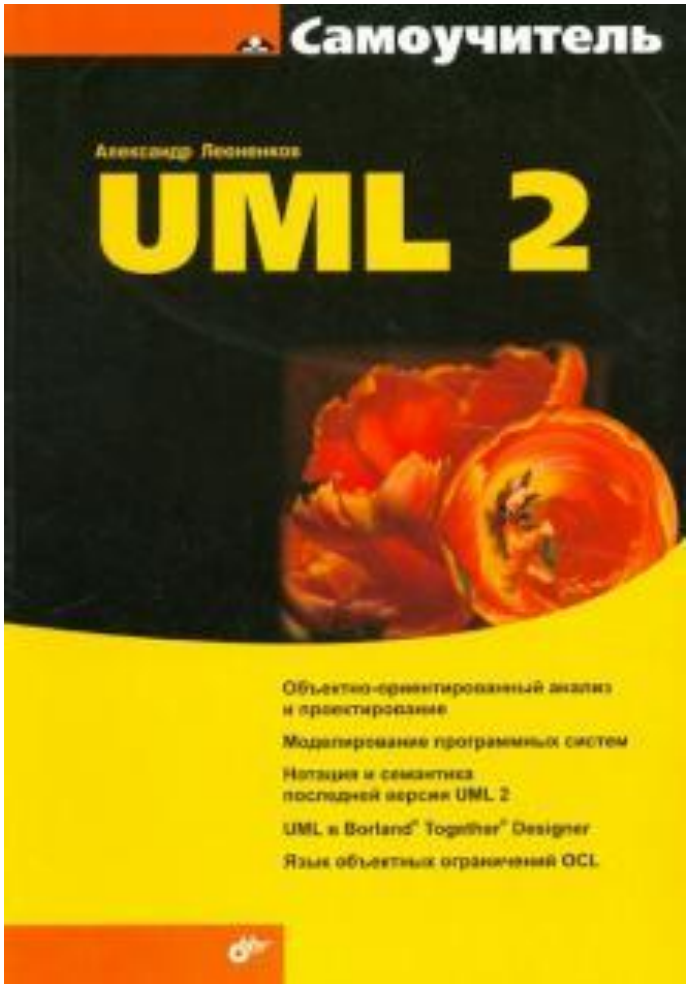
Эта книга - подробное руководство по разработке качественных требований к программному обеспечению. Здесь описаны десятки проверенных на практике приемов выявления, формулирования, разработки, проверки, утверждения и тестирования требований, которые помогут разработчикам, менеджерам и маркетологам создать эффективное ПО. Третье издание дополнено новыми приемами, посвященными разработке требований в проектах гибкой разработки (agile).

Объектно-ориентированный анализ и проектирование

Крэг Ларман, Применение UML 2.0 и шаблонов проектирования (3-е издание). Вильямс 2006. — 736 стр .

В книге рассматриваются основные принципы и приемы объектно-ориентированного анализа и проектирования (ООА/П). В ней содержатся сведения об итеративном и гибком моделировании, шаблонах проектирования, архитектурном анализе и многих других вопросах. Весь материал рассматривается в контексте гибкого подхода к разработке с совместным применением процесса UP и других итеративных методов. Книга будет хорошим руководством для всех, кто интересуется вопросами ООА/П, языком моделирования UML 2 и современными эволюционными подходами к разработке программного обеспечения.

UML



А. Леоненков Самоучитель UML 2. : Издательство: BHV - Санкт – Петербург, 2007, с. 576

Рассмотрена современная технология объектно риентированного анализа и проектирования программных систем и бизнес-процессов в контексте нотации унифицированного языка моделирования UML 2. Подробно изложены все понятия языка UML 2 в полном соответствии с оригинальной спецификацией последней версии этого языка. Приведены конкретные рекомендации по разработке канонических диаграмм языка

UML

Г. Буч, А. Якобсон, Дж. Рамбо UML
Издание второе Издательство: Питер
Серия: Классика Computer Science 2006 г., с. 736

Эта книга представляет собой полный справочник по языку UML. Она адресована в первую очередь разработчикам, системным архитекторам, руководителям проектов, инженерам-системщикам, программистам, аналитикам, заказчикам и вообще всем, кому по роду деятельности приходится описывать, проектировать и строить сложные программные системы, а также разбираться в их функционировании. В книге дается всестороннее описание понятий и конструкций UML, включая их семантику, нотацию и назначение. Материал организован таким образом, чтобы книгой было удобно пользоваться, несмотря на ее объем и полноту содержания.

Программная инженерия

Программная инженерия (Software engineering) – это инженерная дисциплина, которая охватывает все аспекты производства программных продуктов.

Программная инженерия— приложение систематического, дисциплинированного, измеримого подхода к разработке, функционированию и сопровождению программного обеспечения, а также исследованию этих подходов; то есть, приложение дисциплины инженерии к программному обеспечению (ISO/IEC/IEEE 24765:2017)

Области знаний программной инженерии (Software Engineering Body of Knowledge v 3.0 SWEBOK)

Knowledge areas	**Области знаний**	**Приоритет**
software requirements	требования к ПО	Высший
software design	проектирование ПО	Выше среднего
software construction	конструирование ПО	Выше среднего
software testing	тестирование ПО	Выше среднего
software maintenance	сопровождение ПО	Средний
software configuration management	управление конфигурацией: определяет, как вносятся изменения в ПО и как собираются новые выпуски	Выше среднего
software engineering management	управление IT проектом	Высший
software engineering process	процесс программной инженерии: жизненный цикл ПО и составляющие его процессы;	Высший
software engineering models and methods	модели и методы разработки: методы разработки и программные инструменты для всех областей знаний	Выше среднего
software quality	качество ПО	Выше среднего
software engineering professional practice	описание критериев профессионализма и компетентности;	Средний
software engineering economics	экономические аспекты разработки ПО	Высший

Основные области знаний SWEBOK

Организационные области знаний SWEBOK

Языки моделирования

Knowledge areas	Области знаний	Приоритет
Unified Modeling Language (UML)	Унифицированный язык моделирования	Высший
Business Process Model and Notation (BPMN)	Нотация и модель бизнес-процессов	Выше среднего

UML

Unified Modeling Language (UML) — унифицированный язык моделирования для описания, визуализации и документирования объектно-ориентированных систем в процессе их анализа и проектирования

Язык UML предоставляет стандартный способ написания проектной документации на системы

Стандарт UML

Стандарт на язык моделирования разработан консорциумом фирм Object Management Group: http://www.omg.org

Текущая версия стандарта, доступная для свободного скачивания, UML 2.5.1:

https://www.omg.org/spec/UML/2.5.1/PDF

UML 2.4.1 принят в качестве международного стандарта ISOUML 2.4.1 принят в качестве международного
стандарта ISO/IEC 19505-1, 19505-2.

BPMN

Business Process Model and Notation (BPMN) нотация и модель бизнес-процессов — система условных обозначений (нотация) для моделирования бизнес-процессов.

Последняя версия стандарта — BPMN 2.0.2 (январь 2014).

Курс: Программная инженерия

Курсовая работа по курсу «Программная инженерия»

Разделы курсовой работы:

- 1. Концепция проекта, которая отражает в краткой форме все составляющие проекта.
- 2. Технико-экономическое обоснование проекта
- 3. Требования к проекту (Use case документ)
- 4. Описание содержания проекта, которое содержит описание работ, которые предстоит выполнить, и результатов поставки.
- 5. План реализации проекта.
- 6. Описание проекта
- 7. Тест-план

Требования к ПО

Требования к ПО – Software Requirements – свойства программного обеспечения, которые должны быть надлежащим образом представлены в нѐм для решения конкретных практических задач.

Требование определяется как некое свойство ПО, необходимое пользователю для решения проблемы при достижении поставленной цели;

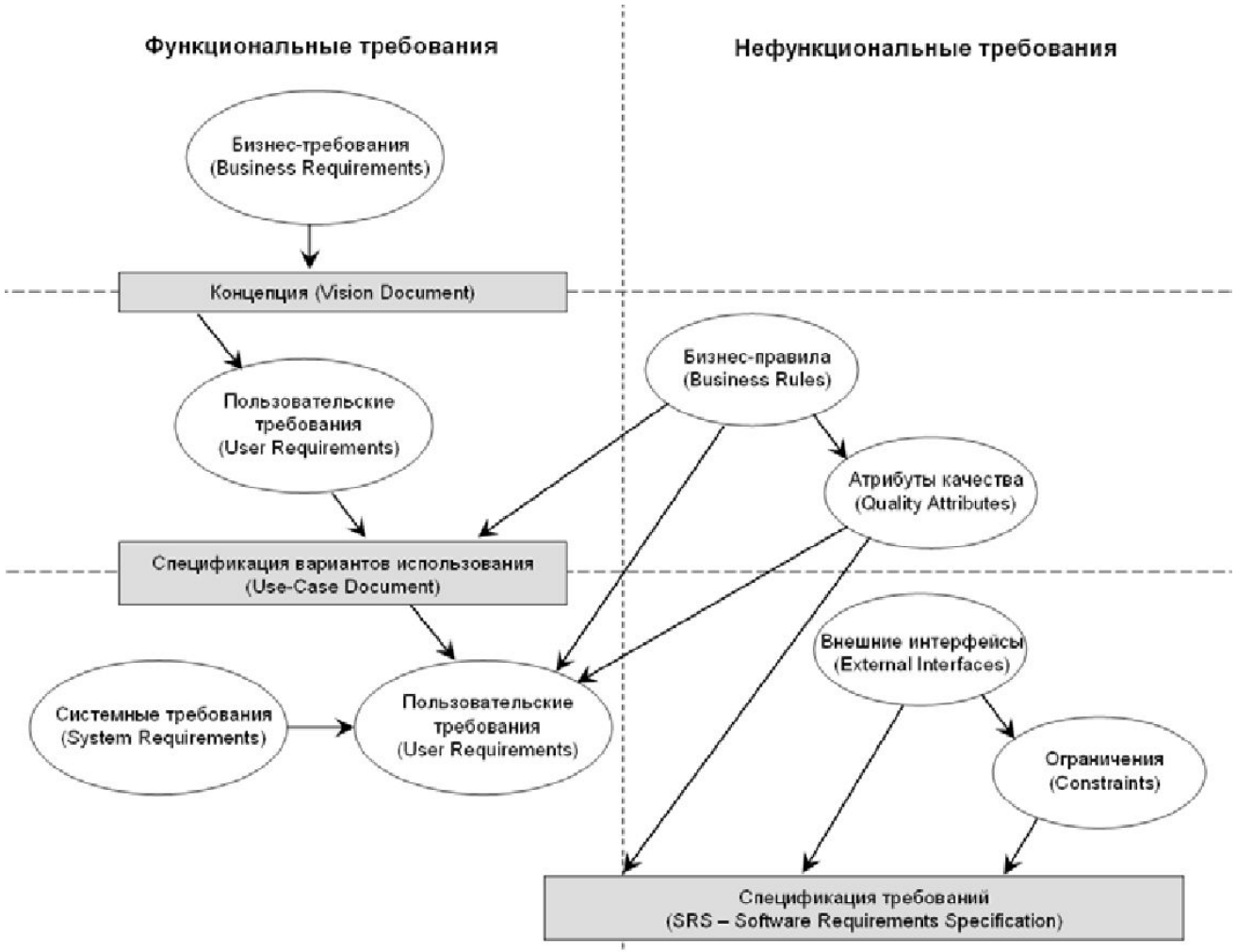
“Условие или возможность, которое система должна выполнять или поддерживать”.

Требования к ПО

Разработка требований включает следующие типы деятельности:

- **Сбор (выявление)требований** — общение с клиентами и пользователями, чтобы определить, каковы их требования; анализ предметной области.
- **Анализ требований** — систематизация требований. Определение, являются ли собранные требования неясными, неполными, неоднозначными или противоречащими; выявление взаимосвязи требований.
- **Документирование требований** — требования могут быть в различных формах, таких как простое описание, сценарии использования, пользовательские истории.
- **Специфицирование требований**.
- **Управление требованиями.**

Требования к ПО



Уровни требований по Вигерсу

Требования к ПО

- **Бизнес-требования (***business requirements*). Примеры бизнес-требования: система должна сократить срок оборачиваемости обрабатываемых на предприятии заказов в три раза. Бизнес-требования обычно формулируются заказчиком.
- **Пользовательские требования** (Пользовательские потребности). (User Requirement Specification - URS) Описание на естественном языке (плюс поясняющие диаграммы) функций, выполняемых системой, и ограничений, накладываемых на нее. Ориентированы на заказчика ПО (руководство и специалисты организации – заказчика, пользователи)
- **Специфицированные требования** (или просто требования) (Software Requirement Specification - SRS). Детализированное описание системных функций и ограничений. Системные требования служат основой для контракта между заказчиком и разработчиком. Ориентированы на заказчика и разработчика (специалисты разработчика и исполнителя)

Требования к ПО

- **Сегодняшняя задача:** разработка пользовательских функциональных требований к ПО с помощью **методики основанной на вариантах использования.**

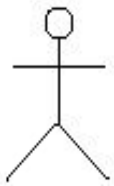
В рамках этой методики разрабатываются:

1. Диаграммы вариантов использования (Use Case диаграмма)
2. Сценарии вариантов использования
3. Создается Use Case документ

Назначение диаграммы вариантов использования

- Определить общие границы функциональности проектируемой системы в контексте моделируемой предметной области.
- Специфицировать требования к функциональному поведению проектируемой системы в форме вариантов использования.
- Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.
- Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями

Основные обозначения на диаграмме вариантов использования



actor

communication association

generalization

use case

«extend»

extend

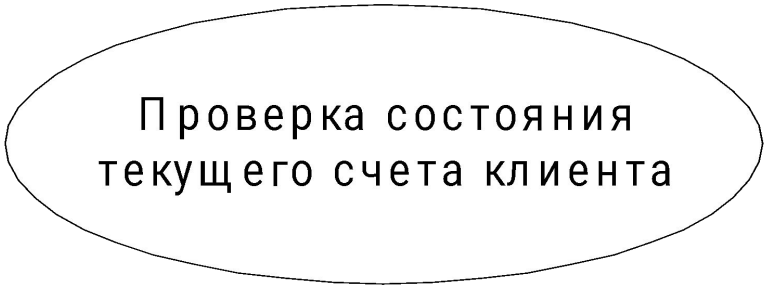
«include»

include

system boundary

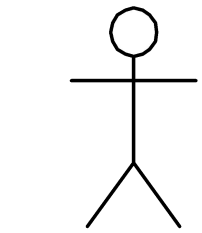
Вариант использования (use case)

- 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



Действующее лицо, 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



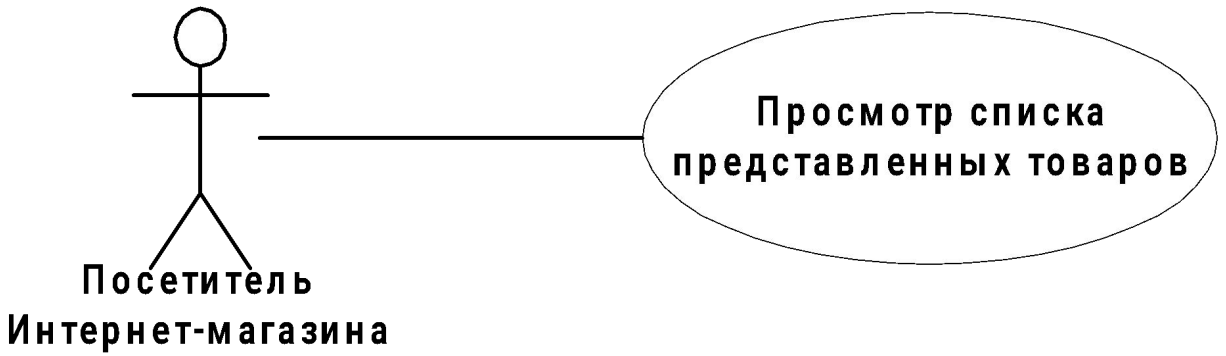
Клиент банка

Вопросы для идентификации действующих лиц 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

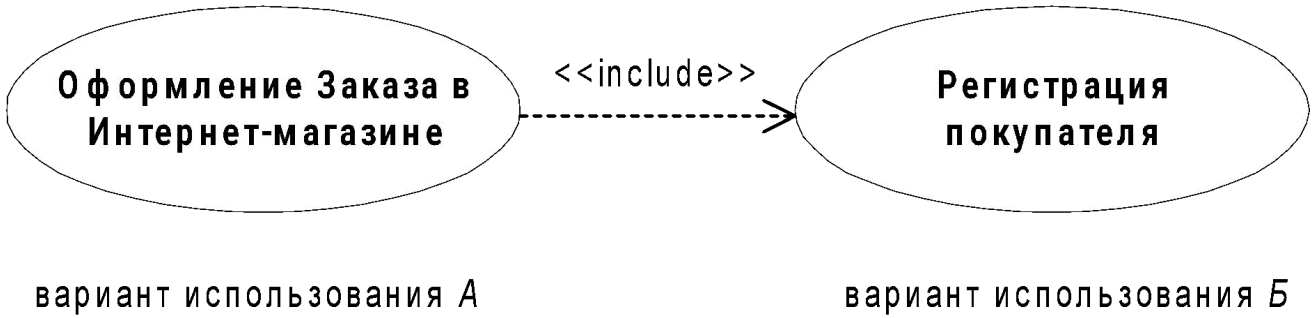
Отношение ассоциации

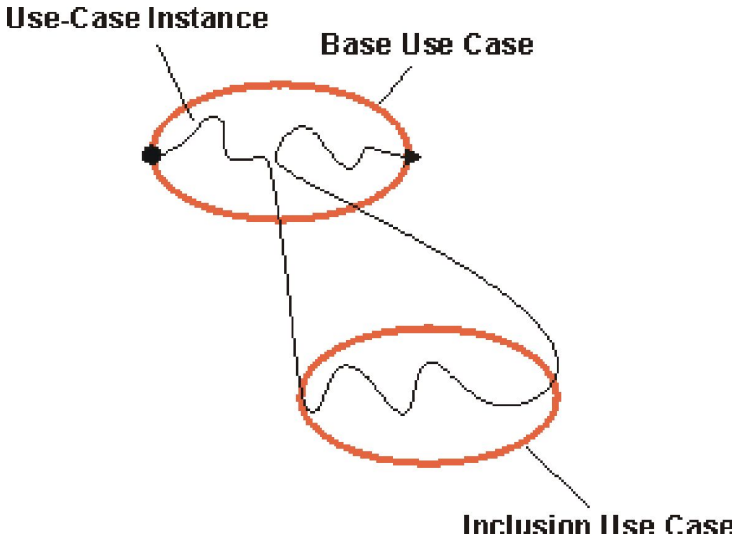
- *Ассоциация* (association) является одним из фундаментальных понятий в языке UML 2.x и может использоваться на различных канонических диаграммах при построении визуальных моделей
- Применительно к диаграммам вариантов использования отношение ассоциации может служить только для обозначения взаимодействия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



Отношение включения

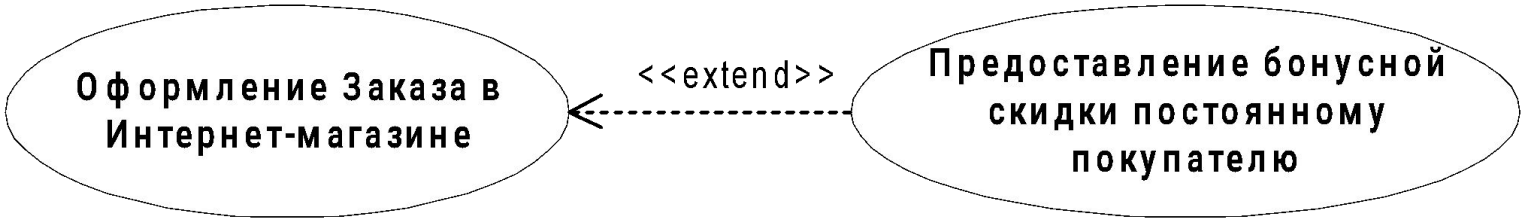
- Отношение *включения (include)* специфицирует тот факт, что некоторый вариант использования содержит поведение, определенное в другом варианте использования





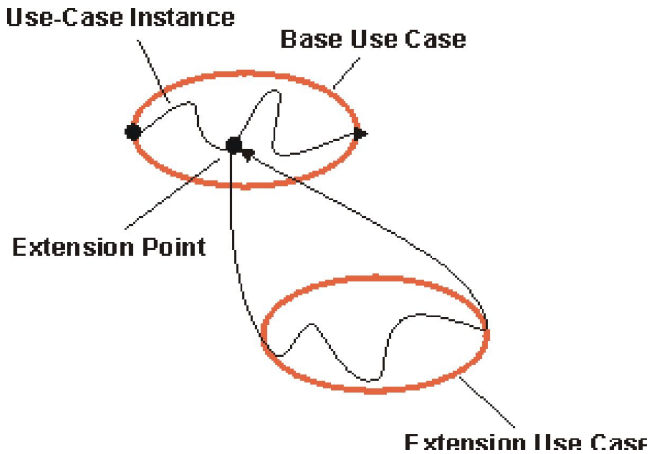
Отношение расширения

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.



вариант использования *А*

вариант использования *Б*



Отношение обобщения

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

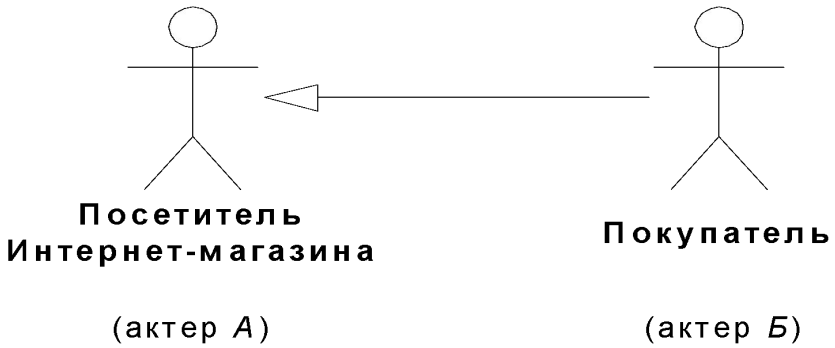


Диаграмма вариантов использования

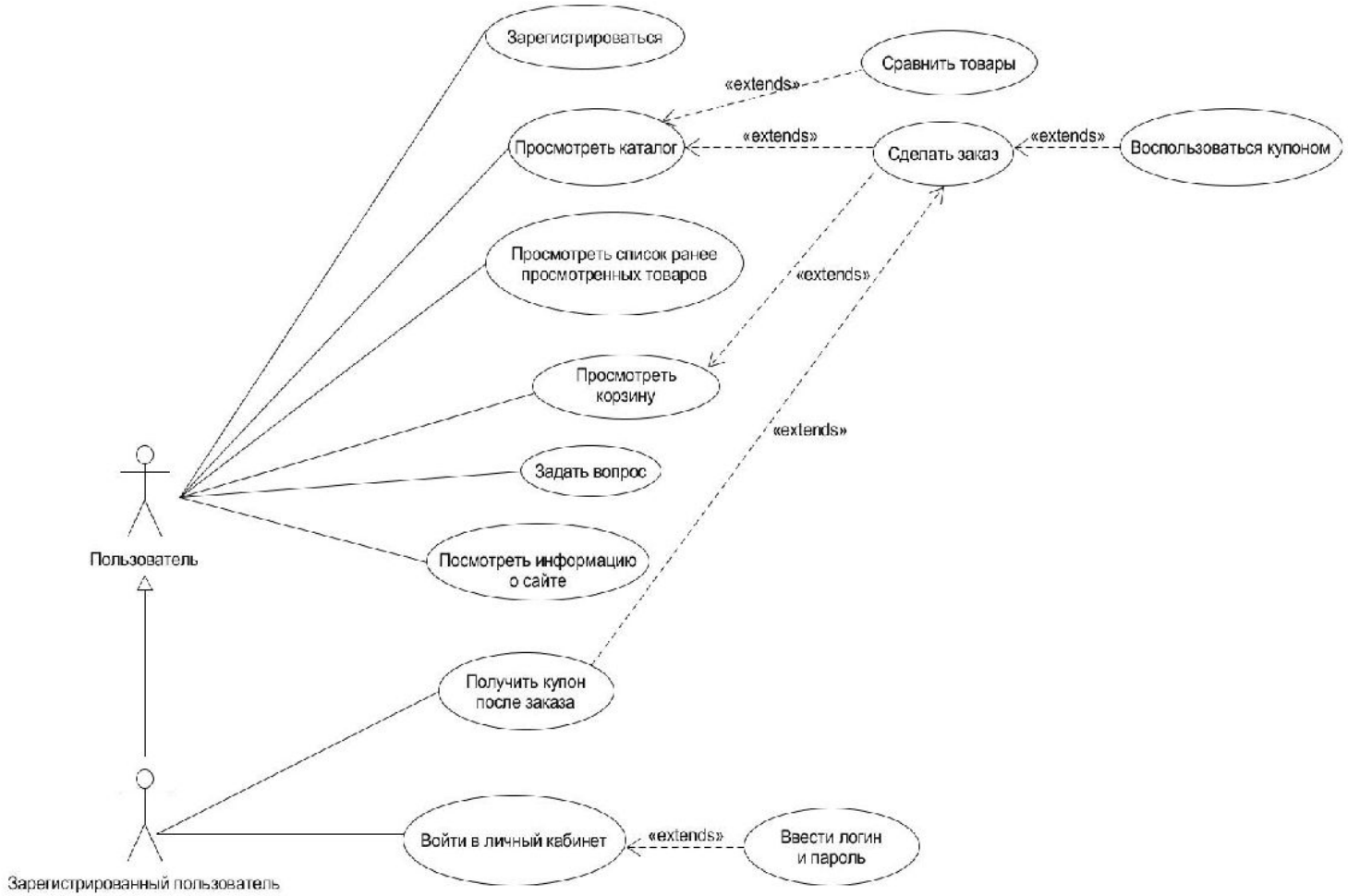


Диаграмма вариантов использования

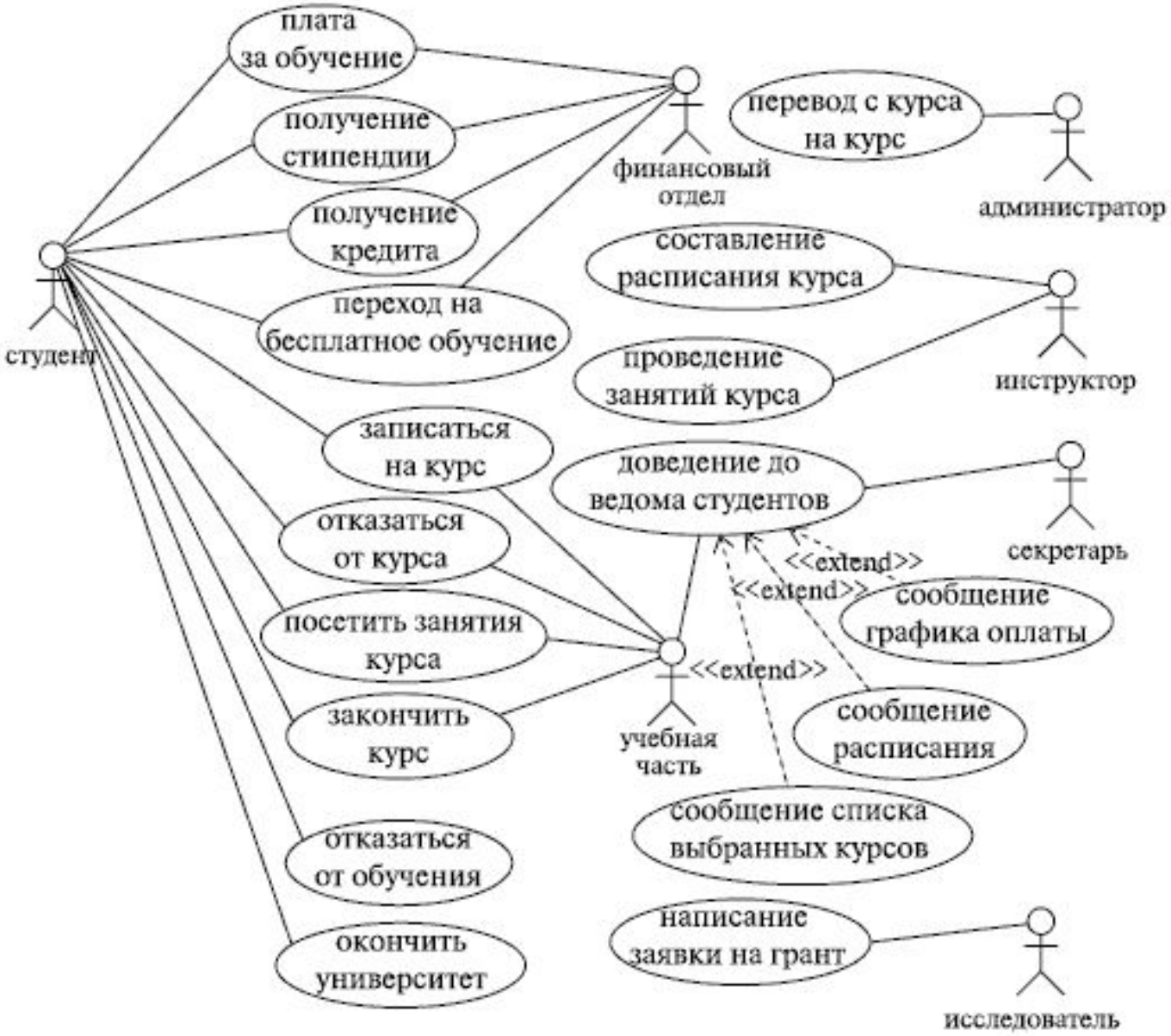


Диаграмма вариантов использования (с ошибками)

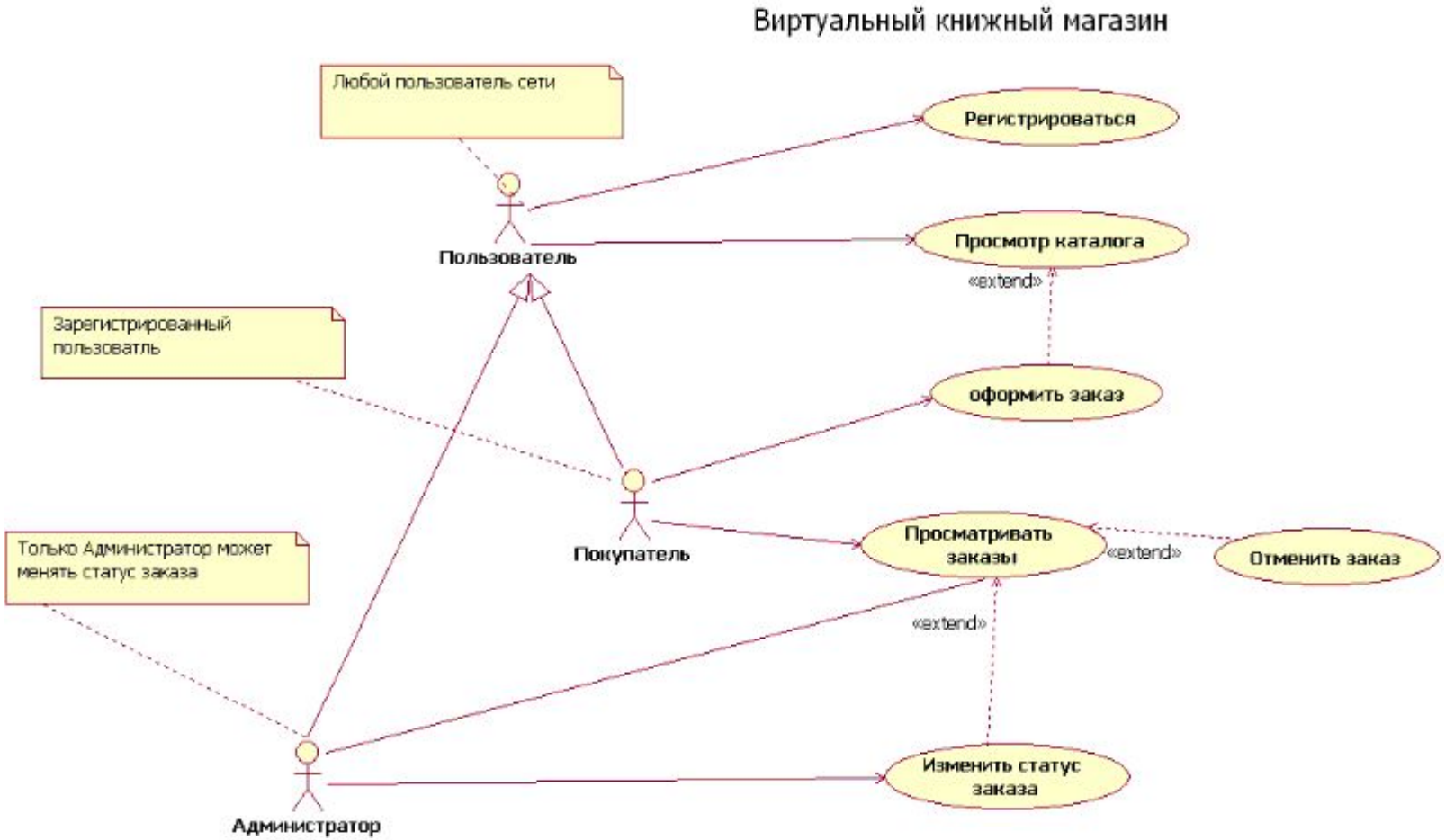


Диаграмма вариантов использования (с ошибками)

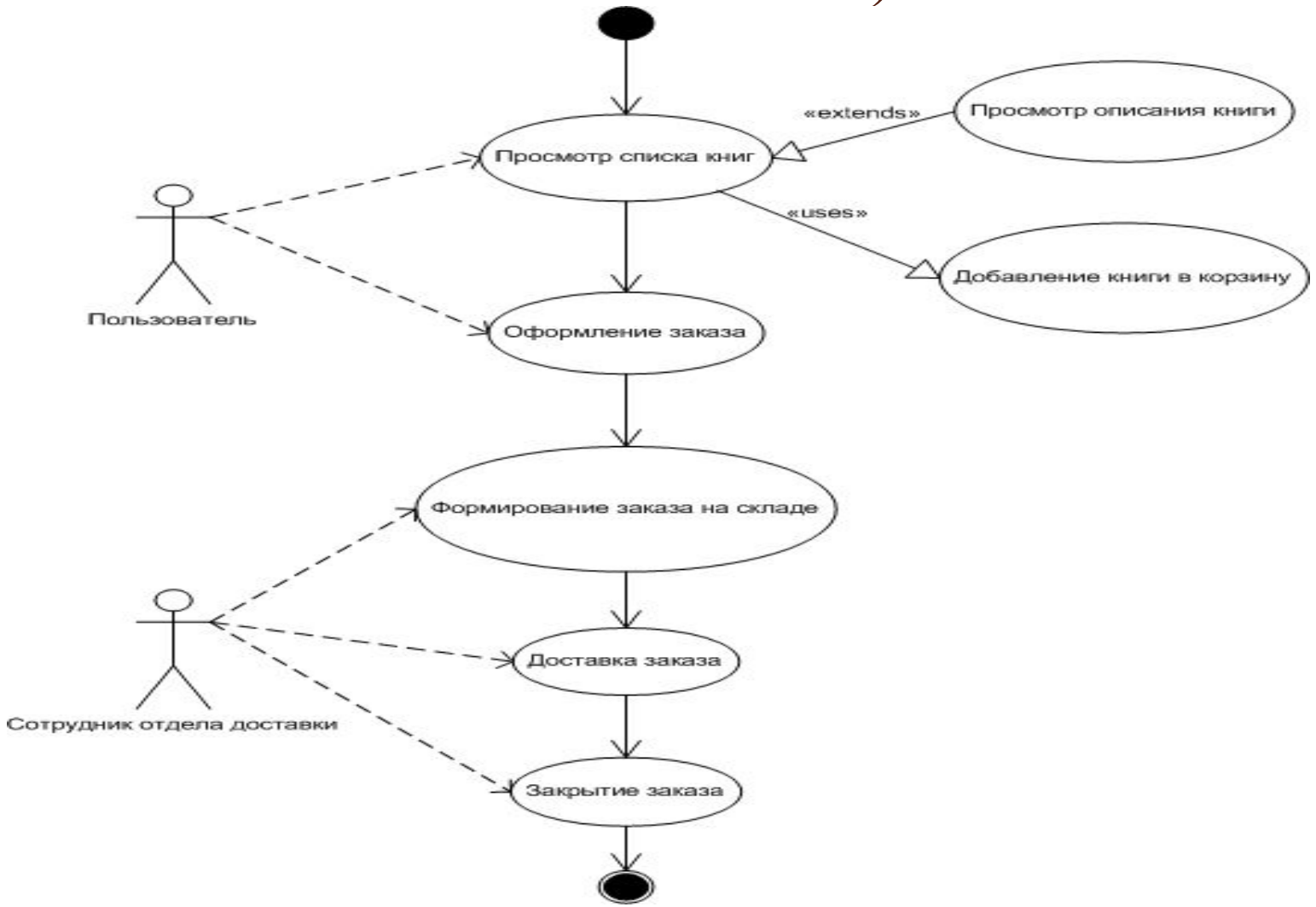


Диаграмма вариантов использования (с ошибками)

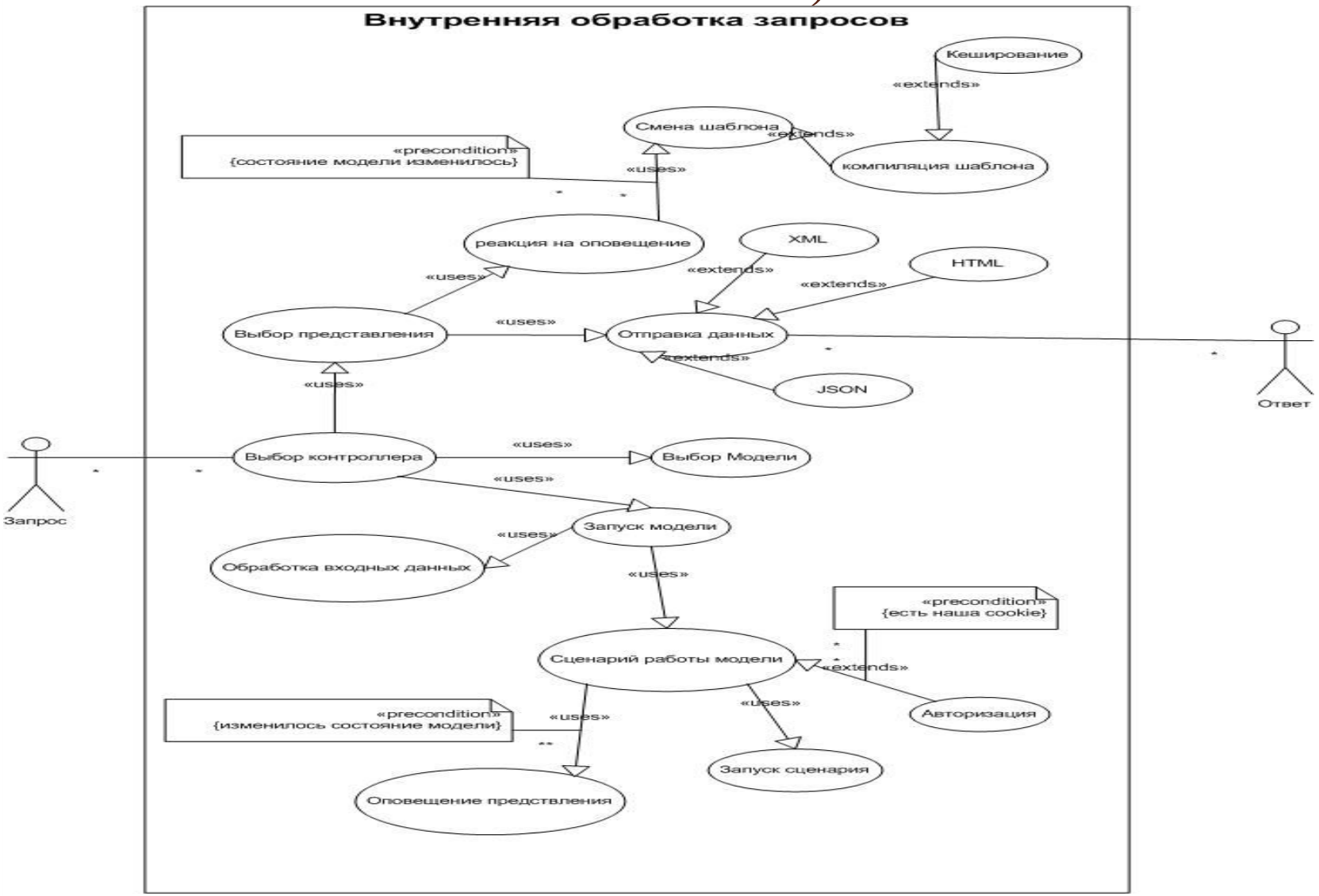


Диаграмма вариантов использования (с ошибками)

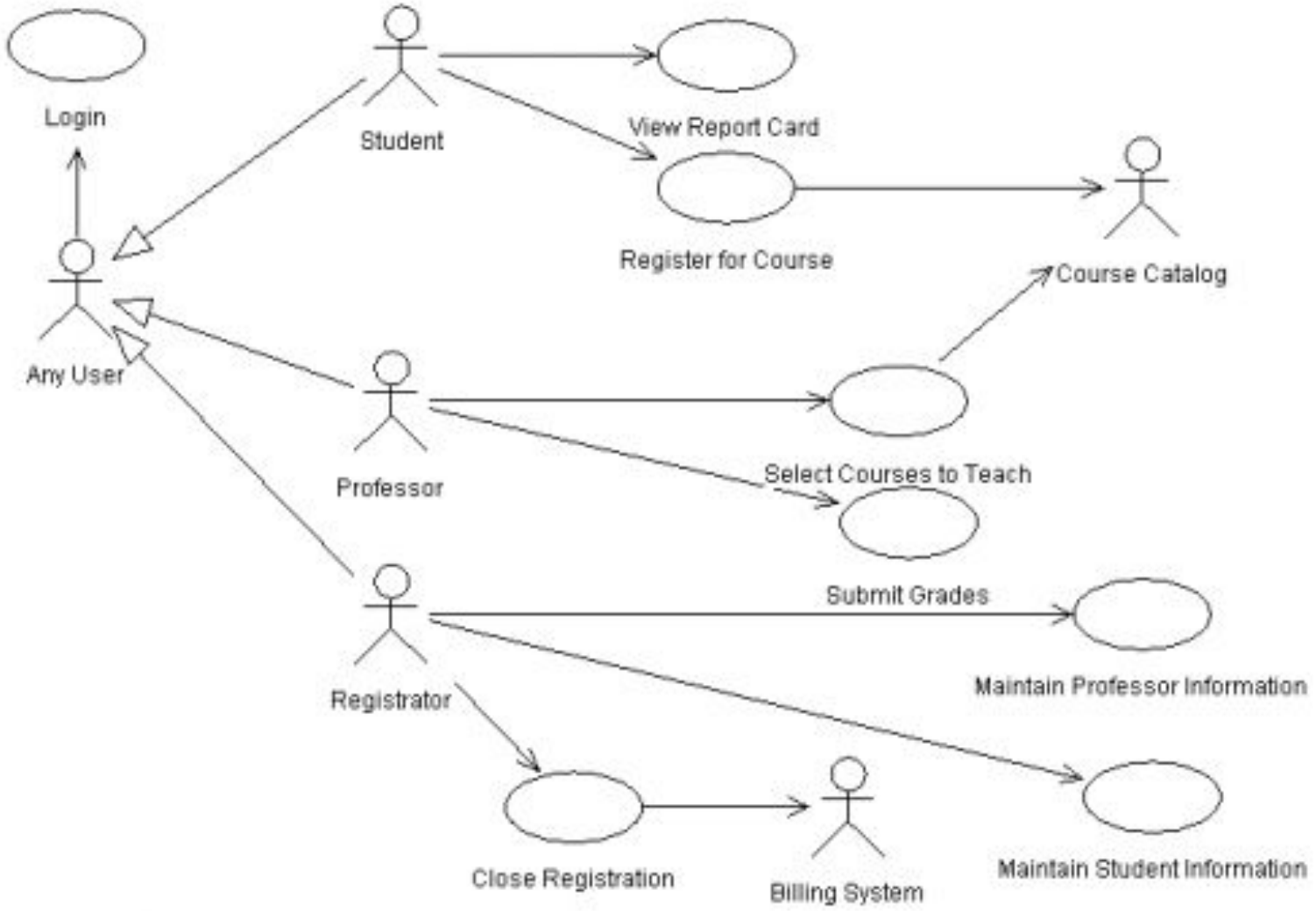


Диаграмма вариантов использования

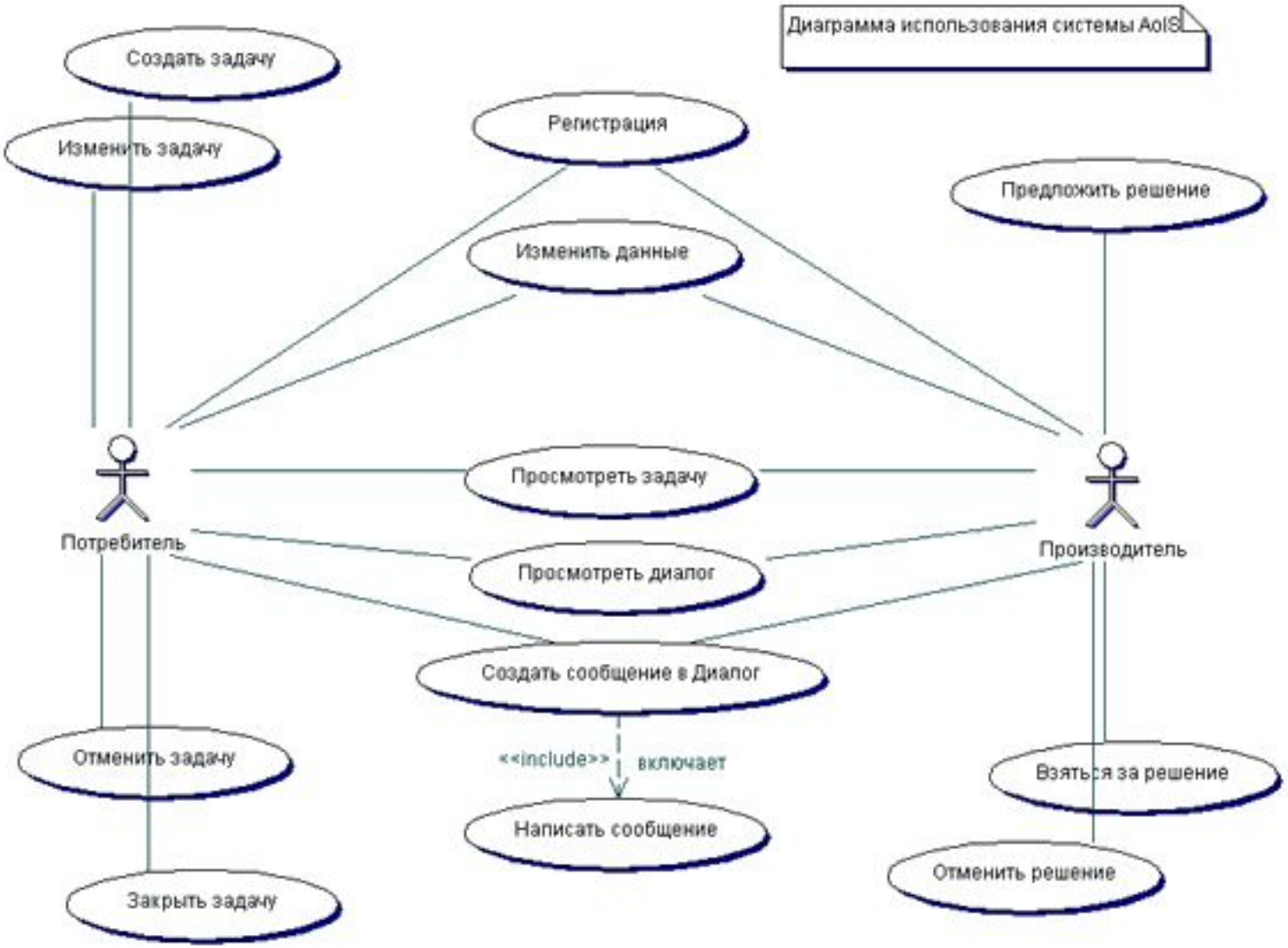


Диаграмма вариантов использования

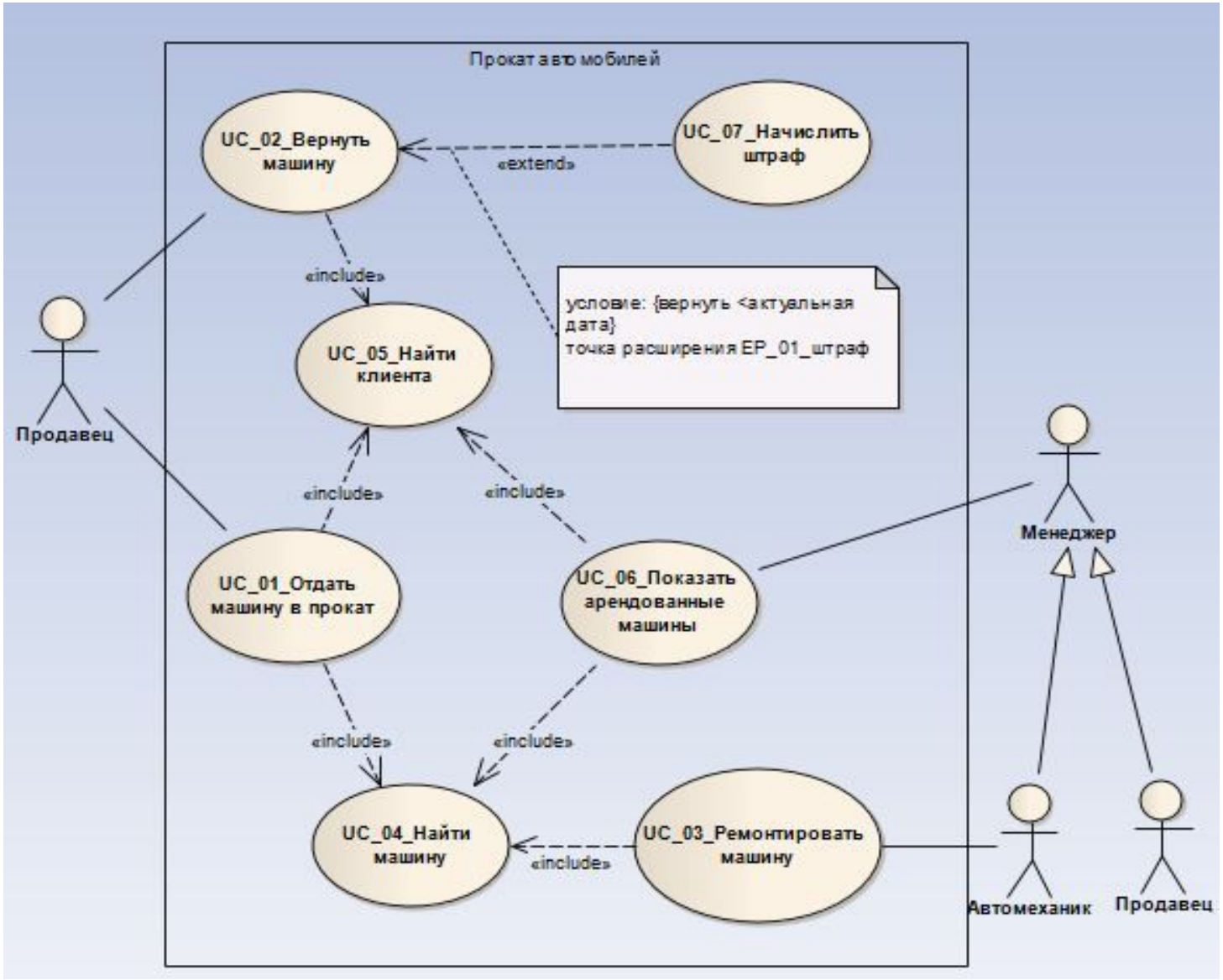


Диаграмма вариантов использования

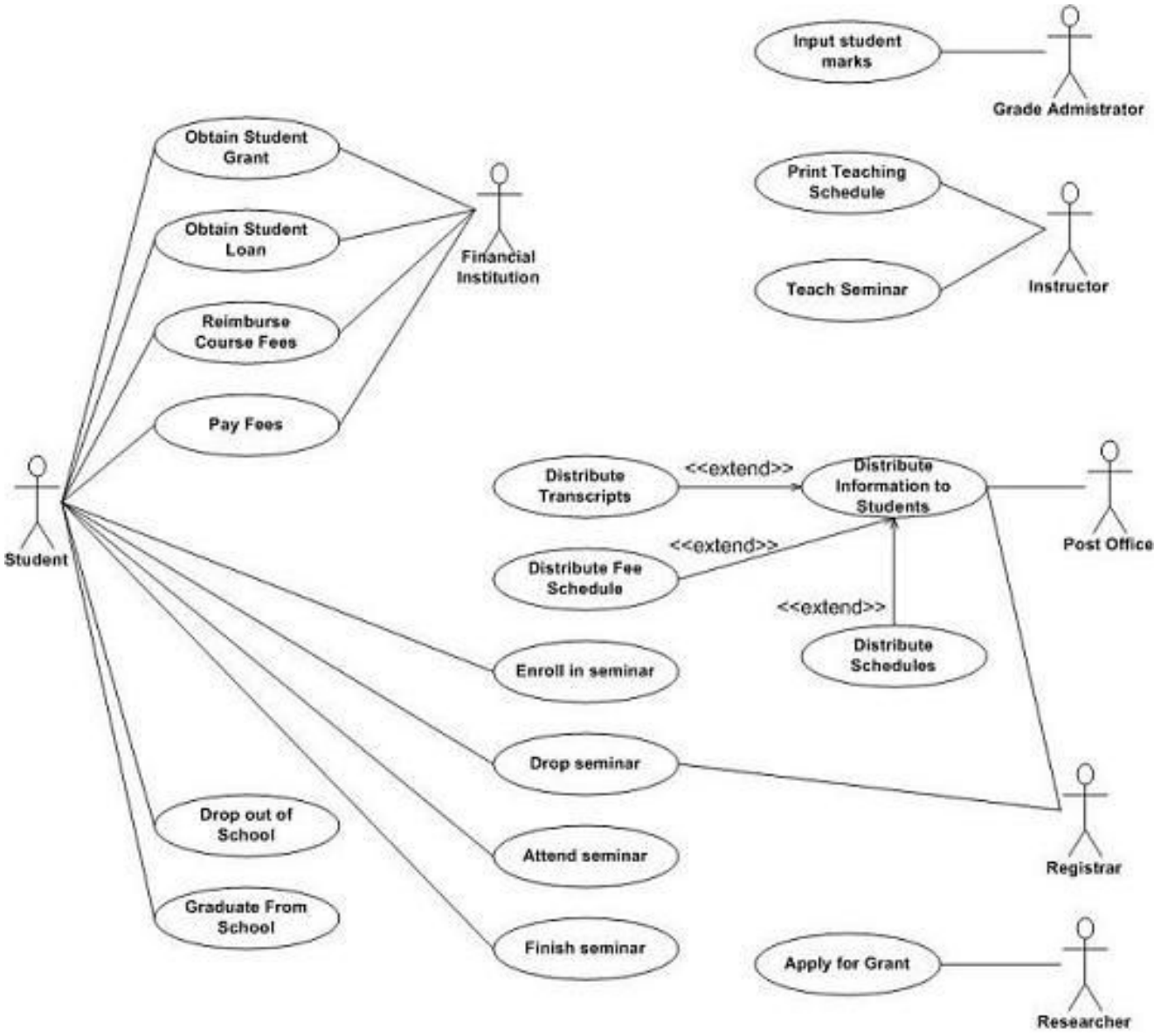
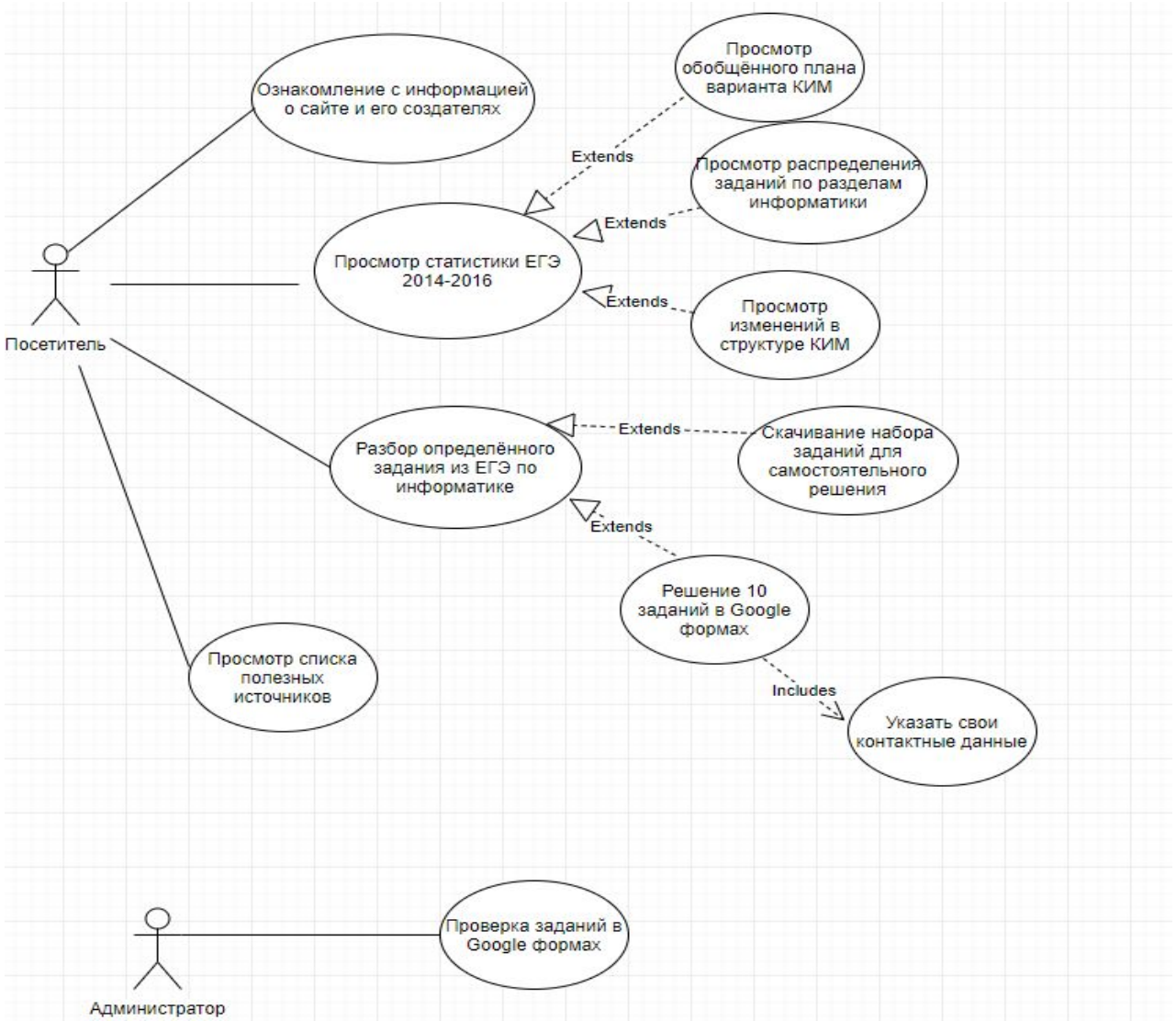


Диаграмма вариантов использования (с ошибками)



Сценарии вариантов использования.

- **Сценарий варианта использования** – последовательность действий (сценарий), включая варианты, при которых система приносит действующему лицу полезный и понятный результат и действия, которые не приносят такого результата.
- **Действующее лицо** – определяет связанный набор ролей, которые пользователи системы могут играть при взаимодействии с этой системой. Пользователем может быть человек или внешняя система.
- **Действующее лицо** – кто-то или что-то обладающее поведением.
- **Основное действующее лицо**: действующее лицо, инициирующее взаимодействие с системой для достижения некоторой цели.
- **Предусловие** объявляет выполнение какого условия гарантируется перед тем, как разрешить запуск этого варианта использования
- **Триггер** – событие, которое запускает вариант использования.
- **Гарантия успеха** – устанавливает, что интересы участников удовлетворены
- **Основной сценарий** – вариант, в котором не возникает никаких ошибок
- **Расширения** – различные отклонения от основного сценария.
- **Область действия** – идентифицирует рассматриваемую систему
- **Уровень цели**: обобщенный (описание бизнес-процесса), уровень цели пользователя (которую можно достигнуть за один сеанс), уровень подцели.

Сценарий варианта использования.

Пример сценария варианта использования.

Оформление продажи

Основное действующее лицо: кассир

Область действия: система автоматизации торговли

Уровень: пользовательский

Участники и интересы:

Кассир – хочет точно и быстро ввести данные

Покупатель – хочет купить товары и быстро оформить покупку с минимальными усилиями. Хочет получить подтверждение факта покупки.

Компания – хочет оформить покупку и удовлетворить интересы покупателя

Налоговые службы – хотят получать налог с каждой продажи.

Триггер: покупатель подходит к кассовому аппарату с выбранными товарами.

Сценарий варианта использования.

Основной сценарий:

1. Кассир открывает новую продажу.
2. Кассир вводит идентификатор товара
3. Система записывает наименование товара и на основании каталога товаров и выдает его описание, цену и общую стоимость. Цена вычисляется на основе набора правил.

   Кассир повторяет действия , описанные в пп. 2-3, для каждого наименования товара
4. Система вычисляет общую стоимость покупки с налогом
5. Кассир сообщает покупателю общую стоимость и предлагает оплатить покупку
6. Покупатель оплачивает покупку, система обрабатывает платеж
7. Система регистрирует продажу в реестре продаж и отправляет информацию о продаже внешней бухгалтерской системе и системе складского учета (для обновления данных)
8. Система выдает товарный чек
9. Покупатель покидает магазин с чеком и товарами (если он что-то купил)

Сценарий варианта использования.

Расширения:

2а. Неправильный идентификатор

2а1. Система уведомляет об ошибке и отменяет ввод данного наименования товара

2б. В рамках одной категории существует несколько наименований товара

2б1. Кассир может ввести идентификатор товара и количество единиц.

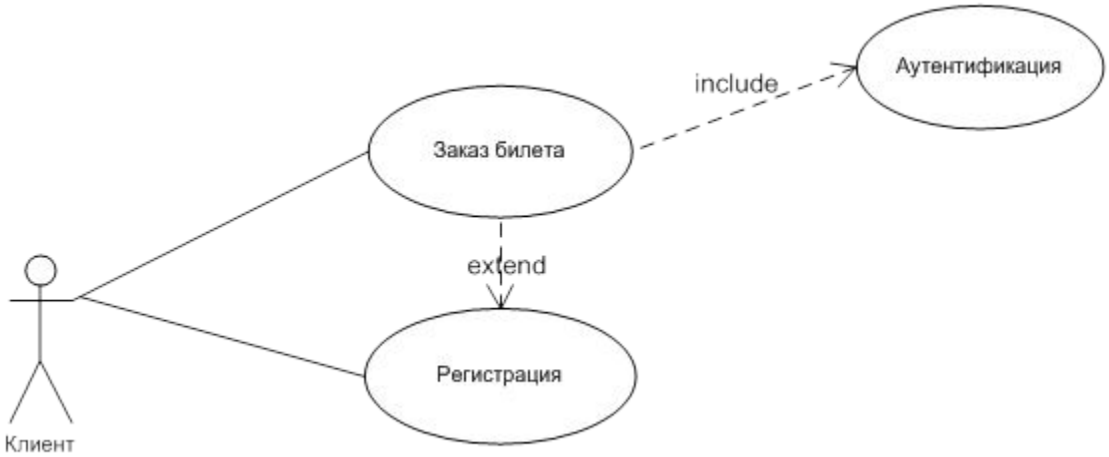
2в. Покупатель просит продавца отменить покупку одного из товаров

2в1. Кассир вводит идентификатор товара для удаления из продажи

2в2. Система отображает обновленную промежуточную стоимость.

Сценарий варианта использования.

Пример. Фрагмент диаграммы вариантов использования для системы заказов



Сценарий варианта использования.

Регистрация
Основное действующее лицо: клиент
Гарантия успеха: пользователь зарегистрировался
Триггер: пользователь выбирает элемент интерфейса «Регистрация»
Основной сценарий:

1. Система предоставляет пользователю условия пользовательского соглашения
2. Пользователь принимает соглашение
3. Система предоставляет регистрационную форму
4. Пользователь выбирает имя пользователя и пароль
5. Пользователь выбирает или вводит вопрос, который будет задан, если он забудет пароль
6. Пользователь вводит ответ на вопрос
7. Пользователь вводит дополнительную информацию (имя, фамилию, день рождения, пол)
8. Система регистрирует пользователя

Расширения:

2 а. Пользователь не принимает соглашение
2 а 1. Система выходит из процесса регистрации
8 а. Пользователь неправильно заполняет форму
8 а 1. Система выводит сообщение об ошибке и предоставляет форму еще раз
8 б. Пользователь вводит существующий имя пользователя
8 б 1. Система выводит сообщение и предоставляет форму еще раз

Сценарий варианта использования.

Аутентификация
Основное действующее лицо: пользователь
Гарантия успеха: пользователь аутентифицировался в системе
Триггер: пользователь входит в систему (выбирает «Вход в систему»)
Основной сценарий:
1. Система предоставляет форму для аутентификации
2. Пользователь вводит имя пользователя и пароль и подтверждает введенную информацию
3. Система проверяет наличие account пользователя
4. Система проверяет пароль
5. Система аутентифицирует пользователя

Расширения:
2 a Пользователь отказывается от аутентификации
    2 a 1. Система возвращается на предыдущий уровень
2 b. Пользователь забыл пароль
    2 b 1 Пользователь выбирает элемент интерфейса «Забыл пароль»
    2 b 2 Система предоставляет форму «Восстановление забытого пароля», содержащую информацию об адресе администратора, к которому может обратиться пользователь
    3 b 3 Пользователь подтверждает получение информации
    3 b 4 Система возвращается на предыдущий уровень
3 a, 4 a. Пользователь вводит неправильные имя пользователя или пароль
    3 a 1. Система выводит сообщение об ошибке и предоставляет форму еще раз
3 b. Пользователь превышает лимит попыток аутентификации
    4 b 1. Система выводит сообщение, блокирует имя пользователя и возвращается на предыдущий уровень

Основные разработчики языка UML (Three amigos)

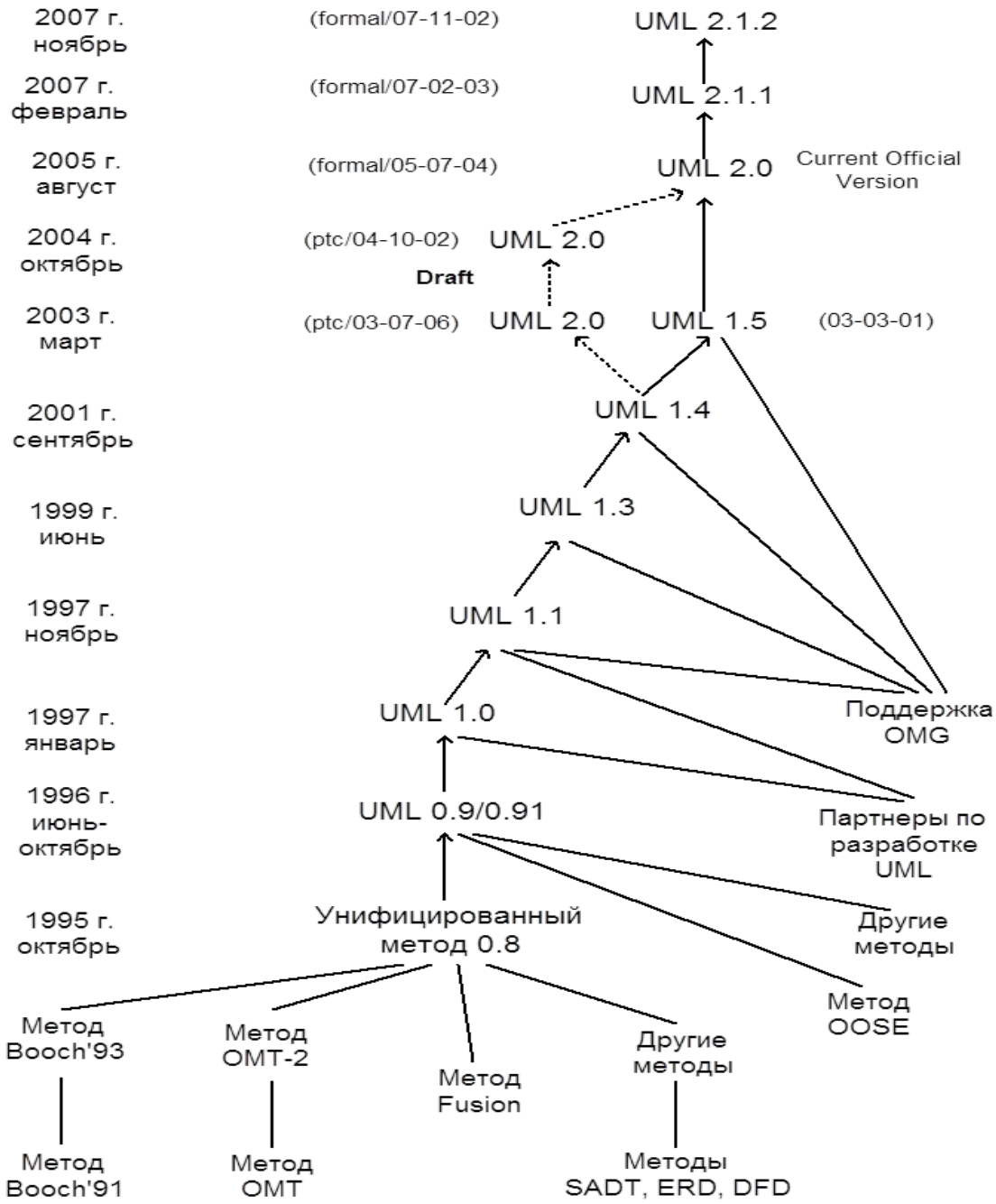
Grady Booch
Гради Буч

Dr. James Rumbaugh
Джеймс Рамбо
(Джим Румбах)

Dr. Ivar Jacobson
Айвар Джекобсон
(Ивар Якобсон)

- **OMG** (Object Management Group) — название консорциума, созданного в 1989 году для разработки индустриальных стандартов с их последующим использованием в процессе создания масштабируемых неоднородных распределенных объектных сред.
- Официальный сайт: www.omg.org

История развития языка UML



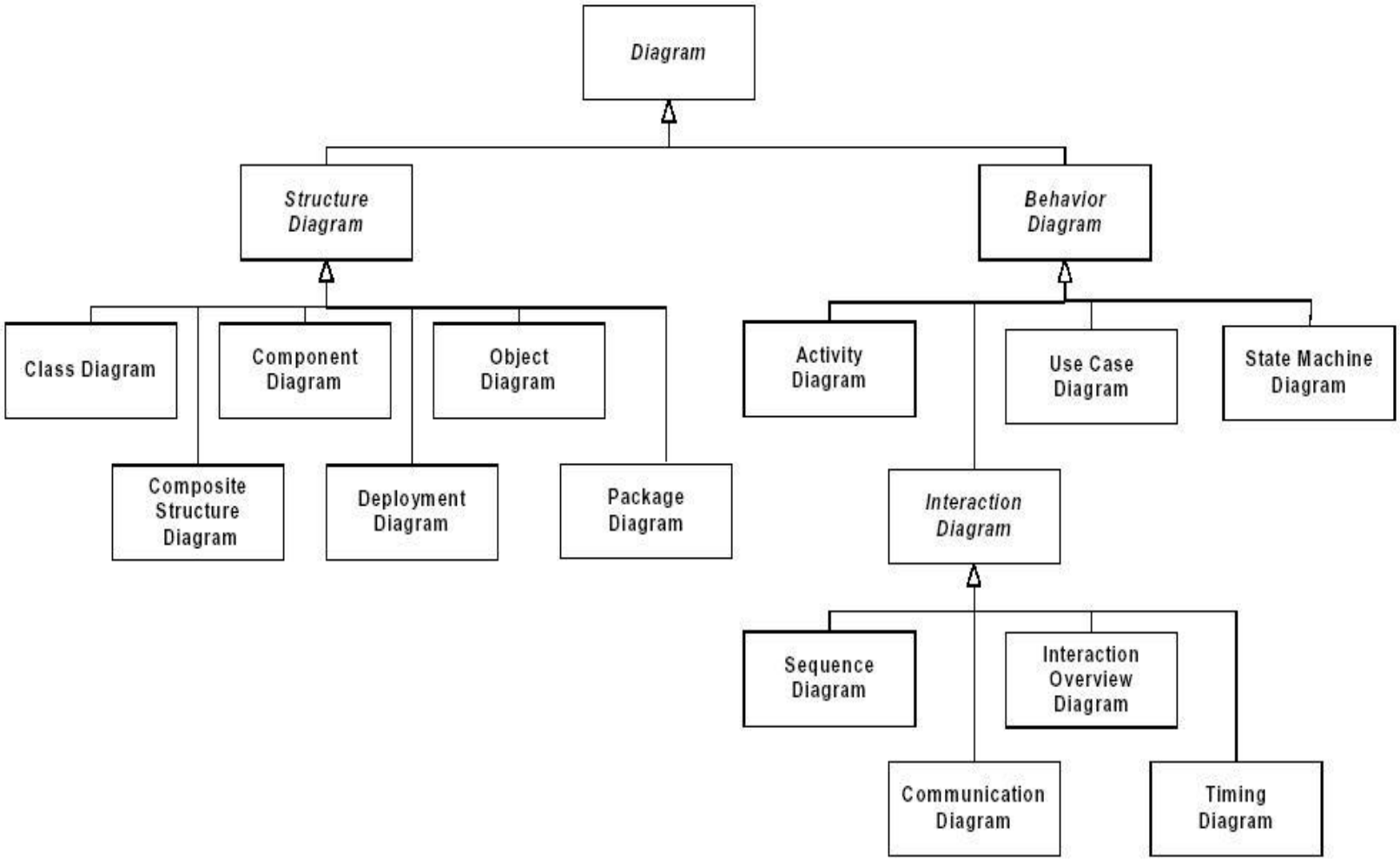
Назначение языка UML

- Предоставить разработчикам легко воспринимаемый и выразительный язык визуального моделирования, специально предназначенный для разработки и документирования моделей сложных систем различного целевого назначения
- Снабдить исходные понятия языка UML возможностью расширения и специализации для более точного представления моделей систем в конкретной предметной области
- Графическое представление моделей в нотации UML не должно зависеть от конкретных языков программирования и инструментальных средств проектирования
- Описание языка UML должно включать в себя семантический базис для понимания общих особенностей ООАП
- Способствовать распространению объектных технологий и поощрять развитие рынка программных инструментальных средств
- Интегрировать в себя новейшие и наилучшие достижения практики ООАП

Классификация моделей в языке UML

- *Структурные модели (structured models)* – модели, предназначенные для описания статической структуры сущностей или элементов некоторой системы, включая их классы, интерфейсы, атрибуты и отношения.
- *Модели поведения (behavioral models)* – модели, предназначенные для описания процесса функционирования элементов системы, включая их методы и взаимодействие между ними, а также процесс изменения состояний отдельных элементов и системы в целом.

Канонические диаграммы языка UML 2.x



Канонические диаграммы языка UML 2.x