Медицинские информационные системы

Модуль 1. Проектирование медицинских информационных систем

Анализ требований ТЗ. Выбор и обоснование подхода к проектированию архитектуры МИС

Иванова Г.С., Ничушкина Т.Н.
Проектирование программного обеспечения: Учебное пособие. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2003. – 104 с.

Ланцберг Анна Вильямовна
К.т.н., доцент кафедры ИУ6 (Компьютерные системы и сети)
lantsberg_av@bmstu.ru
Каб. 801 ГК

Анализ требований к программному обеспечению

Результат анализа – спецификации разрабатываемого ПО:

- Выполняют декомпозицию и содержательную постановку решаемых задач
- Уточняют их взаимодействие и эксплуатационные ограничения

Спецификации – это полное и точное описаний функций и ограничений разрабатываемого ПО.

Функциональные спецификации описывают функции ПО

Эксплуатационные спецификации определяют требования к техническим средствам, надежности, безопасности и т.д.

Требования к функциональным спецификациям

- ***Полнота***: спецификации должны содержать всю существенную информацию и не должны содержать несущественной информации, например, деталей реализации;
- ***Точность***: спецификации должны однозначно восприниматься как заказчиком, так и разработчиком

Точные спецификации можно определить только разработав некоторую ***формальную модель ПО.***

Подходы к разработке программных систем

Объектно-ориентированный

- Ориентирован на комбинирование данных и процедур в унифицированные объекты, а не на моделирование отдельных бизнес-процессов

Функционально-ориентированный (структурный)

- Последовательный, ориентированный на набор определенных бизнес-функций и предполагает разработку «с чистого листа»

Формальные модели этапа анализа и определения спецификаций

- Диаграммы переходов состояний
- Математические модели предметной области

Модели, не зависящие от подхода к разработке

- Функциональные диаграммы
- Диаграммы потоков данных
- Диаграммы отношений компонентов данных

Модели структурного подхода

- Диаграммы вариантов использования
- Контекстные диаграммы классов
- Диаграммы последовательностей
- Диаграммы деятельности

Модели объектного подхода

Функциональные спецификации

Описывают одни и те же характеристики ПО: ***перечень функций*** и ***состав обрабатываемых данных.***

Отличия – система приоритетов и акцентов, используемых разработчиком ПО

Например:

Диаграммы переходов состояний определяют аспекты поведения ПО во времени

Диаграммы потоков данных определяют направление и структуру потоков данных

Концептуальные диаграммы классов определяют отношение между основными понятиями предметной области

Анализ требований. Структурный подход

Основан *на моделировании потоков данных* и использует комплексное представление проектируемого ПО в виде совокупности моделей:

I. Диаграммы потоков данных (*DFD – Data Flow Diagrams*) описывают взаимодействие источников и потребителей информации через процессы, которые должны быть реализованы в системе
II. Диаграммы «сущность-связь» (*ERD – Entity-Relationship Diagrams*) описывают базы данных разрабатываемой системы
III. Диаграммы переходов состояний (*STD – State Transition Diagrams*) характеризуют поведение системы во времени
IV. Спецификации процессов
V. Словарь данных

Структурный подход (спецификация ПО)

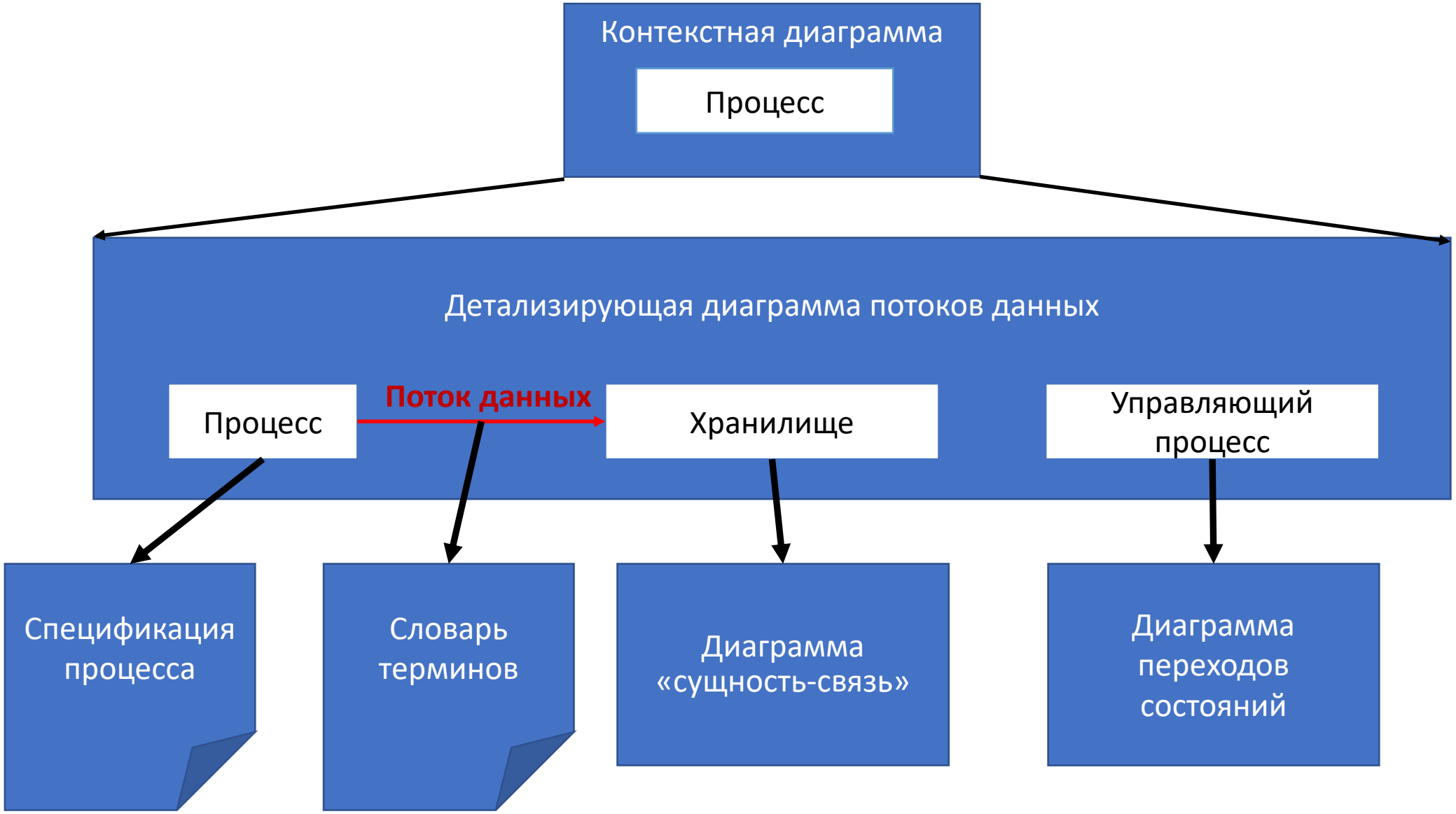


Диаграмма переходов состояний

Демонстрирует поведение программной системы при получении управляющих воздействий

Управляющие воздействия – это управляющая информация, получаемая системой извне, например, команды пользователя и сигналы датчиков, подключенных к системе

Построение диаграммы переходов основано на *теории конечных автоматов*. Необходимо определить:

- Основные состояния
- Управляющие воздействия (или условия перехода)
- Выполняемые действия
- Возможные переходы разрабатываемого ПО

Механизм работы: *при получении управляющего воздействия система должна выполнить определенные действия, а затем, либо остаться в том же состоянии, либо перейти в другое состояние, зафиксировав некоторые изменения в системе*

Различия интерактивного ПО и систем реального времени

Интерактивое ПО

- Основные управляющие действия – команды пользователей
- Характерно получение команд различных типов

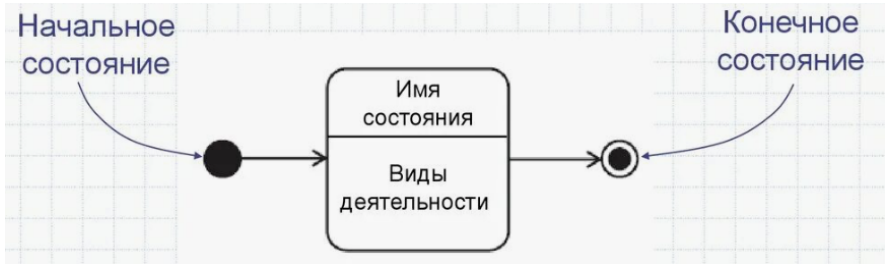
Общим для интерактивного ПО и систем реального времени является наличие состояния ожидания, когда система приостанавливает работу до получения очередного управляющего воздействия

Системы реального времени

- Основные управляющие действия – сигналы от датчиков и/или оператора производственного процесса
- Характерно получение однотипных сигналов
- Установлено жесткое ограничение на время обработки дополнительного сигнала
- Необходимо дополнительное исследование поведения системы во времени, например, с использованием сетей Петри или марковских процессов

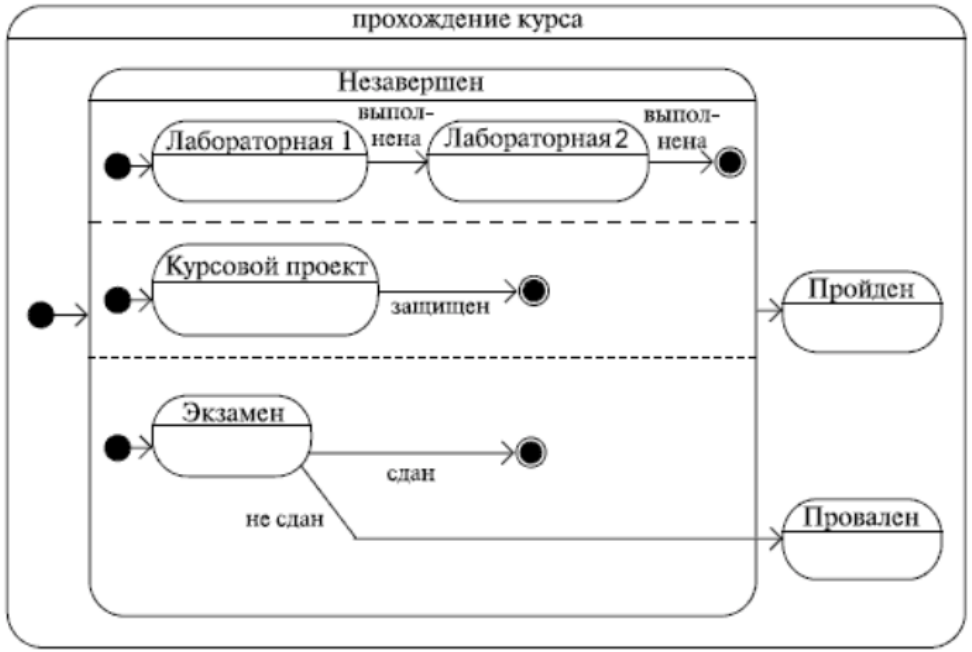
Диаграмма переходов состояний

Обозначения (для UML)



Пример (установка таймера)

Пример (Прохождение курса)

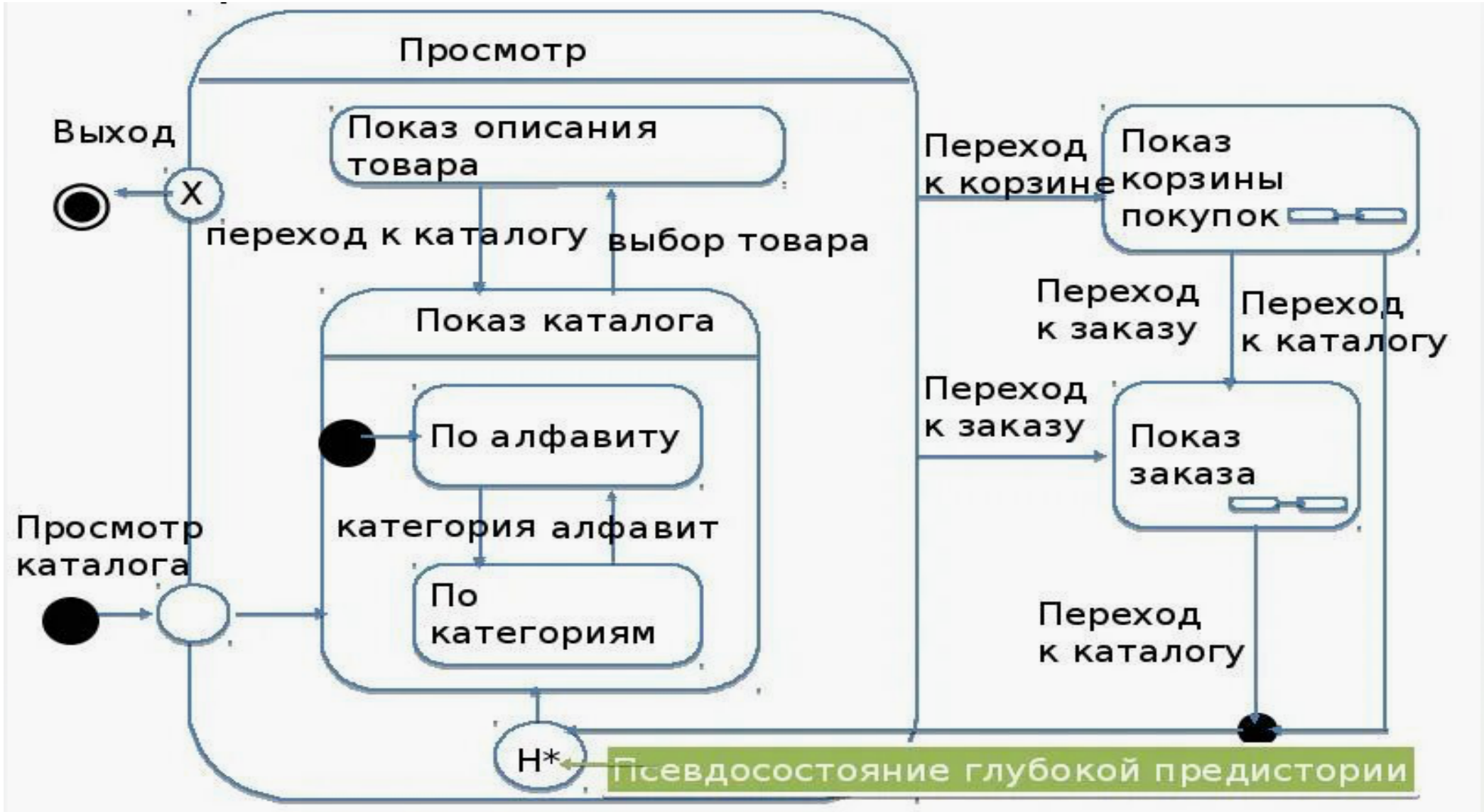


Параллельные процессы обработки данных

Вложенные процедуры

Диаграмма переходов состояний

Пример (Просмотр каталога товаров)



Проектирование ПО. Структурный подход

Структурная схема ПО отражает состав и взаимодействие по управлению частей ПО

Структурная схема показывает наличие подсистем и других структурных компонентов (программы, подсистемы, базы данных, библиотеки ресурсов)

Разработка структурной схемы выполняется *методом пошаговой детализации*

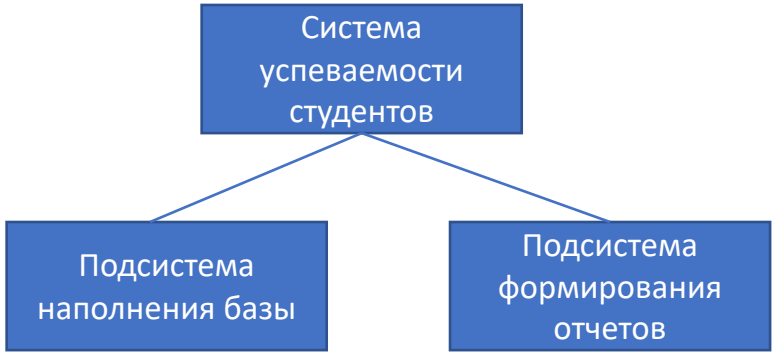
Функциональная схема ПО (регламентирована ГОСТ 19.701-90) отражает схему взаимодействия компонентов ПО с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств

Важно!

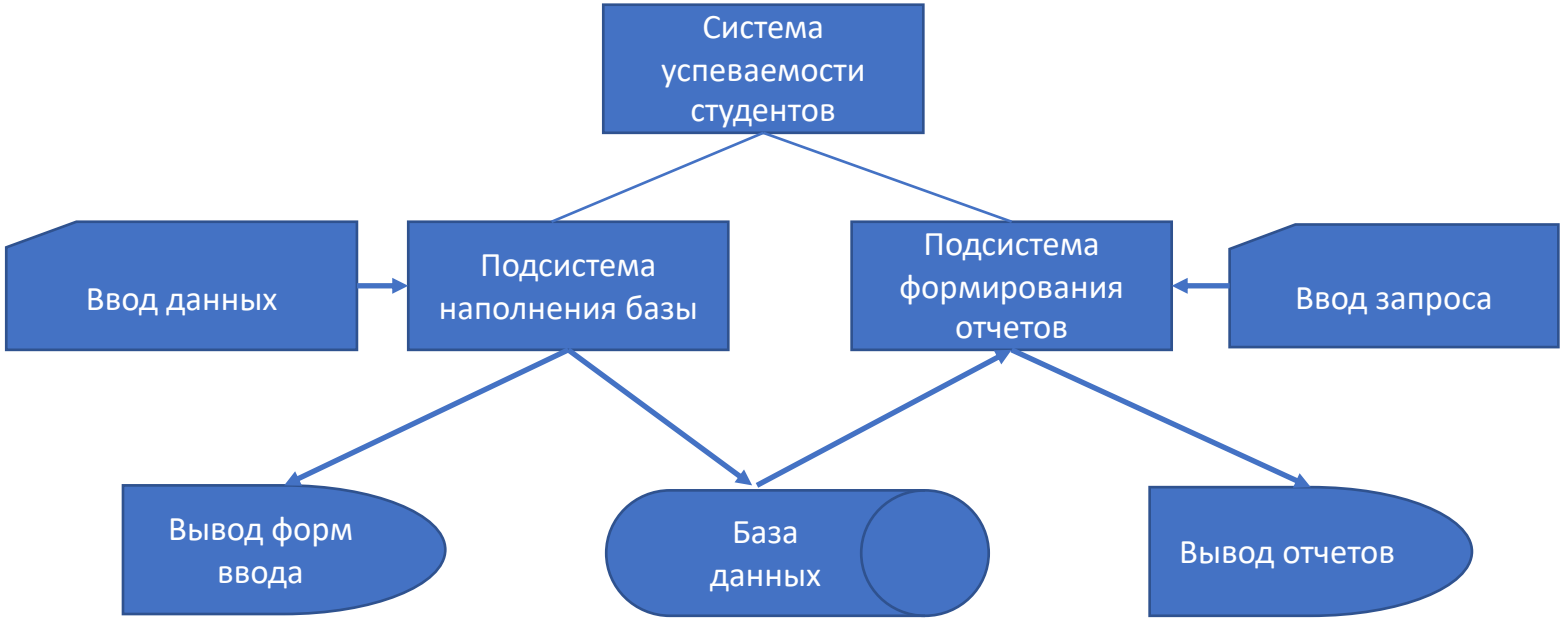
- Все компоненты структурных и функциональных схем должны быть описаны
- При структурном подходе особенно тщательно необходимо прорабатывать спецификации межпрограммных интерфейсов

Проектирование ПО. Структурный подход

Структурная схема программной системы (успеваемость студентов)

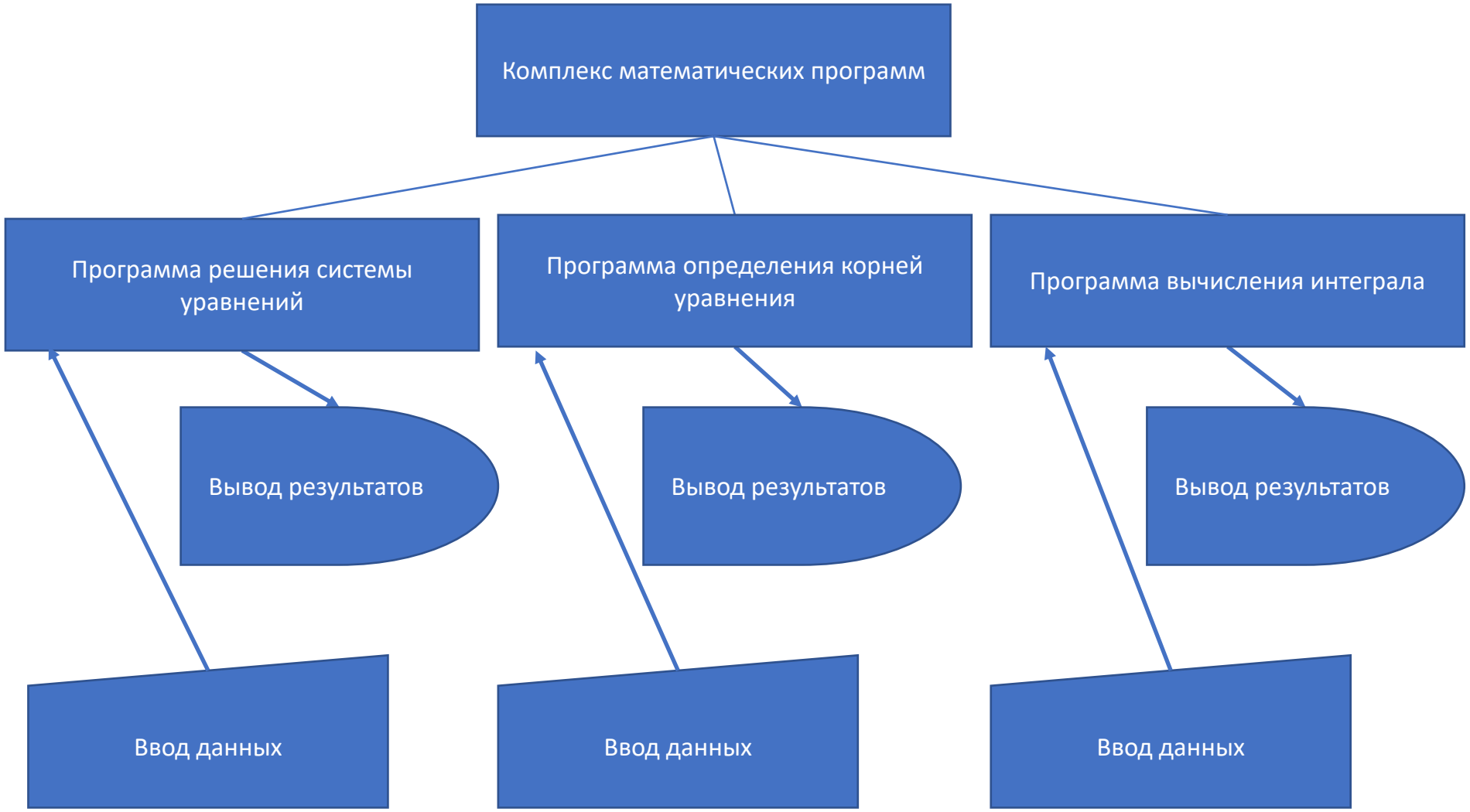


Функциональная схема программной системы (успеваемость студентов)



Проектирование ПО. Структурный подход

Функциональная схема комплекса программ



Метод пошаговой детализации

Важно! В первую очередь детализируются управляющие процессы декомпозируемого компонента, уточнение операций остается «на потом»

Целесообразно придерживаться следующих правил:

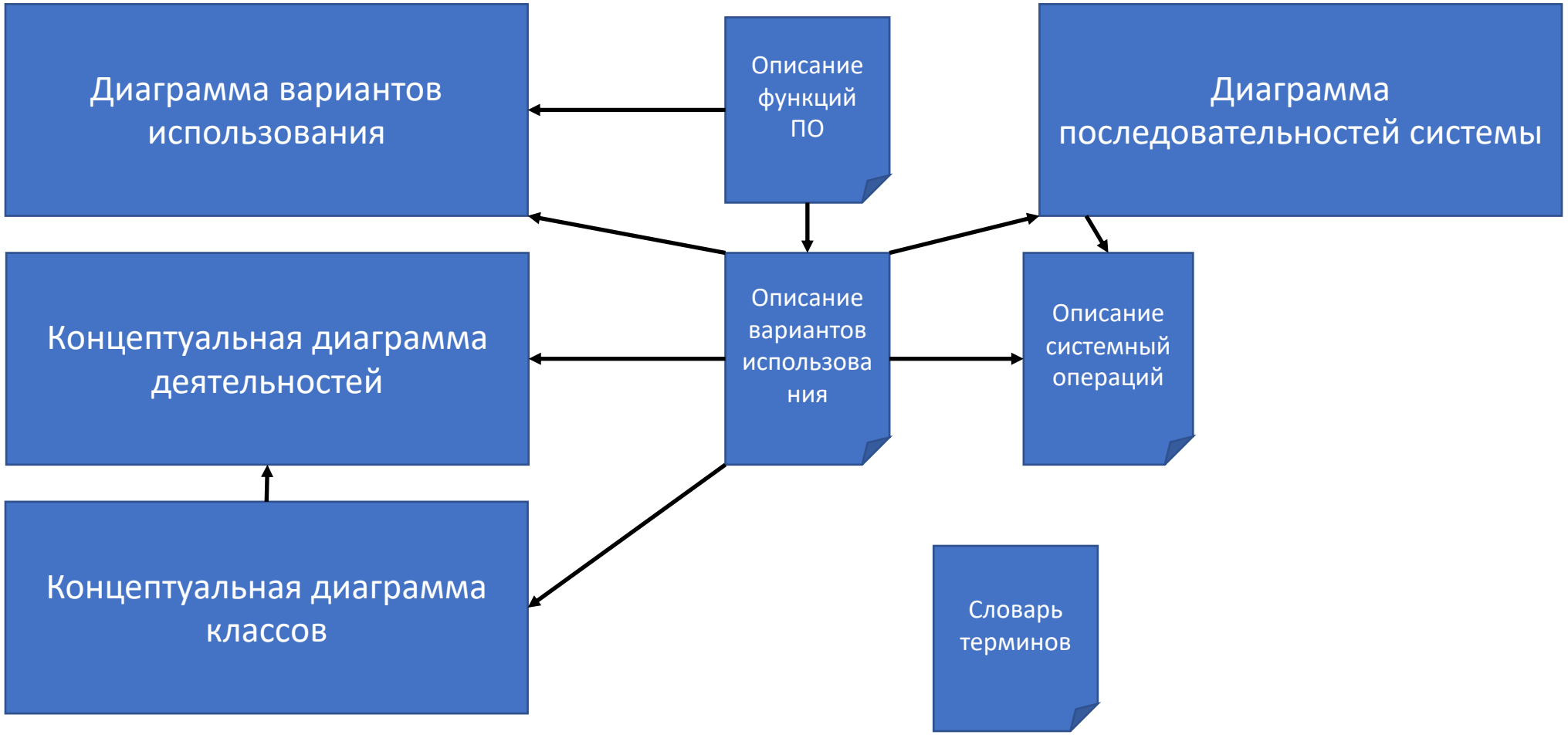
- ✓ не отделять операции инициализации и завершения от соответствующей обработки, так как модули инициализации и завершения имеют плохую связность (временную) и сильное сцепление (по управлению);
- ✓ не проектировать сильно специализированные (или универсальные) модули, так как проектирование излишне специализированных модулей увеличивает их количество, а излишне универсальных – их сложность;
- ✓ избегать дублирования действий в различных модулях, так как при их изменении исправления придется вносить везде, где они выполняются; в этом случае целесообразно реализовать данные действия в отдельном модуле;
- ✓ группировать сообщения об ошибках в один модуль по типу библиотеки ресурсов, тогда будет легче согласовать формулировки, избежать дублирования сообщений, а также перевести сообщения на другой язык.

Анализ требований. Объектно-ориентированный подход. UML

Полное описание объектно-ориентированной модели на языке UML содержит несколько моделей, каждая из которых характеризует один из аспектов проектируемой системы:

- *модель использования* – описание функциональности ПО с точки зрения пользователя;
- *логическая модель* – описание ключевых абстракций ПО (классов, интерфейсов,…) – средств, обеспечивающих функциональность;
- *модель реализации* – описывает организацию программных модулей;
- *модель процессов* – отображает организацию вычислений и оперирует понятиями процессы и нити (позволяет оценить производительность, масштабируемость и надежность ПО)
- *модель развертывания* – показывает особенности размещения программных компонентов на конкретном оборудовании

Анализ требований. Объектно-ориентированный подход. Спецификация ПО



Анализ требований. Объектно-ориентированный подход. UML - диаграммы

Диаграмма прецендентов (вариантов) использования	• Отображает функциональность ИС в виде набора выполняющихся последовательностей транзакций
Диаграмма классов объектов	• Отображает структуру совокупности взаимосвязанных классов объектов наподобие ER-диаграмм, относящихся к структурному подходу
Диаграммы состояний	• Отображают динамику состояний объектов одного класса и связанных с ними событий
Диаграммы взаимодействия объектов	• Отображают динамическое взаимодействие объектов в рамках одного прецедента использования
Диаграммы деятельностей	• Отображают потоки работ во взаимосвязанных прецедентах (могут быть разбиты на более детальные диаграммы)
Диаграммы пакетов	• Отображают распределение объектов по функциональным или обеспечивающим подсистемам (могут быть разбиты на более детальные диаграммы)
Диаграмма компонентов	• Отображает физические модули программного кода
Диаграмма размещения	• Отображает распределение объектов по узлам вычислительной сети

* Это диаграммы концептуального уровня проектирования систем

Анализ требований. Объектно-ориентированный подход. Определение вариантов использования

Необходимо:

I. Выявить внешних пользователей разрабатываемого ПО

II. Перечень аспектов его поведения в процессе взаимодействия с конкретным пользователем (это варианты или прецеденты)

Вариант использования – это характерная процедура применения разрабатываемой системы конкретным действующим лицом, в качестве которого выступают, как люди, так и другие системы и устройства

Каждый вариант использования связан с некоторой **целью**.

В зависимости от **цели** выполнения процедуры различают следующие варианты использования:

- Основные – обеспечивают требуемую функциональность разрабатываемого ПО;
- Вспомогательные – обеспечивают выполнение необходимых настроек системы и ее обслуживание;
- Дополнительные - обеспечивают дополнительные удобства для пользователя

Анализ требований. Объектно-ориентированный подход. Краткая форма варианта использования

Краткая форма содержит: название варианта использования, цель, действующих лиц, тип варианта использования (основной, вспомогательный или дополнительный) и его краткое описание.

Пример: вариант использования «Выполнение задания»

Название варианта	**Выполнение задания**
Цель	Получение результатов решения задачи
Действующие лица	Пользователь
Краткое описание	Решение задачи предполагает выбор задачи, выбор алгоритма, задание данных и получение результатов решения
Тип	Основной

Анализ требований. Объектно-ориентированный подход. Подробная форма варианта использования

Подробная форма содержит: всю информацию краткой формы, а также описание типичного хода событий в виде диалога между пользователями и системой и возможных альтернатив

Пример: вариант использования «Выполнение задания»

Типичный ход событий

Действия пользователя	Отклик системы
1. Пользователь инициирует новое задание	2.Система регистрирует новое задание и предлагает список типовых задач
3. Пользователь выбирает тип задачи	4. Система регистрирует тип задачи и предлагает список способов задания данных
5. Пользователь выбирает способ задания данных: а) если выбран ввод с клавиатуры, см. раздел Ввод данных; б) если выбран ввод из базы данных, см. раздел Выбор данных из базы	6. Система регистрирует данные и предлагает список алгоритмов решения
7. Пользователь выбирает алгоритм	8. Система регистрирует алгоритм и предлагает начать решение
9. Пользователь инициирует процесс решения	10. Система запускает подпрограмму решения задачи
11. Пользователь ожидает	12. Система демонстрирует результаты и предлагает сохранить их в базе
13. Пользователь анализирует результаты и выбирает, сохранять их в базе или нет	14. Если выбрано сохранение данных, то система выполняет запись данных задания в базу
	15. Система переходит в состояние ожидания

Альтернатива

11. Если время выполнения программы с точки зрения пользователя велико, то он прерывает процесс выполнения

12. Система прерывает расчеты, предлагает список алгоритмов решения и возвращается на шаг 7

Дополнительная информация:

А. Необходимо обеспечить произвольную последовательность выбора типа задачи, данных и алгоритма

Б. Необходимо обеспечить возможность выхода из варианта на лбом этапе

Анализ требований. Объектно-ориентированный подход. Диаграмма варианта использования. Пример 1

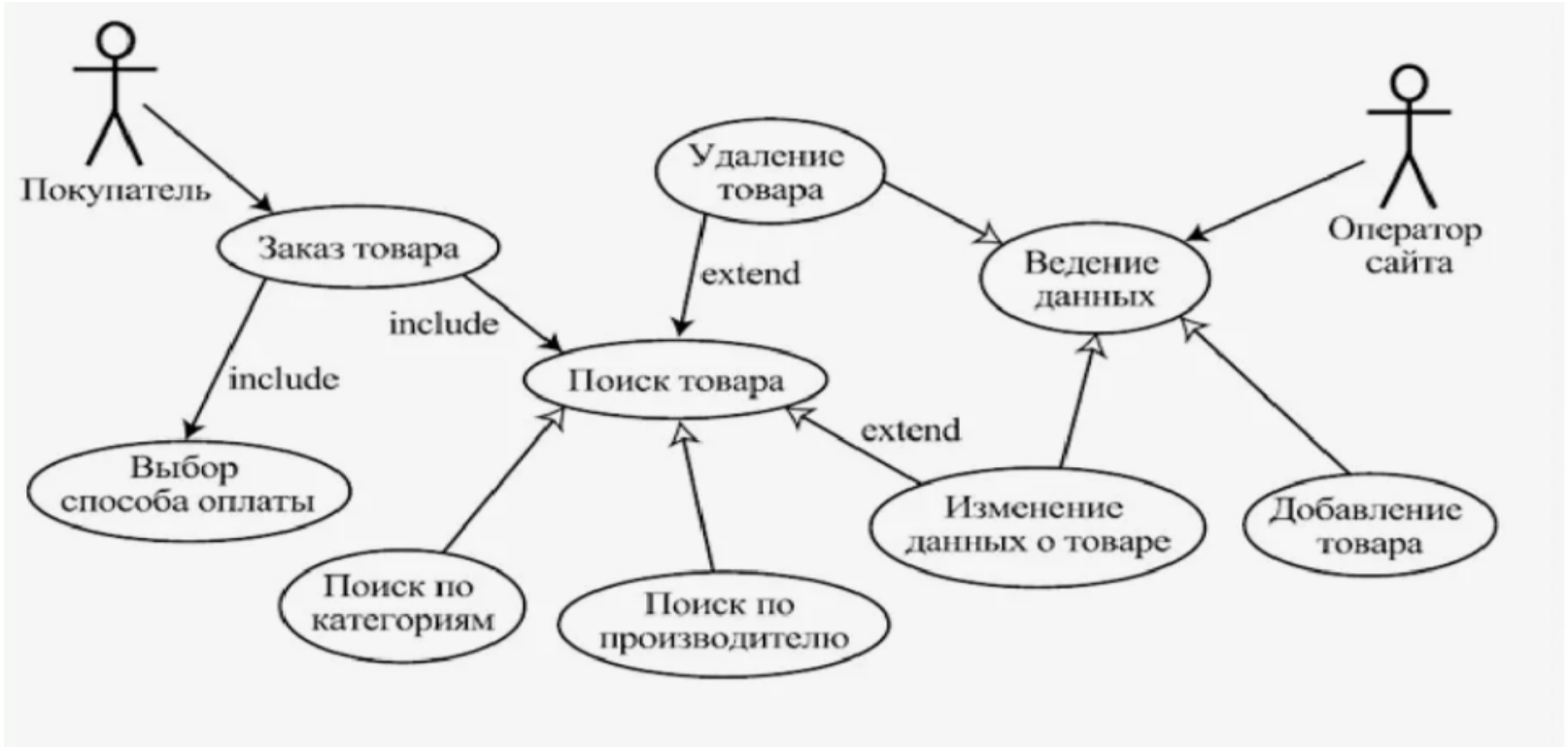
Действующее лицо – внешняя по отношению к разрабатываемому ПО сущность, которая взаимодействует с ним с целью получения или предоставления какой-либо информации

Вариант использования – некоторая очевидная для действующего лица процедура, решающая конкретную задачу

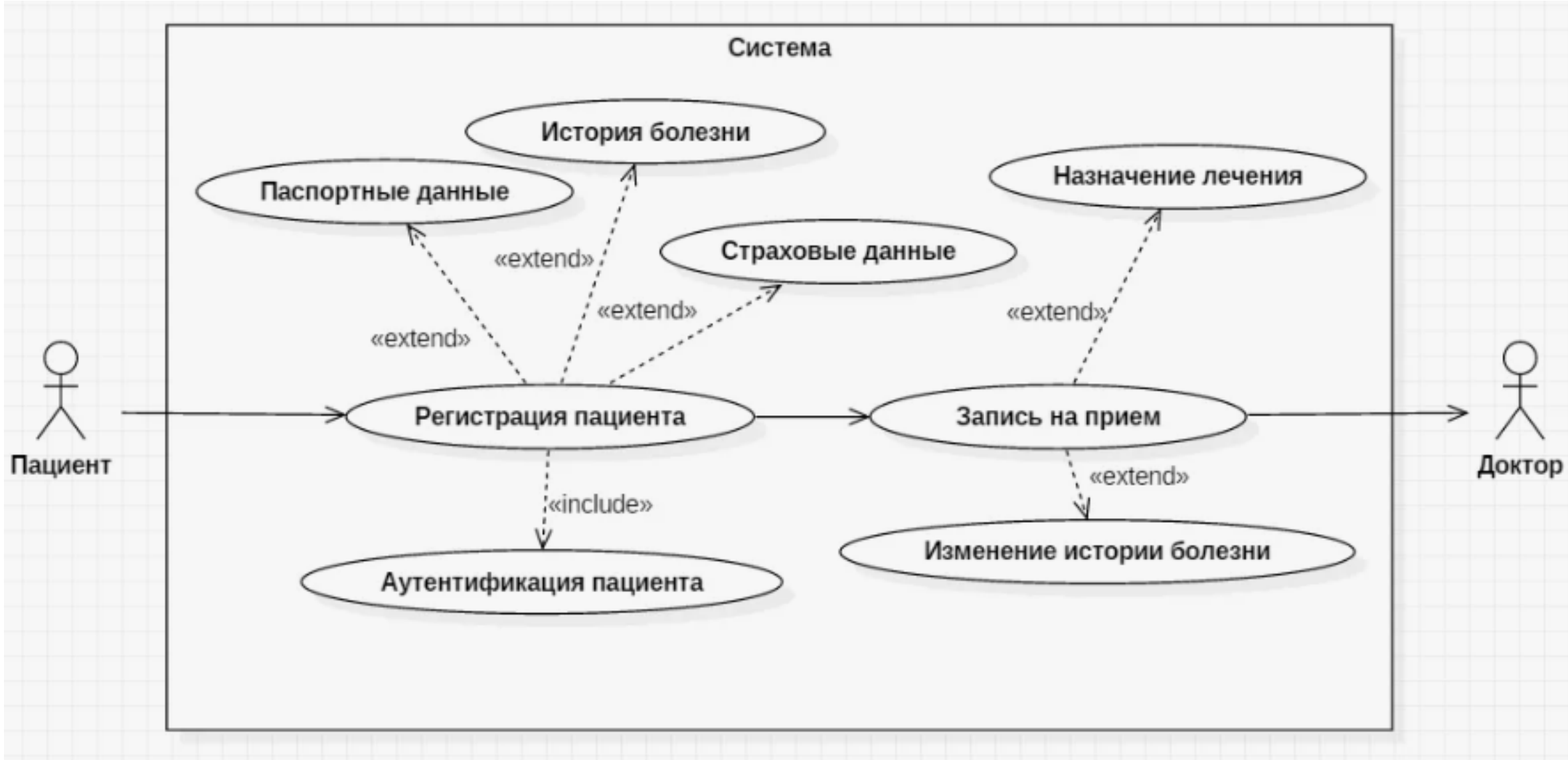
Связь – взаимодействие действующих лиц и соответствующих вариантов использования

Использование применяют, когда существует некоторый фрагмент поведения разрабатываемого ПО, который повторяется в нескольких вариантах использования.

Расширение применяют, если имеются два подобных варианта использования, различающиеся присутствием в одном из них некоторых дополнительных действий



Анализ требований. Объектно-ориентированный подход. Диаграмма варианта использования. Пример 2



Построение концептуальной модели предметной области. Объектно-ориентированный подход.

Диаграмма классов – основа объектно-ориентированного подхода.

В UML используется три уровня диаграмм классов в зависимости от степени их детализации:

I. **Концептуальный** уровень – диаграммы классов (называются **контекстными**) демонстрируют связи между основными понятиями предметной области (*используются на этапе анализа*)

II. Уровень **спецификаций** – диаграммы классов отображают **интерфейсы** классов предметной области, т.е. связи объектов этих классов (*используются на этапе проектирования*)

III. Уровень **реализации** – диаграмм классов показывают **поля и методы** конкретных классов (*используются на этапе реализации*)

Построение концептуальной модели предметной области. Объектно-ориентированный подход.

Основные понятия:

Класс – совокупность общих признаков некоторой группы объектов предметной области (может обладать общими и собственными признаками, например, класс «студент»)

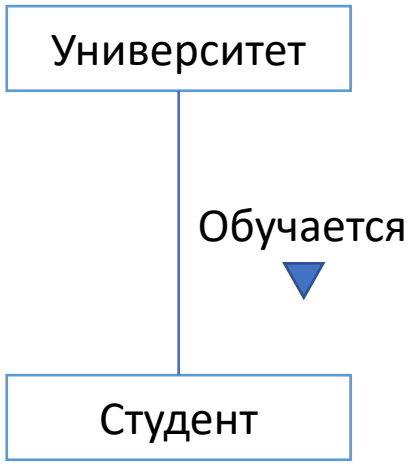
Атрибут – некоторый существенный с точки зрения решаемой задачи характеристика объекта (например, имя или номер зачетной книжки студента)

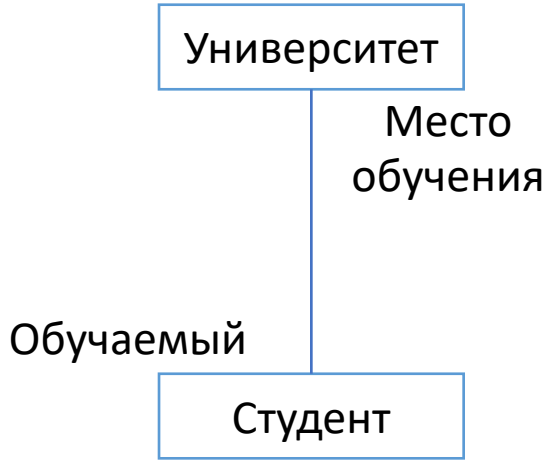
Отношение – статическая, т.е. независящая от времени, связь классов. Различают: отношение ассоциации и отношение обобщения

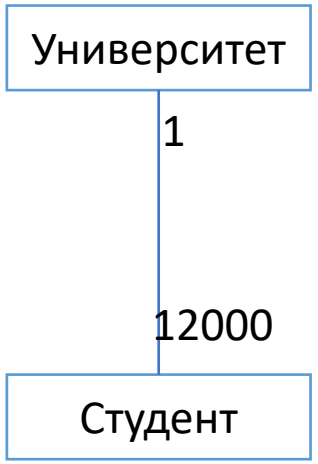
- *Отношение ассоциации* – наличие связи между экземплярами классов или объектами (например, класс «студент» ассоциирован с классом «университет»). Ассоциация может иметь ***имя*** (например, «обучается»). Рядом с именем ставят ***стрелку***, указывающую направление чтения (например, «Студент обучается в университете»). Связи между объектами подразумевают ***роли***, которые объекты играют в отношении друг друга, роль связана с ***направлением ассоциации*** (например, роль университета «Место учебы», а роль студента «Обучаемый»). Название роли может совпадать с именем класса. Роль обладает характеристикой ***множественности***.
- Отношение обобщения – такое отношение между классами, при котором любой объект одного класса (*подтипа*) обязательно является также и объектом другого класса (*супертипа*) (если студент Петров С.П. является объектом подтипа «Студент первого курса» супертипа «Студент», то он является объектом этого супертипа)

Построение концептуальной модели предметной области. Объектно-ориентированный подход.

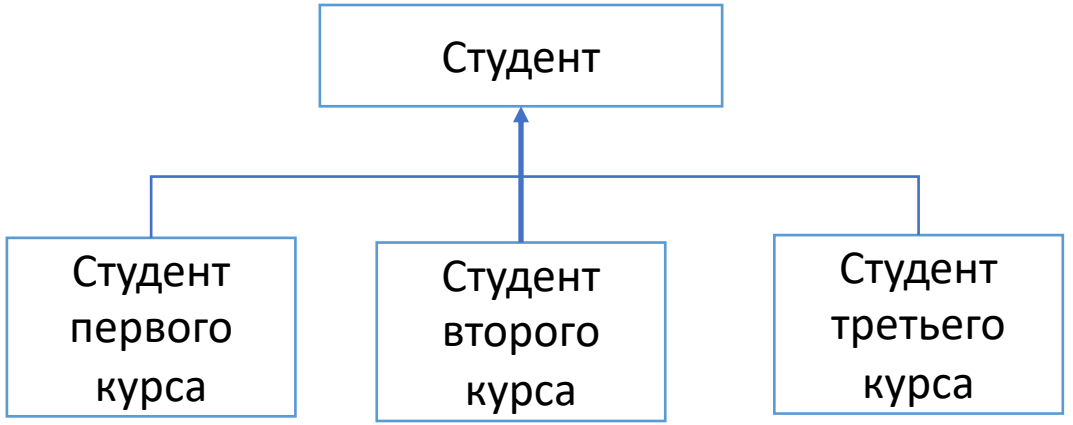
Отношение ассоциации



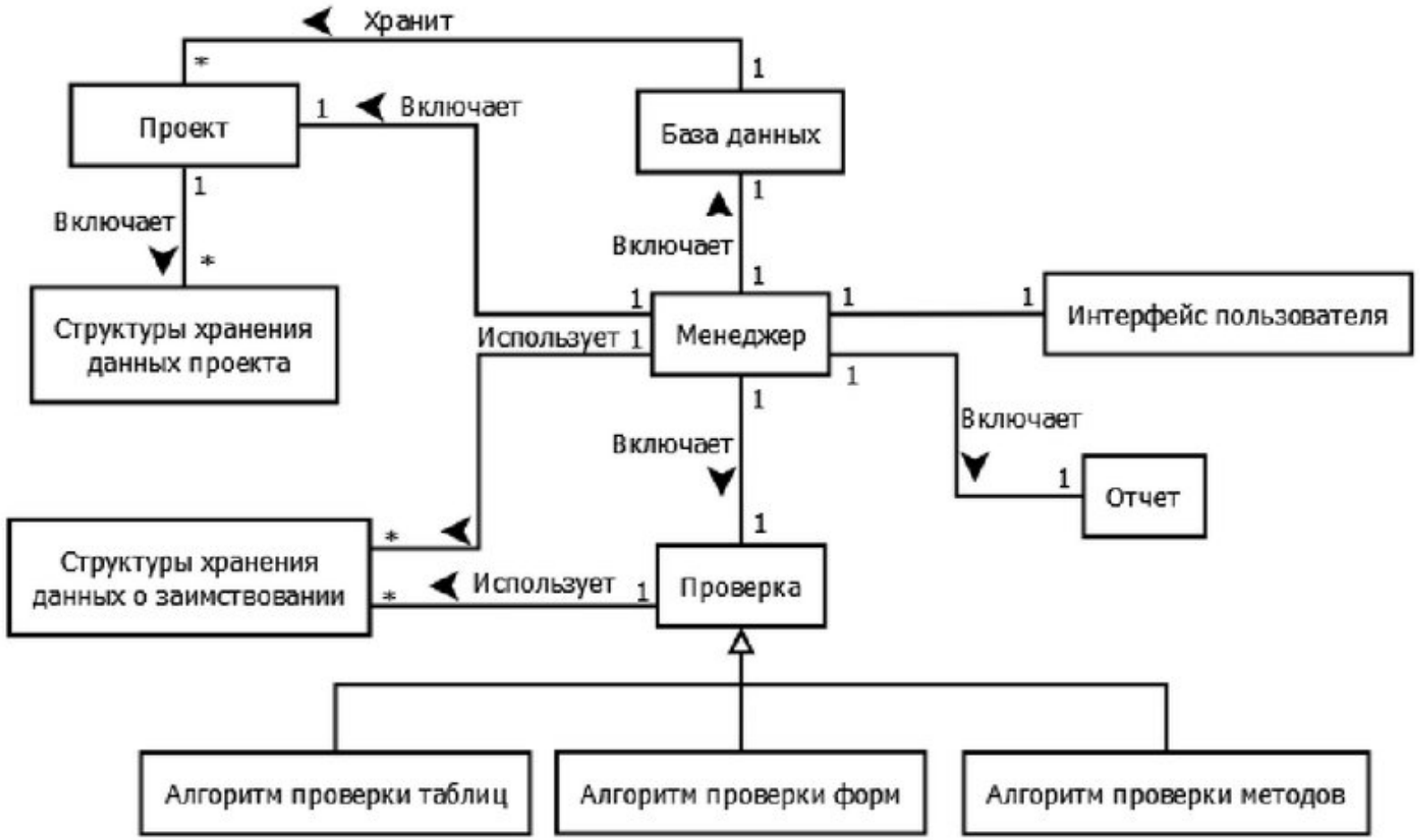




Отношение обобщения



Построение концептуальной модели предметной области. Объектно-ориентированный подход. Пример контекстной диаграммы классов



Описание поведения. Объектно-ориентированный подход.

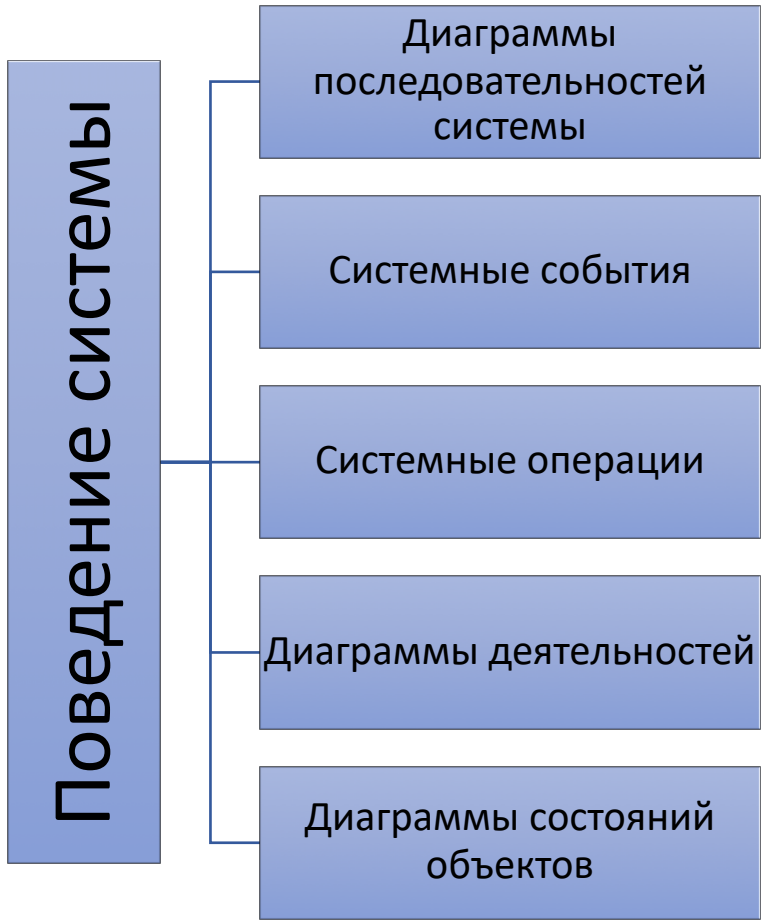


Диаграмма последовательностей системы – графическая модель, которая для определенного сценария варианта использования показывает генерируемые действующими лицами события и их порядок.

Необходимо:

- Система представляется в виде «черного ящика», для системы рисуется линия жизни»
- Идентифицируется каждое действующее лицо и изображается для него линия жизни
- Из описания варианта использования определяется множество системный событий и их последовательность
- Изображаются системные события в виде линий со стрелкой на конце между линиями жизни действующих лиц и системы, а также указывается имя события и список передаваемых значений

События, которые генерируются для системы действующими лицами, называются *системными*.

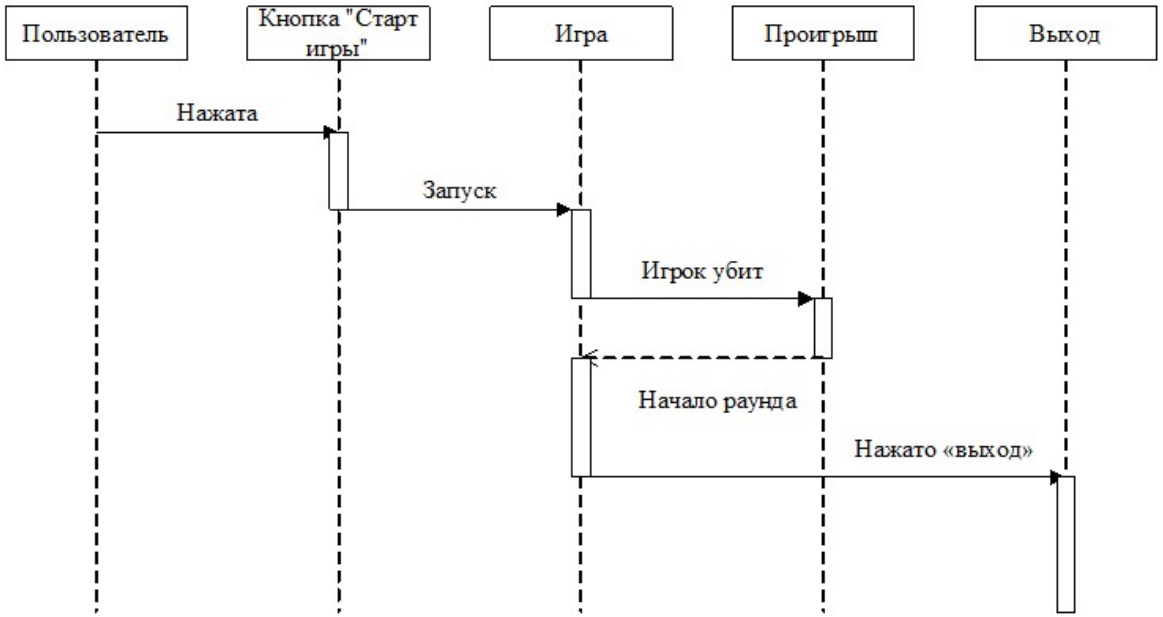
Они инициируют выполнение множества операций, также называемых *системными*

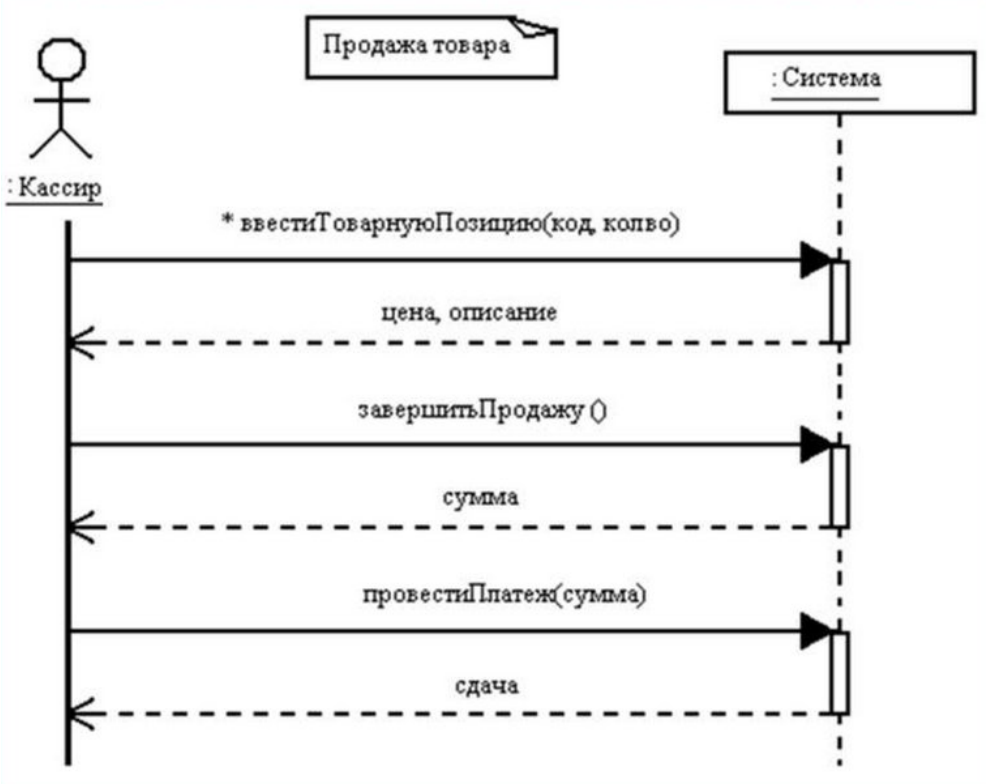
Каждая системная операция называется по имени соответствующего события

Системные операции изображают в виде операций абстрактного класса (типа) System

Каждая системная операция должна быть описана

Описание поведения. Диаграмма последовательностей системы. Объектно-ориентированный подход.





Описание поведения. Диаграмма деятельности. Объектно-ориентированный подход.

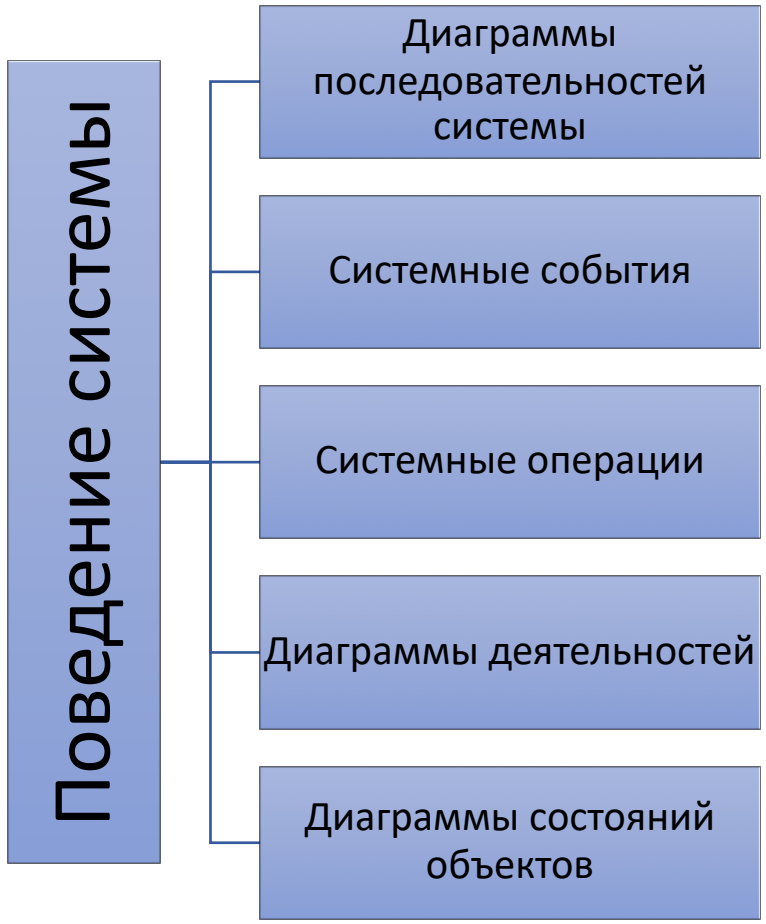


Диаграмма деятельности - обобщенное представление алгоритма, реализующего анализируемый вариант использования.

Деятельность - задача (операция), которую необходимо выполнить вручную или с помощью средств автоматизации.

Каждому варианту использования соответствует своя последовательность задач.

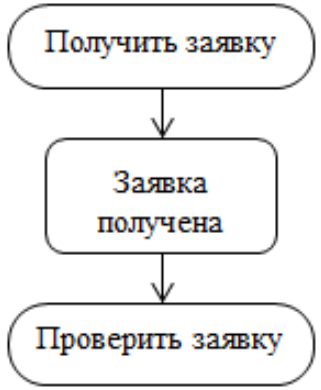
Диаграммы деятельности позволяют описывать альтернативные и параллельные процессы.

На этапе определения спецификаций имеет смысл уточнять только те варианты использования, краткое описание которых недостаточно для понимания сущности решаемых проблем.

Диаграммы деятельности можно использовать вместо вариантов использования или в дополнение к ним

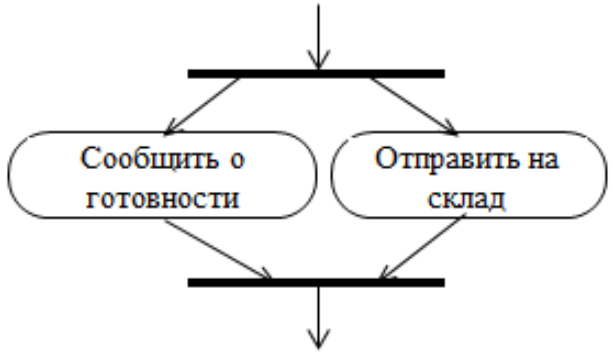
Описание поведения. Диаграмма деятельности. Объектно-ориентированный подход.

Чередование событий и состояний

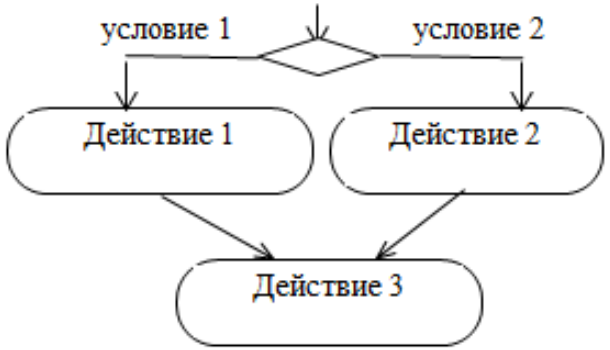


Каждый шаг (действие) переводит прецедент в новое состояние. В свою очередь, новое состояние является стимулом для выполнения следующего шага.
Т.о. прецедент –это ***машина состояний-событий***

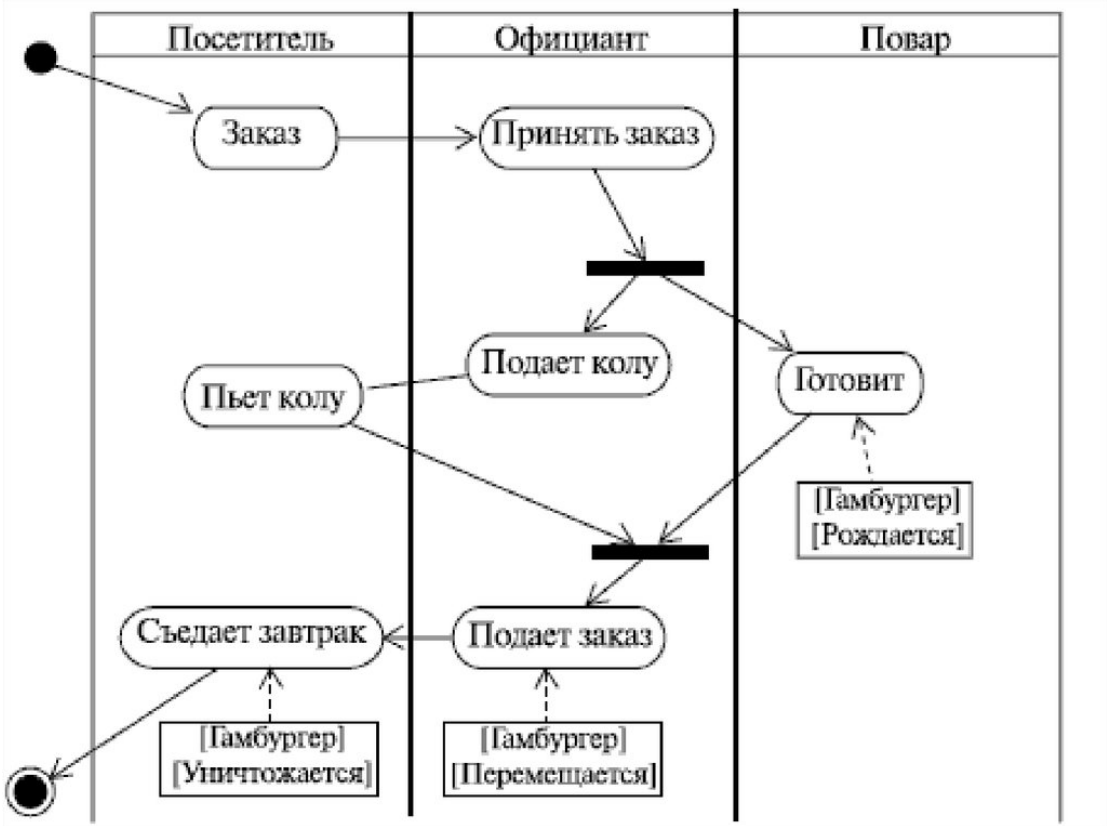
Распараллеливание потока

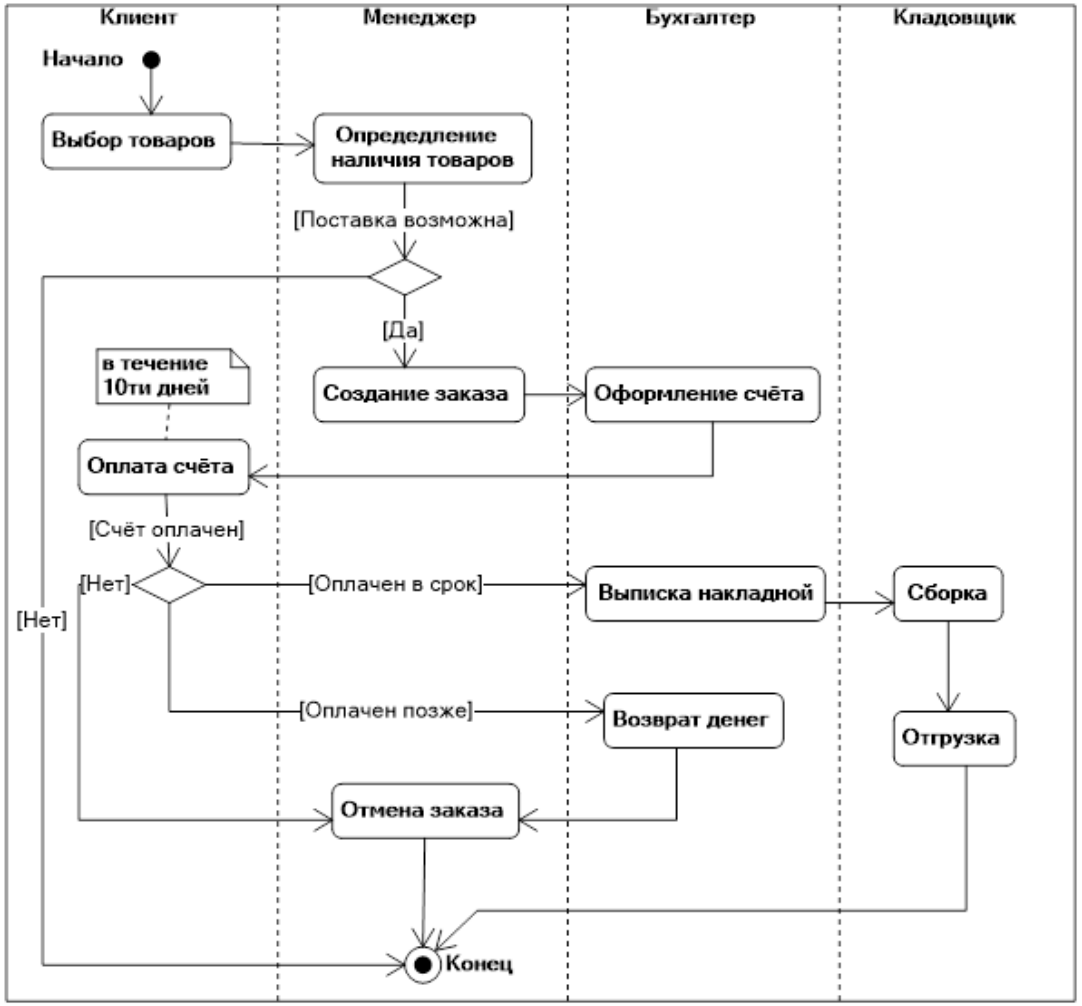


Разветвление потока



Описание поведения. Диаграмма деятельности. Объектно-ориентированный подход.





Проектирование ПО. Объектно-ориентированный подход

Основная задача логического проектирования – разработка классов для реализации объектов, полученных при объектной декомпозиции, в том числе, полное описание полей и методов каждого класса.

Физическое проектирование – проектирование объединения классов и других программных ресурсов в программные компоненты и размещение этих компонентов на конкретных вычислительных установках

Этапы логического и физического проектирования ПО. Объектно-ориентированный подход

I. Выделение классов и пакетов классов

II. Пакет – совокупность описаний классов и других программных ресурсов. Объединение в пакеты используют только для удобства создания больших пакетов, количество классов в которых велико. В один пакет собираются классы и другие ресурсы единого назначения

III. Определение отношений между объектами (используют *диаграммы последовательностей* этапа проектирования и диаграммы кооперации)

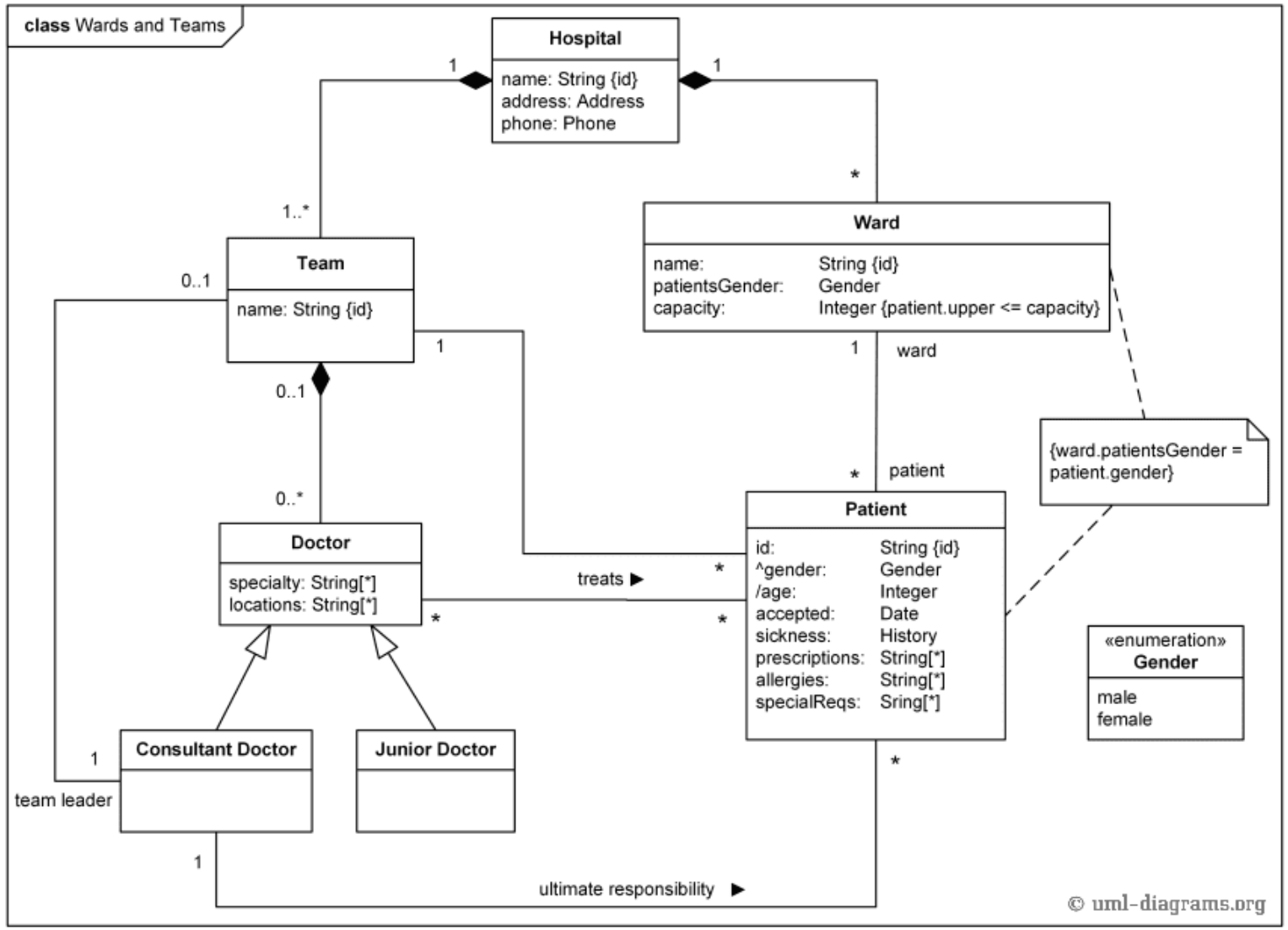
IV. Уточнение отношений классов (используют *диаграммы классов*): кроме ассоциации и обобщения используют также агрегацию и композицию

V. Проектирование классов (используют *диаграммы состояний объекта, диаграммы последовательностей действий (*для проектирования методов класса, алгоритм каждого метода прорабатывается детально в виде схемы))

VI. Компоновка программных компонентов (используют *диаграммы компонентов*)

VII. Проектирование размещения программных компонентов для распределенных программных систем (используют *диаграммы размещения*)

Пример диаграммы классов. Объектно-ориентированный подход



Пример диаграммы компонентов. Объектно-ориентированный подход