Курс SE201

Введение в программную инженерию

Модуль 11

Виды диаграмм, их назначение, последовательность построения

Бабич А.В., Полтава, Украина, ПГПУ, alexander.babich@rambler.ru

Виды диаграмм, их назначение, последовательность построения

О разновидностях диаграмм и о том, как применение UML вписывается в процесс ООП

© Бабич А.В. 2004

О чем мы узнаем…

- Почему нужно несколько видов диаграмм
- Виды диаграмм
- ООП и последовательность построения диаграмм

Почему нужно несколько видов диаграмм

Понятие диаграммы и модели (Буч)

- ***Диаграмма*** - графическое представление множества элементов. Обычно изображается в виде графа с вершинами (сущностями) и ребрами (отношениями).
- С помощью диаграмм можно визуализировать систему с различных точек зрения.
- ***Моделью*** называется семантически замкнутая абстракция системы
- система рассматривается с разных точек зрения с помощью моделей, многообразные представления которых отображены в форме диаграмм.

Почему нужно несколько видов диаграмм

- Сложную систему можно представить в виде набора небольших и почти независимых моделей
- Ни одна из моделей не является достаточной
- Каждая модель выражает разный уровень абстракции
- Каждая модель соответствует некоторой точке зрения на проектируемую систему

Почему нужно несколько видов диаграмм

- Диаграммы – лишь средство визуализации модели
- Одна отдельная диаграмма не является моделью
- Лишь ***набор*** диаграмм составляет модель системы и наиболее полно ее описывает

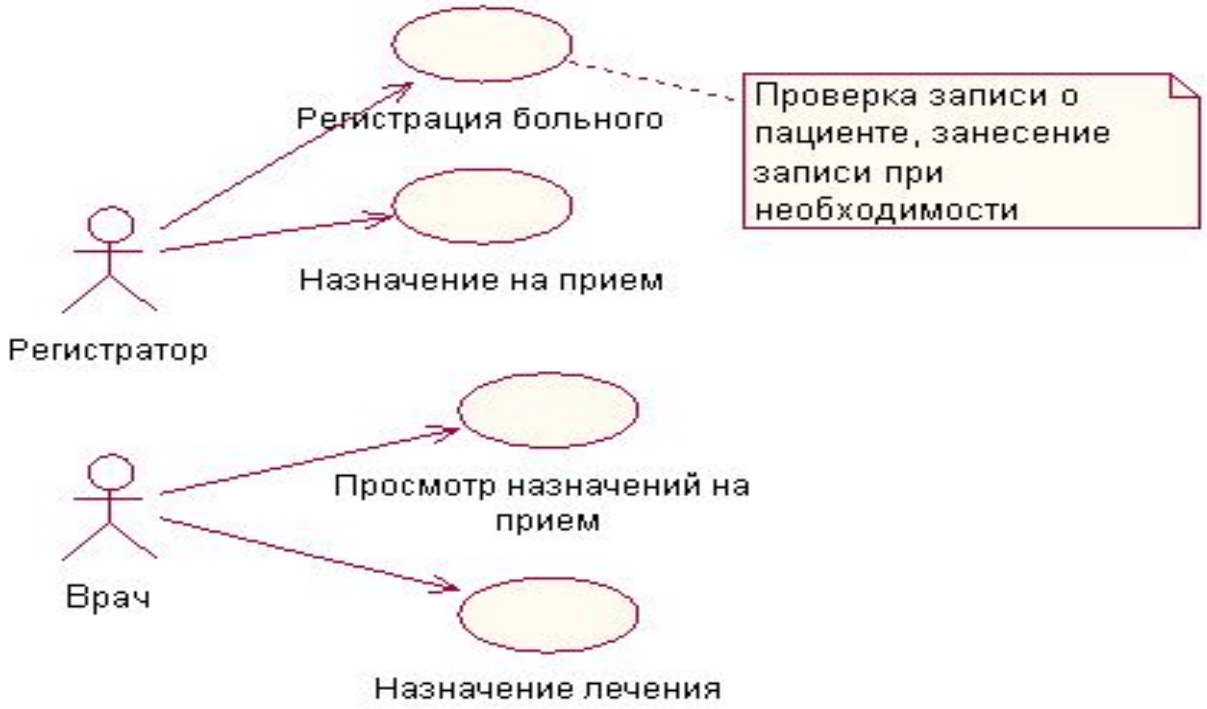
Виды диаграмм

Виды диаграмм

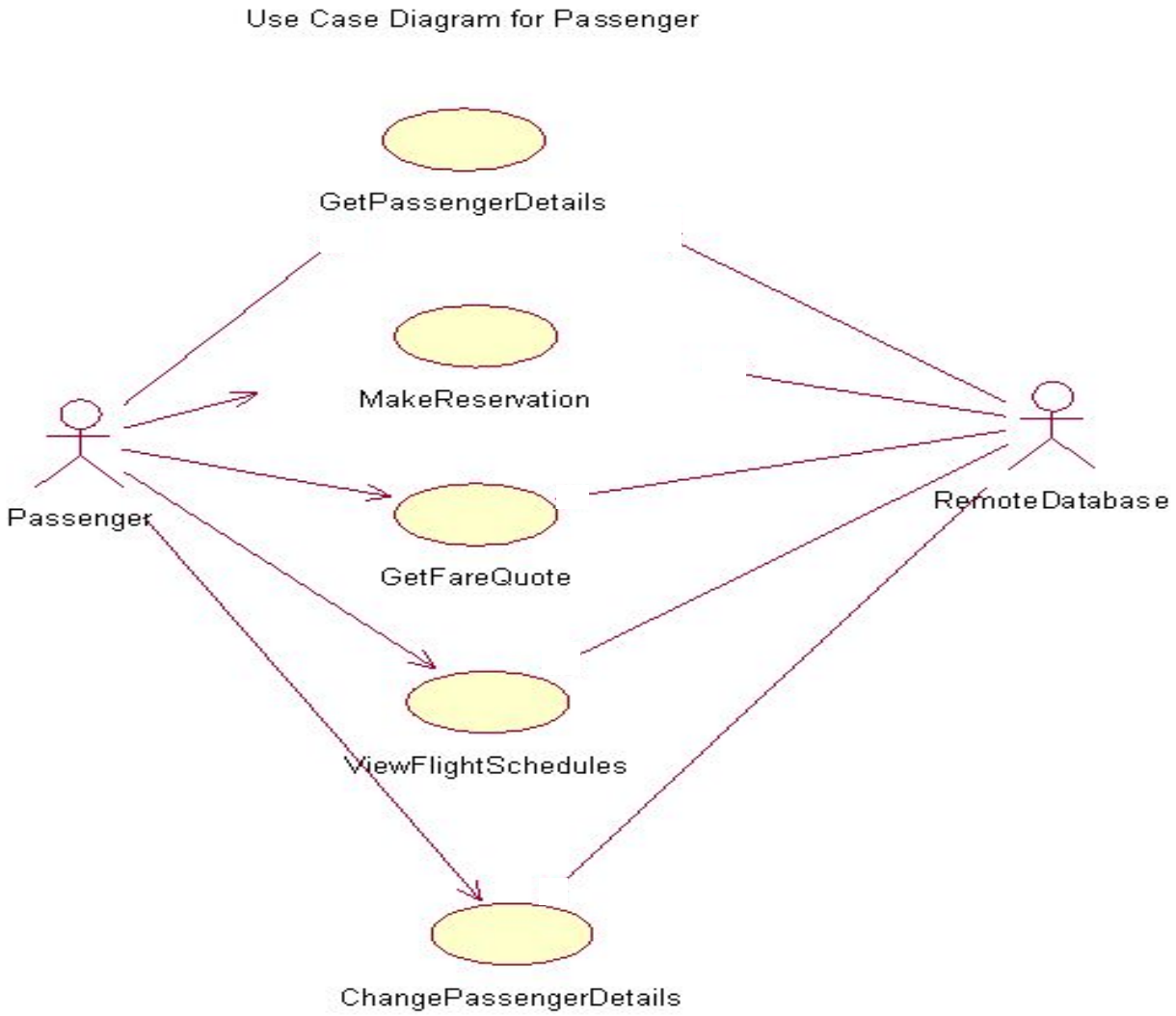
- Виды диаграмм:
  - диаграмма прецедентов
  - диаграмма классов
  - диаграмма объектов
  - диаграмма последовательностей
  - диаграмма взаимодействия
  - диаграмма состояний
  - диаграмма активности
  - диаграмма развертывания
  - …
- Не всегда нужно строить все диаграммы
- Правильный выбор нужных диаграмм позволит сформулировать вопросы о системе и выявить ее «скользкие» моменты, которые необходимо учесть

Диаграмма прецедентов (use case diagram)

- Прецедент (use case) – описание отдельного аспекта поведения системы с точки зрения пользователя (Буч)
- Прецедент позволяет сформировать пользовательские требования к системе
- Пример:



Пример диаграммы прецедентов



Пример диаграммы прецедентов

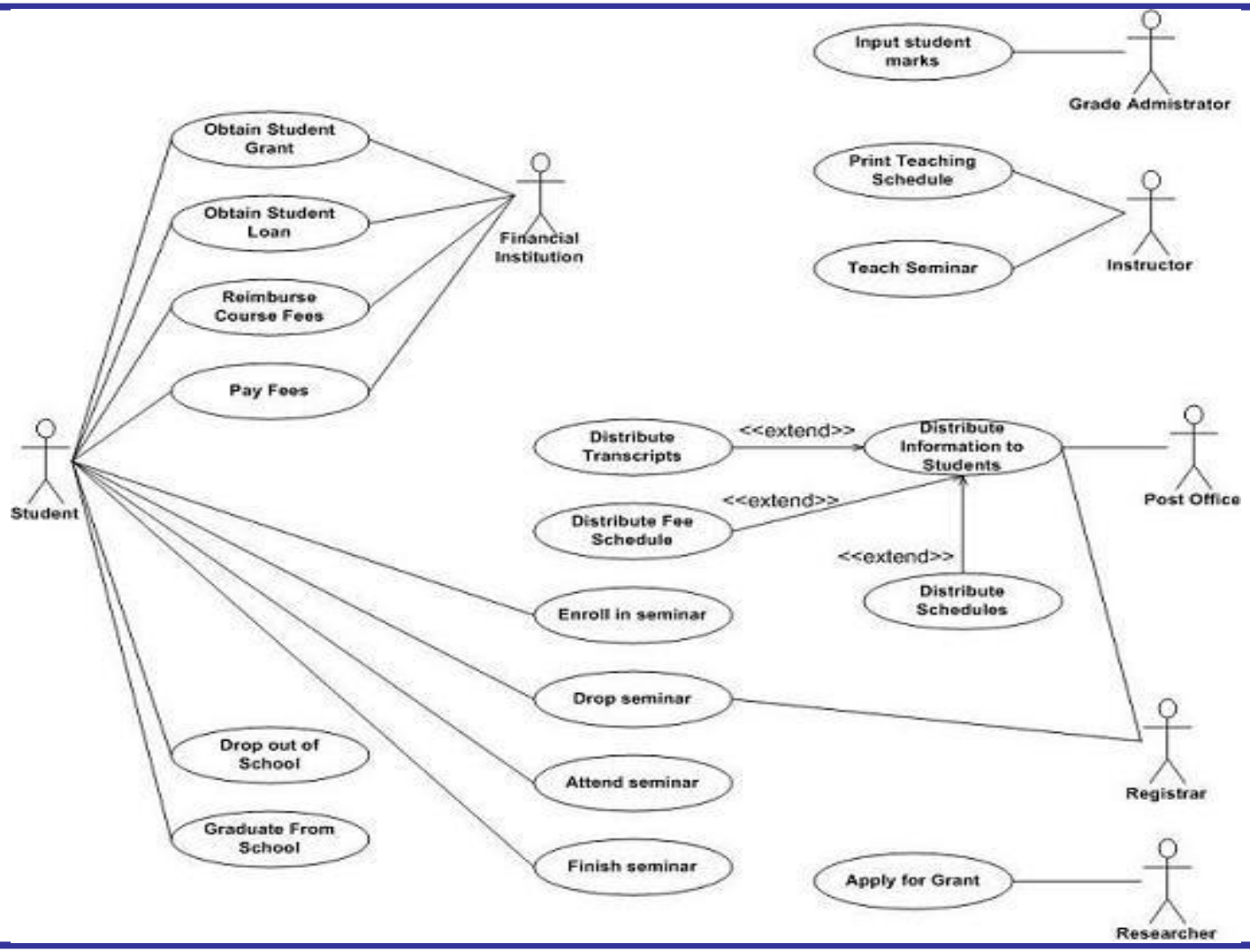
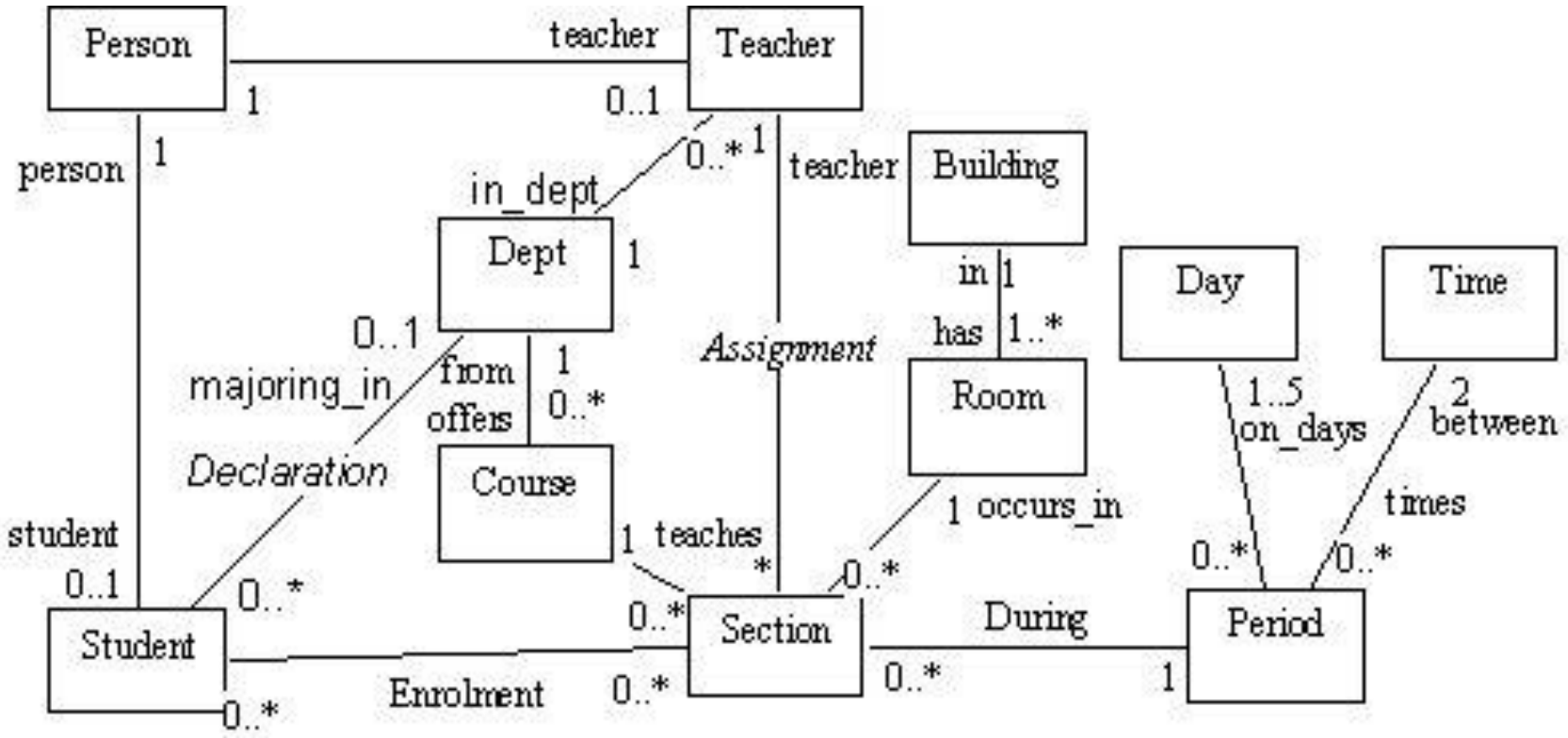


Диаграмма классов (class diagram)

- Класс – категория вещей, которые имеют общие атрибуты и операции (Буч)
- Диаграмма классов – конечный результат проектирования и отправная точка процесса разработки
- Диаграммы классов полезны также при анализе предметной области
- Пример:

Пример диаграммы классов



Еще пример

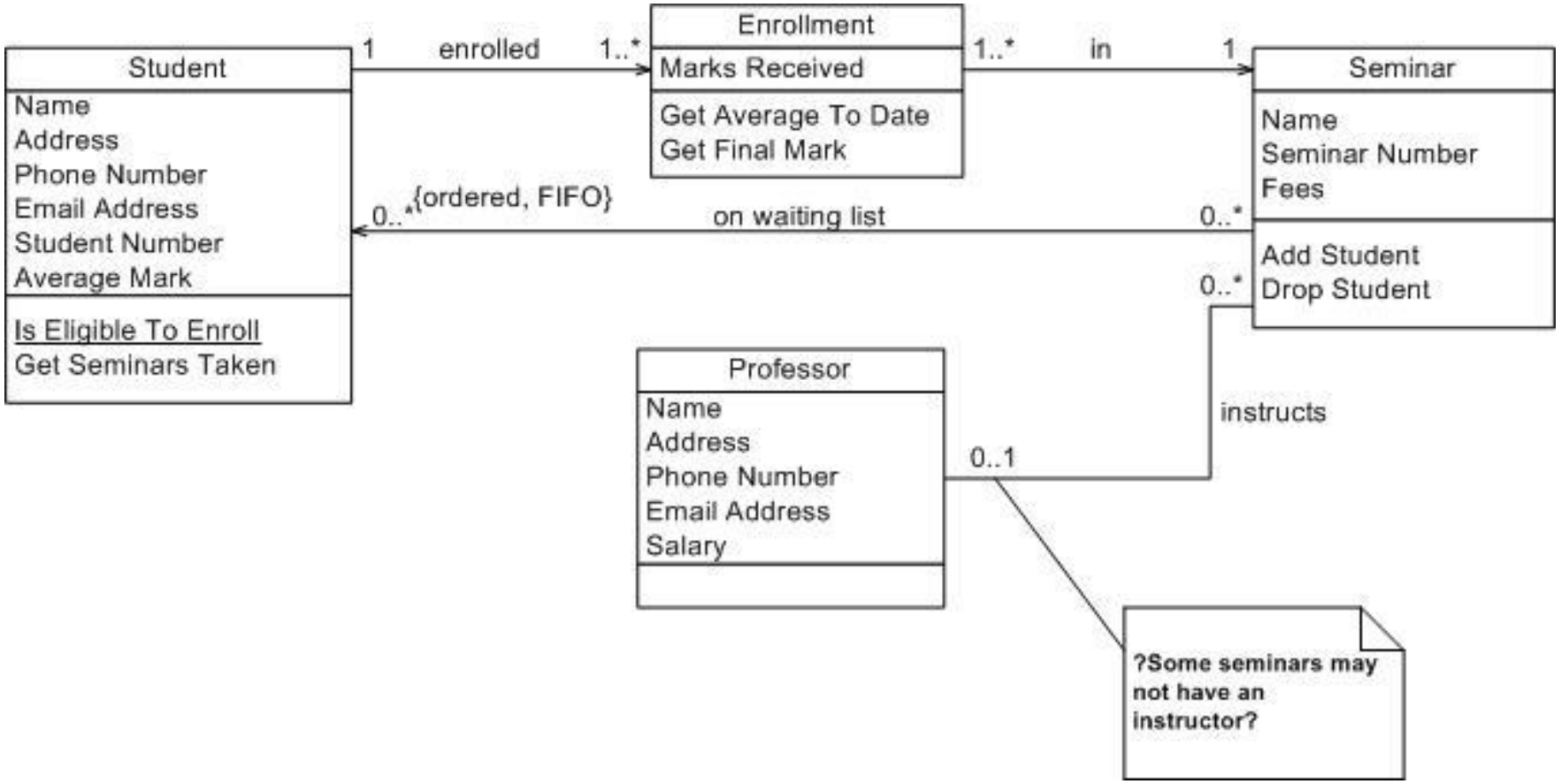
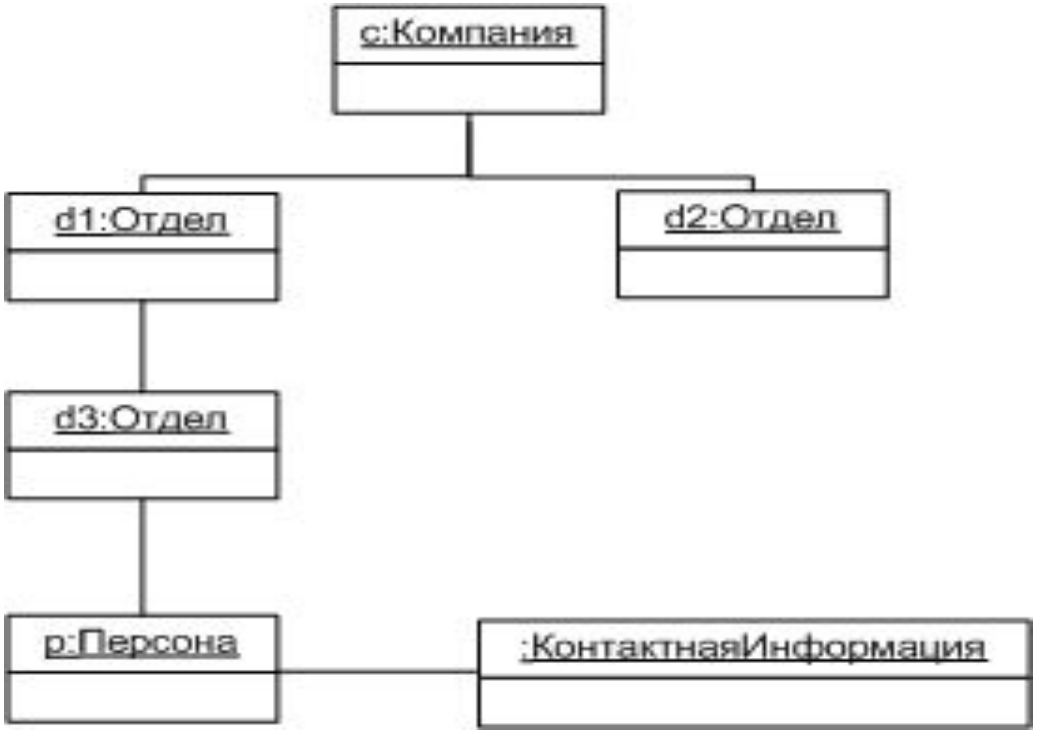
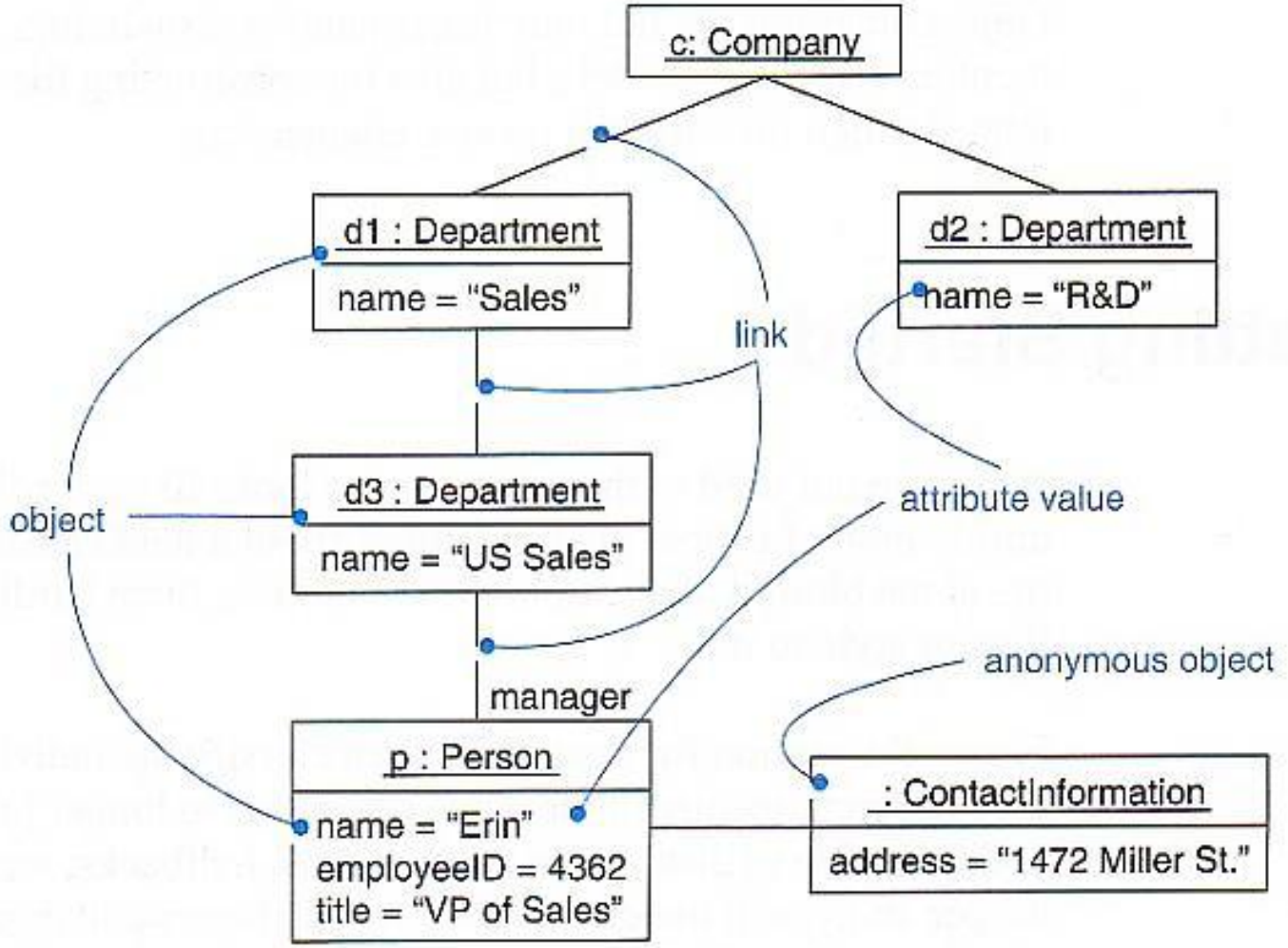

Student
Name
Address
Phone Number
Email Address
Student Number
Average Mark
Is Eligible To Enroll
Get Seminars Taken
1
enrolled
1..*
Enrollment
Marks Received
Get Average To Date
Get Final Mark
1..*
in
1
Seminar
Name
Seminar Number
Fees
Add Student
Drop Student
0..*
{ordered, FIFO}
on waiting list
0..*
0..*
instructs
Professor
Name
Address
Phone Number
Email Address
Salary
0..1
?Some seminars may not have an instructor?

Диаграмма объектов (Object diagram)

- Объект – экземпляр класса
- На диаграмме объектов показаны объекты и их отношения в некоторый момент времени
- Пример:



Пример диаграммы объектов



Еще пример

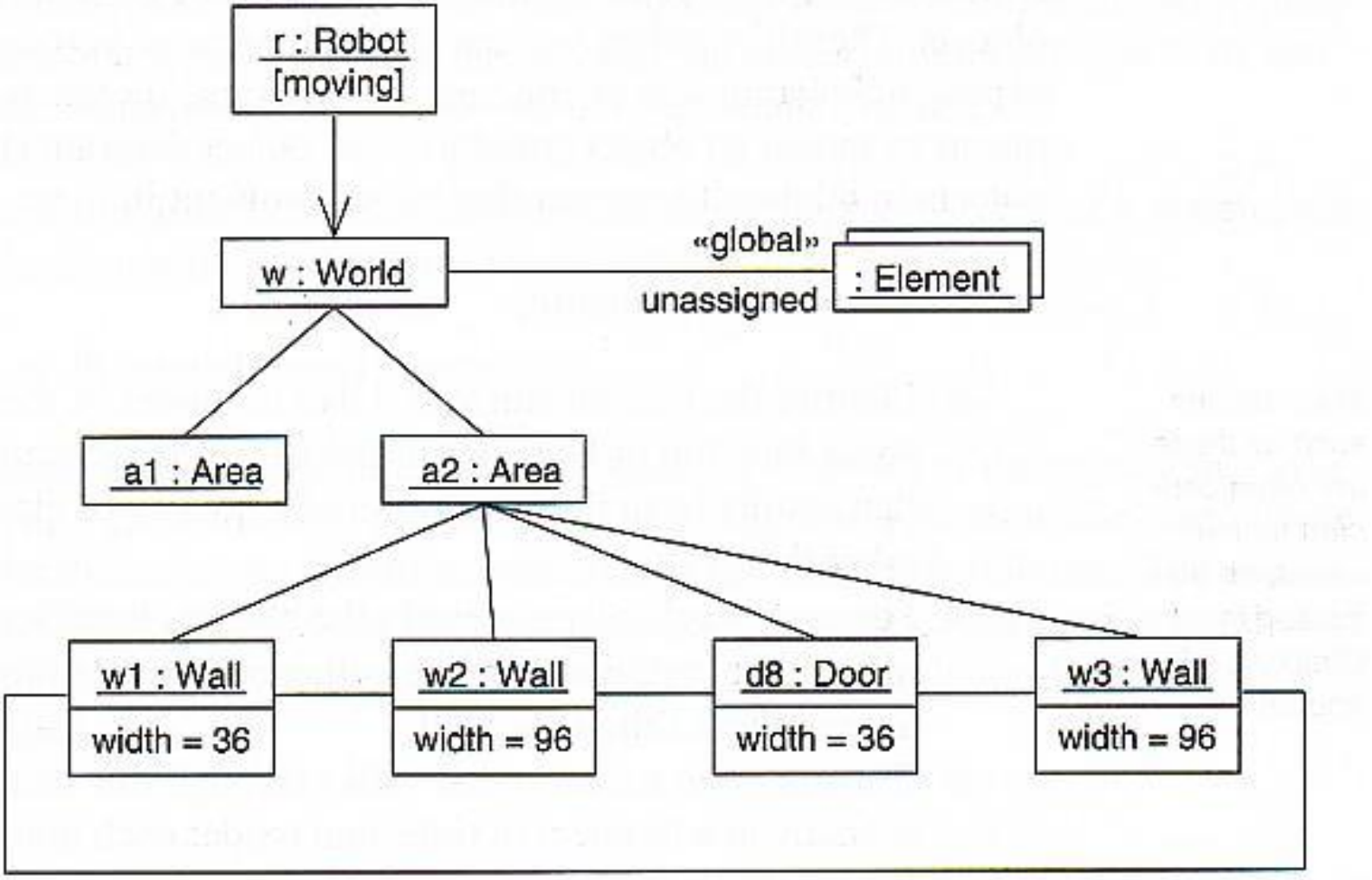
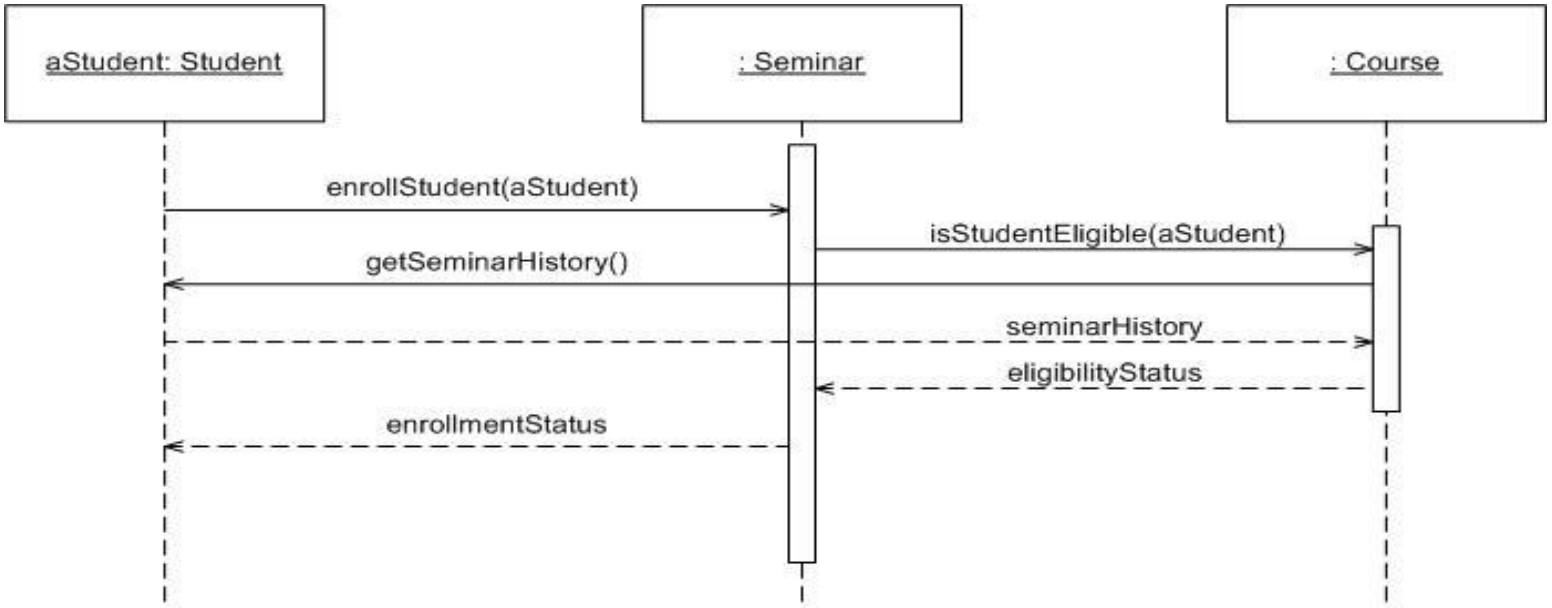
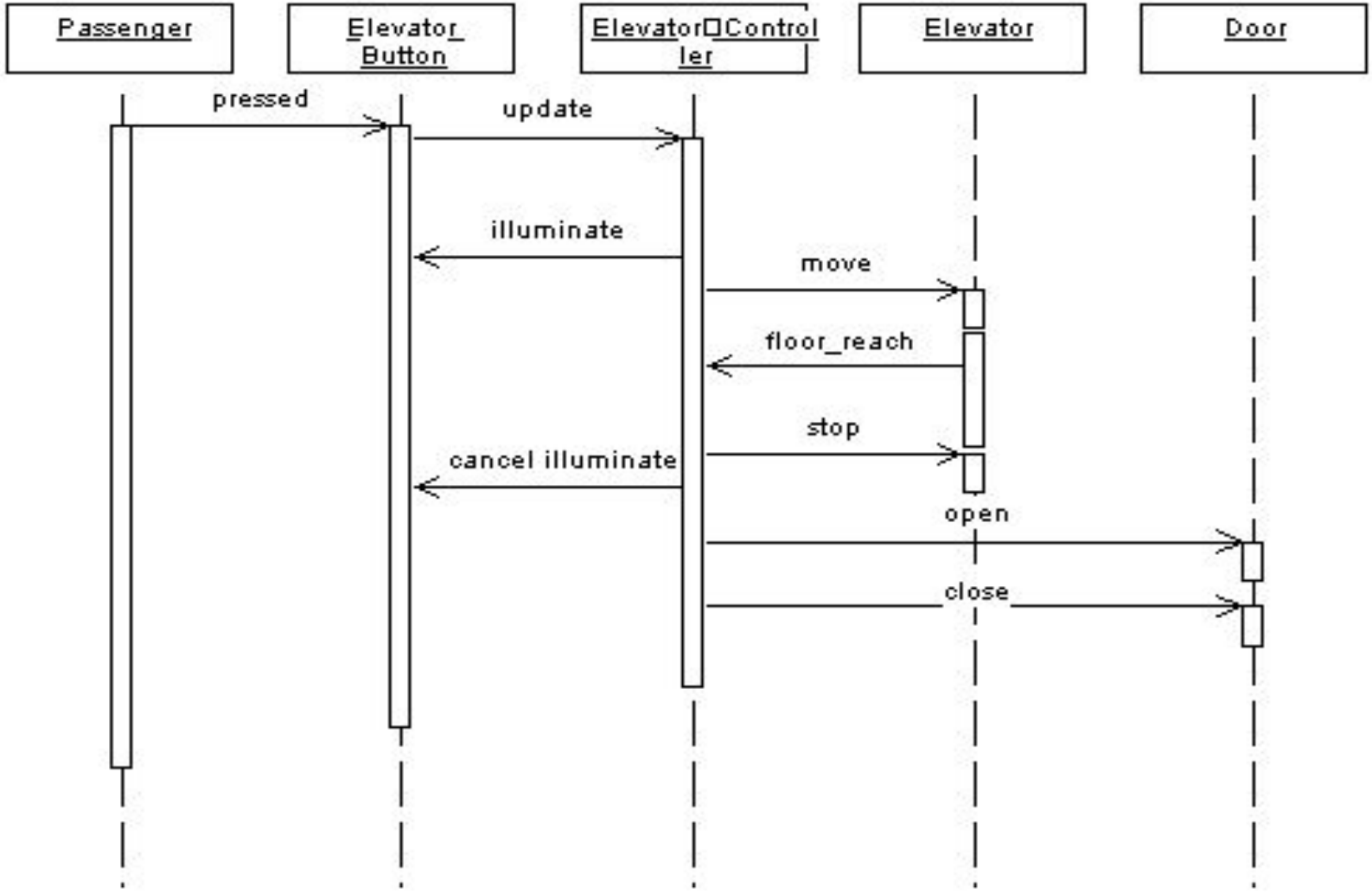

r : Robot
[moving]
w : World
«global»
unassigned
: Element
a1 : Area
a2 : Area
w1 : Wall
width = 36
w2 : Wall
width = 96
d8 : Door
width = 36
w3 : Wall
width = 96

Диаграмма последовательностей (sequence diagram)

- Диаграмма последовательностей отображает взаимодействие объектов в динамике
- Диаграммы последовательностей часто используются для точного определения логики сценария
- Пример:



Пример диаграммы последовательностей



Еще пример

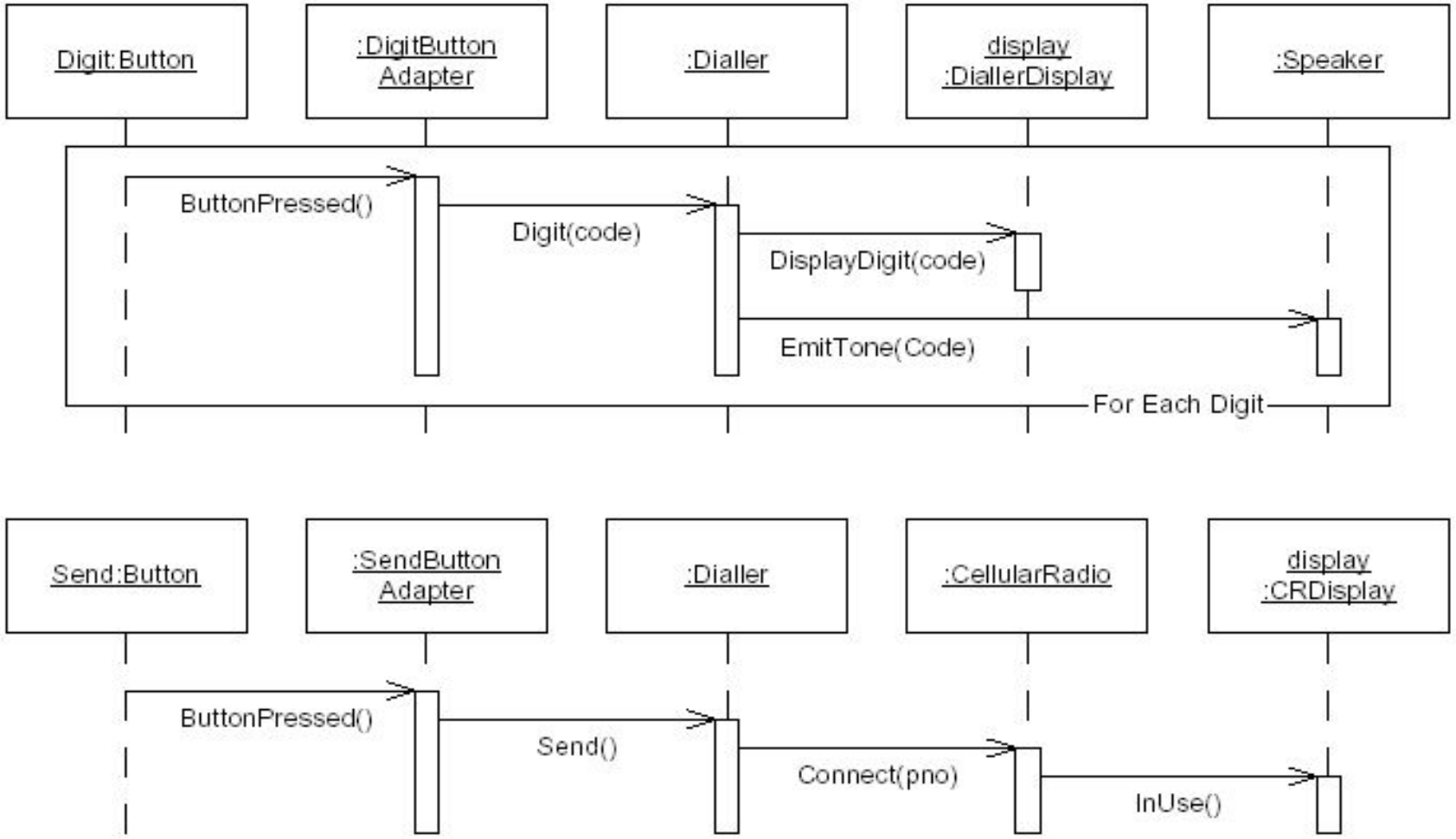
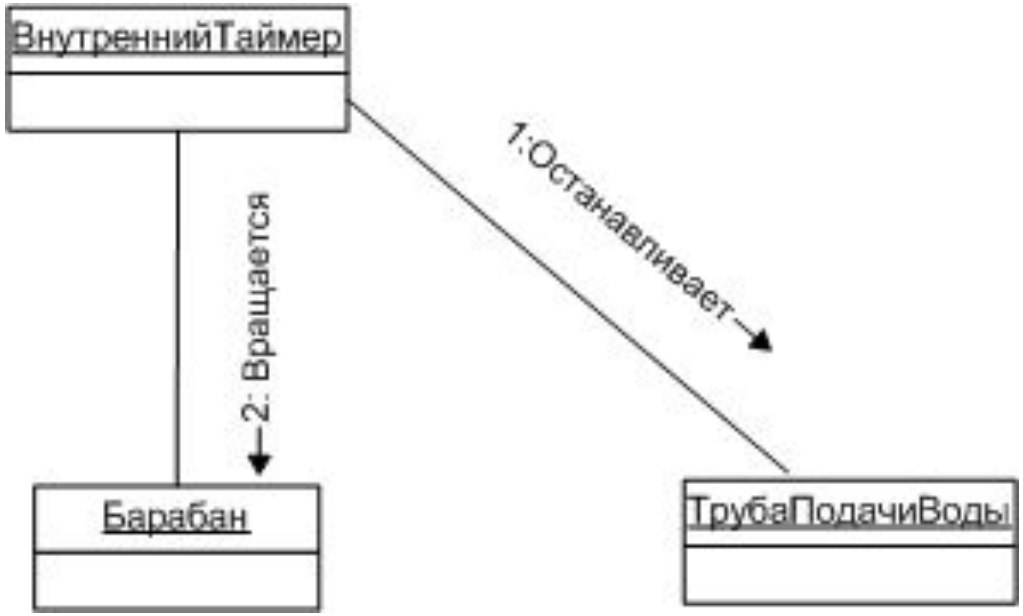
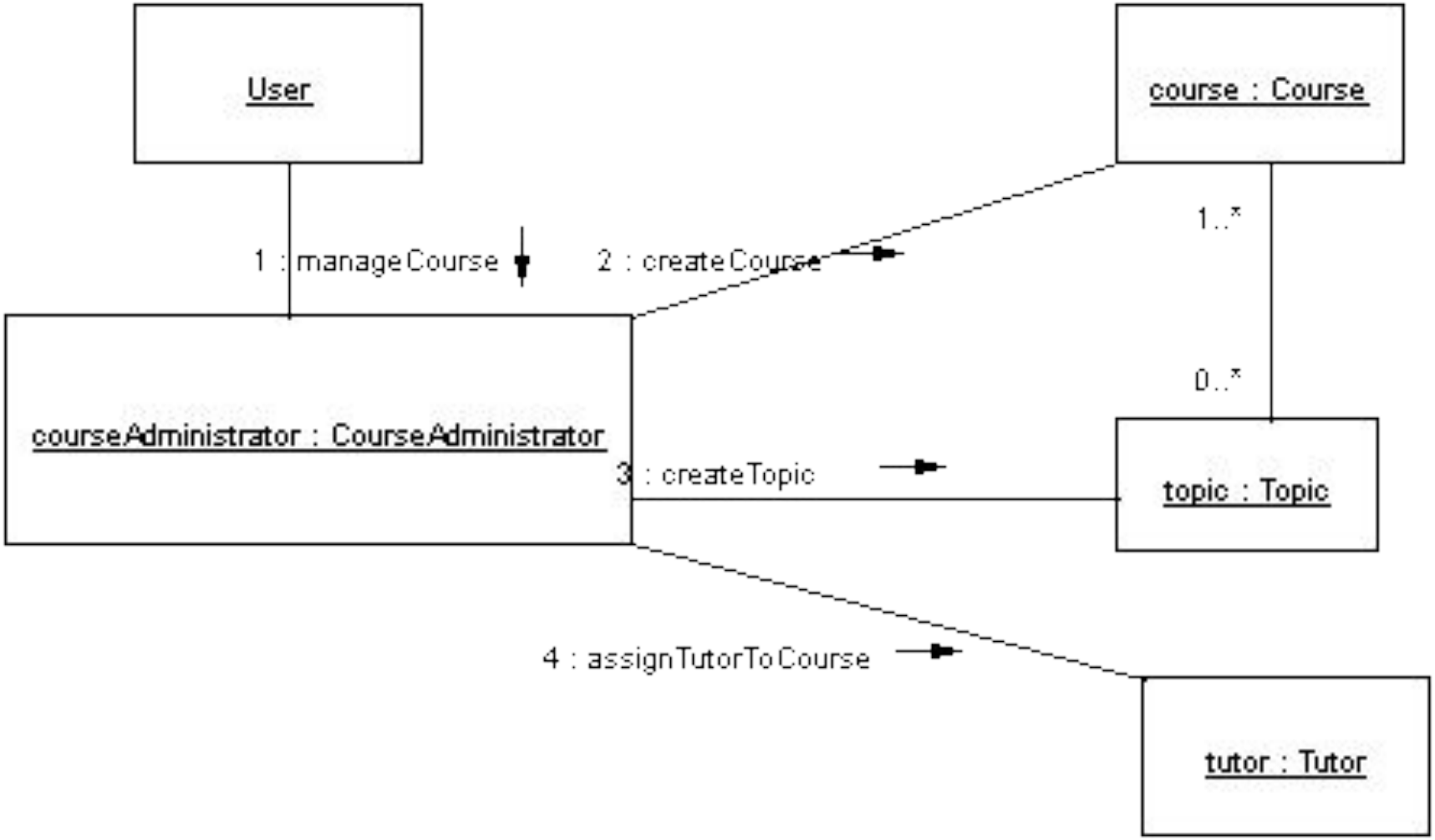

Digit:Button
:DigitButton Adapter
:Dialler
display :DiallerDisplay
:Speaker
ButtonPressed()
Digit(code)
DisplayDigit(code)
EmitTone(Code)
For Each Digit
Send:Button
:SendButton Adapter
:Dialler
:CellularRadio
display :CRDisplay
ButtonPressed()
Send()
Connect(pno)
InUse()

Диаграмма взаимодействия (collaboration diagram)

- Диаграммы взаимодействия показывают как элементы системы работают совместно для достижения общих целей
- Диаграммы взаимодействия аналогичны диаграммам последовательности
- Пример:



Пример диаграммы взаимодействия



Еще пример

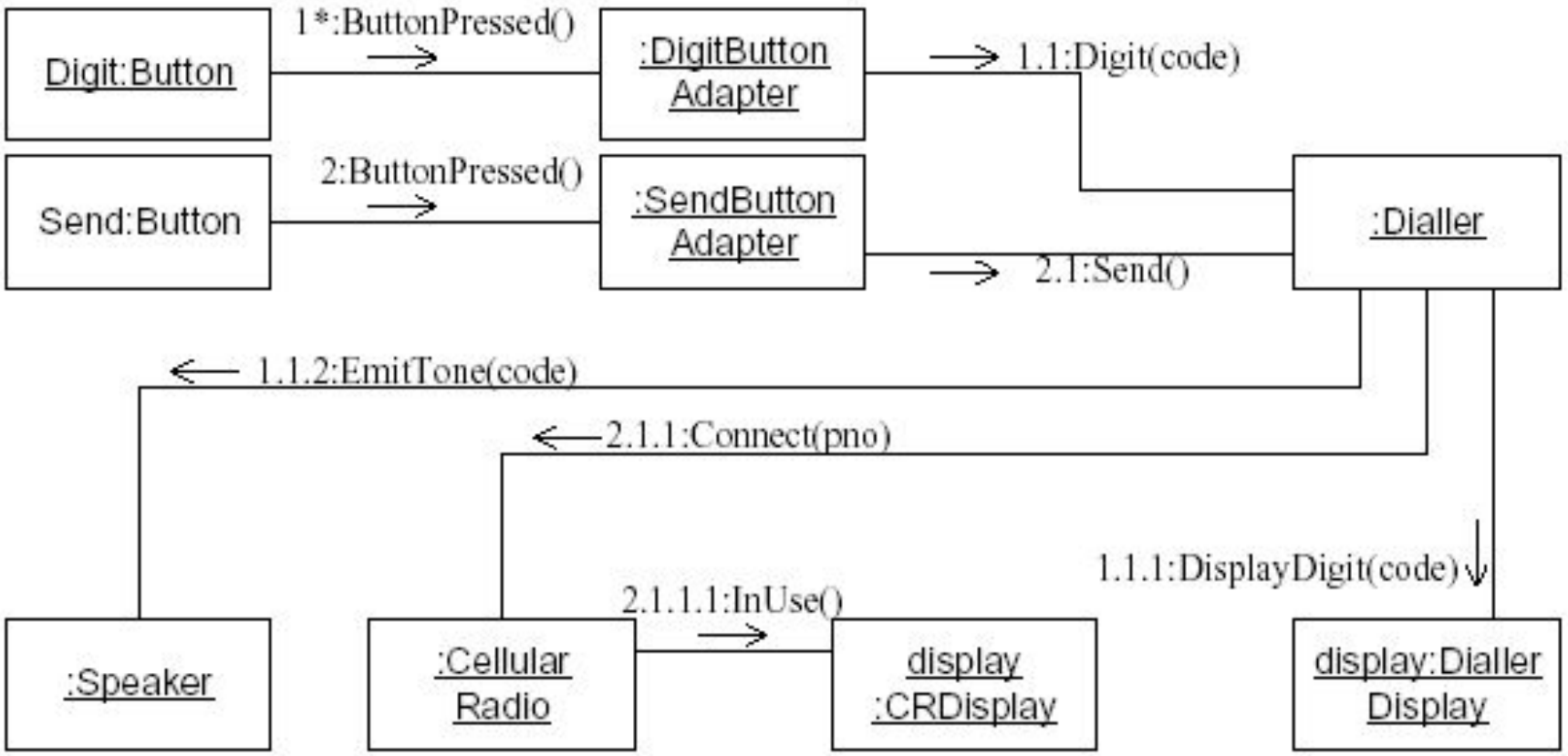
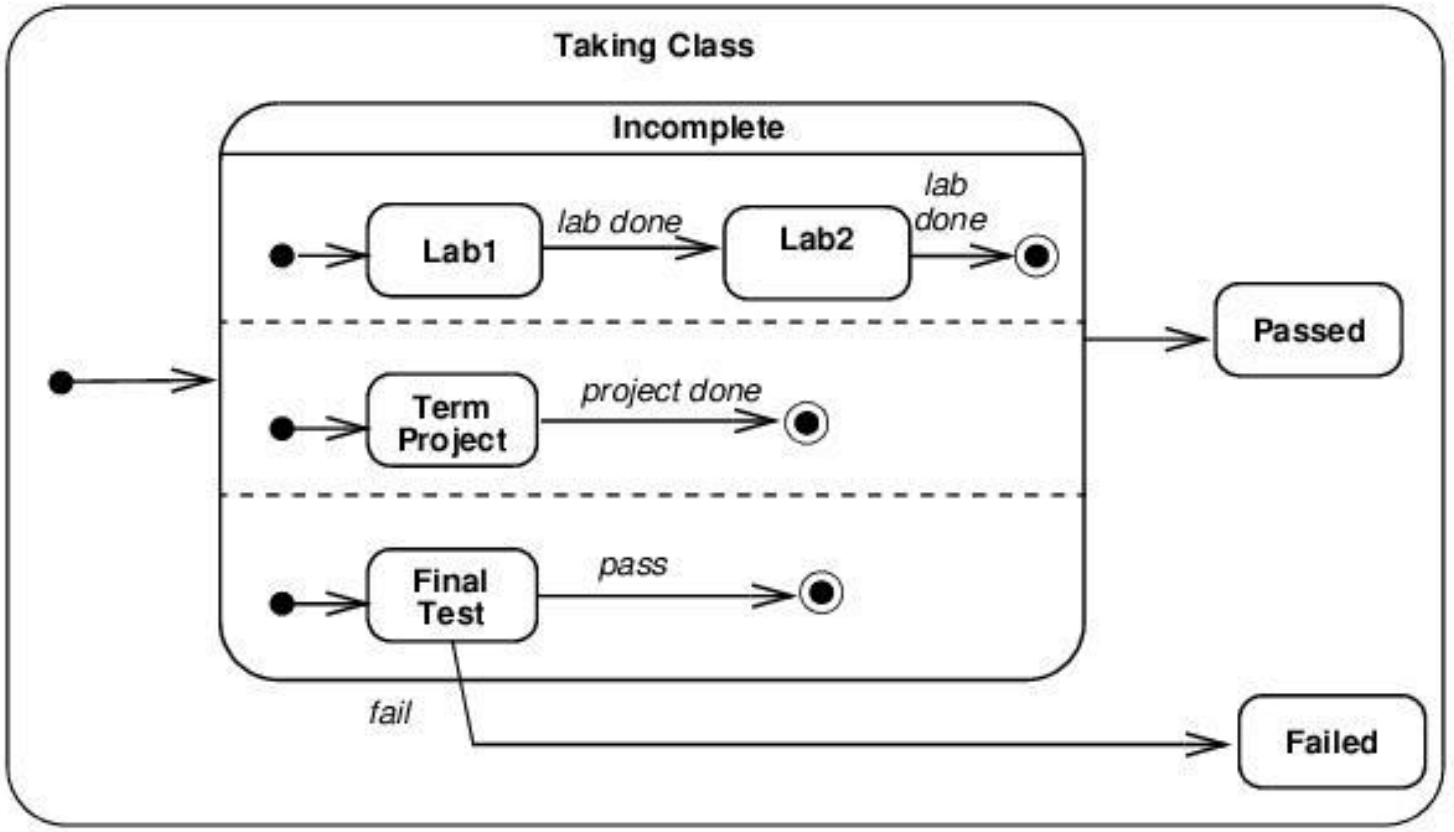

Digit:Button
1*:ButtonPressed()
:DigitButton Adapter
1.1:Digit(code)
Send:Button
2:ButtonPressed()
:SendButton Adapter
2.1:Send()
:Dialler
1.1.2:EmitTone(code)
2.1.1:Connect(pno)
1.1.1:DisplayDigit(code)
2.1.1.1:InUse()
:Speaker
:Cellular Radio
display :CRDisplay
display:Dialler Display

Диаграмма состояний (statechart diagram)

- Объекты имеют как поведение, так и состояние
- В каждый момент времени объект находится в некотором определенном состоянии
- Диаграмма состояний показывает, как объект переходит из одного состояния в другое
- При

Пример диаграммы состояний



Более сложная диаграмма

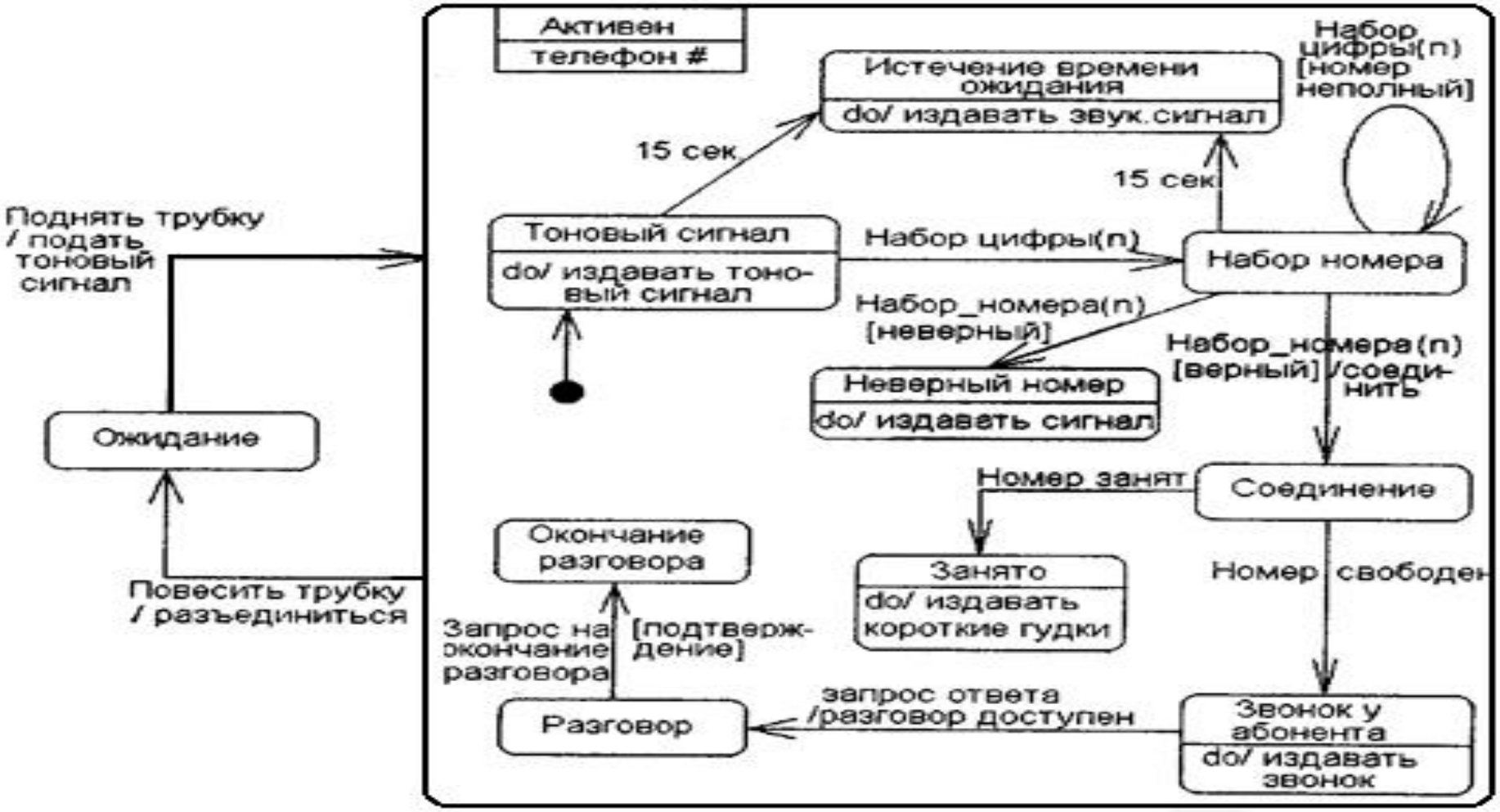
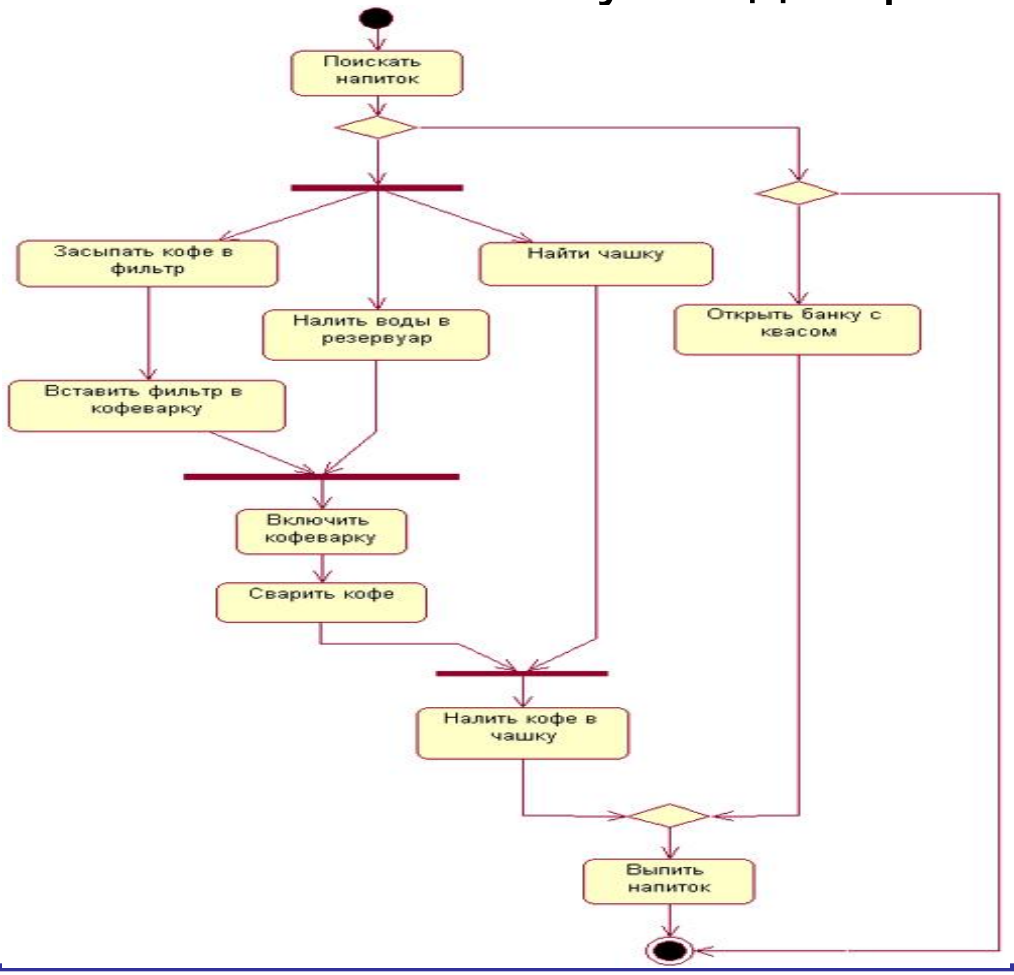
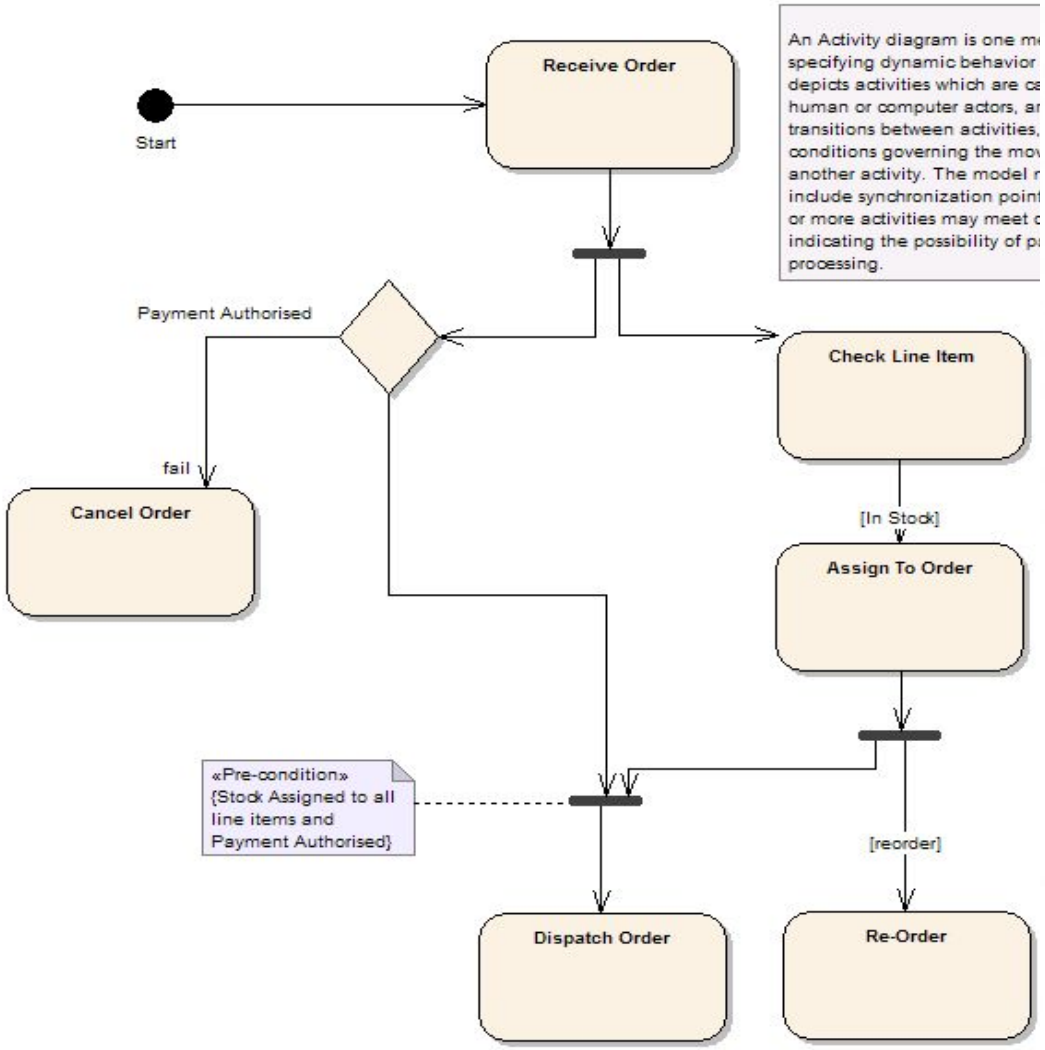


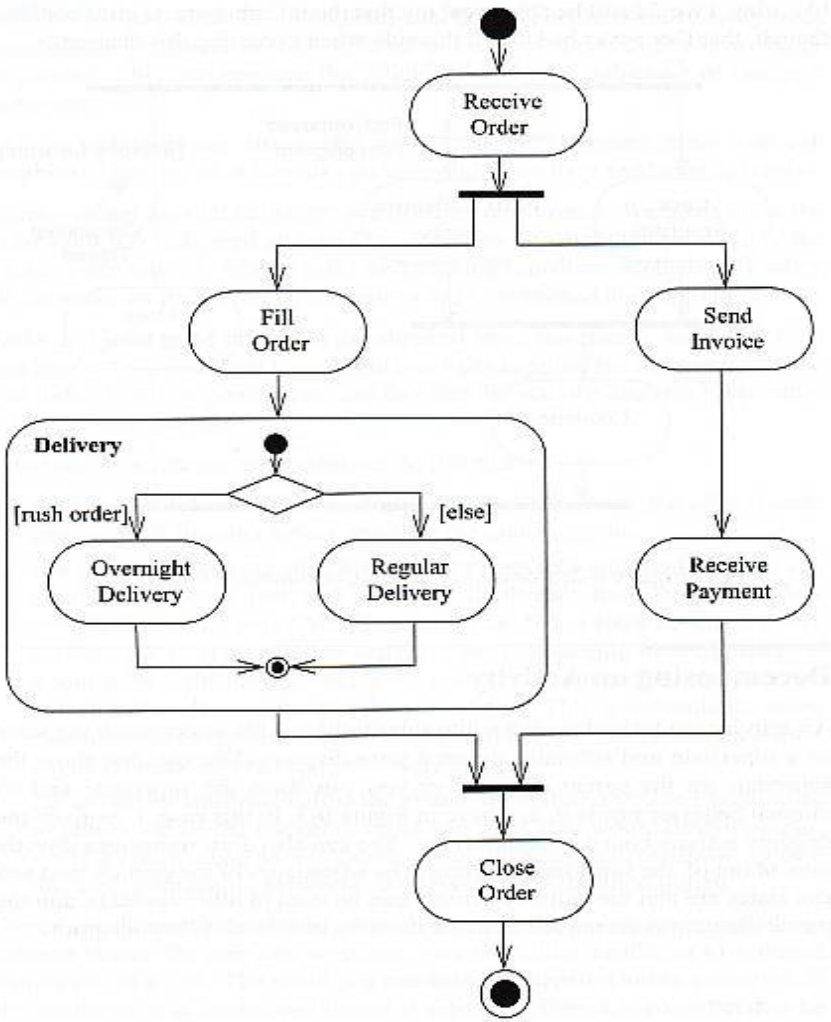
Диаграмма активности (activity diagram)

- Диаграммы деятельности – частный случай диаграмм состояний
- Такие диаграммы детализируют особенности алгоритмической и логической реализации операций, выполняемых системой.
- Пример ⇒



Примеры диаграмм активности





Еще пример

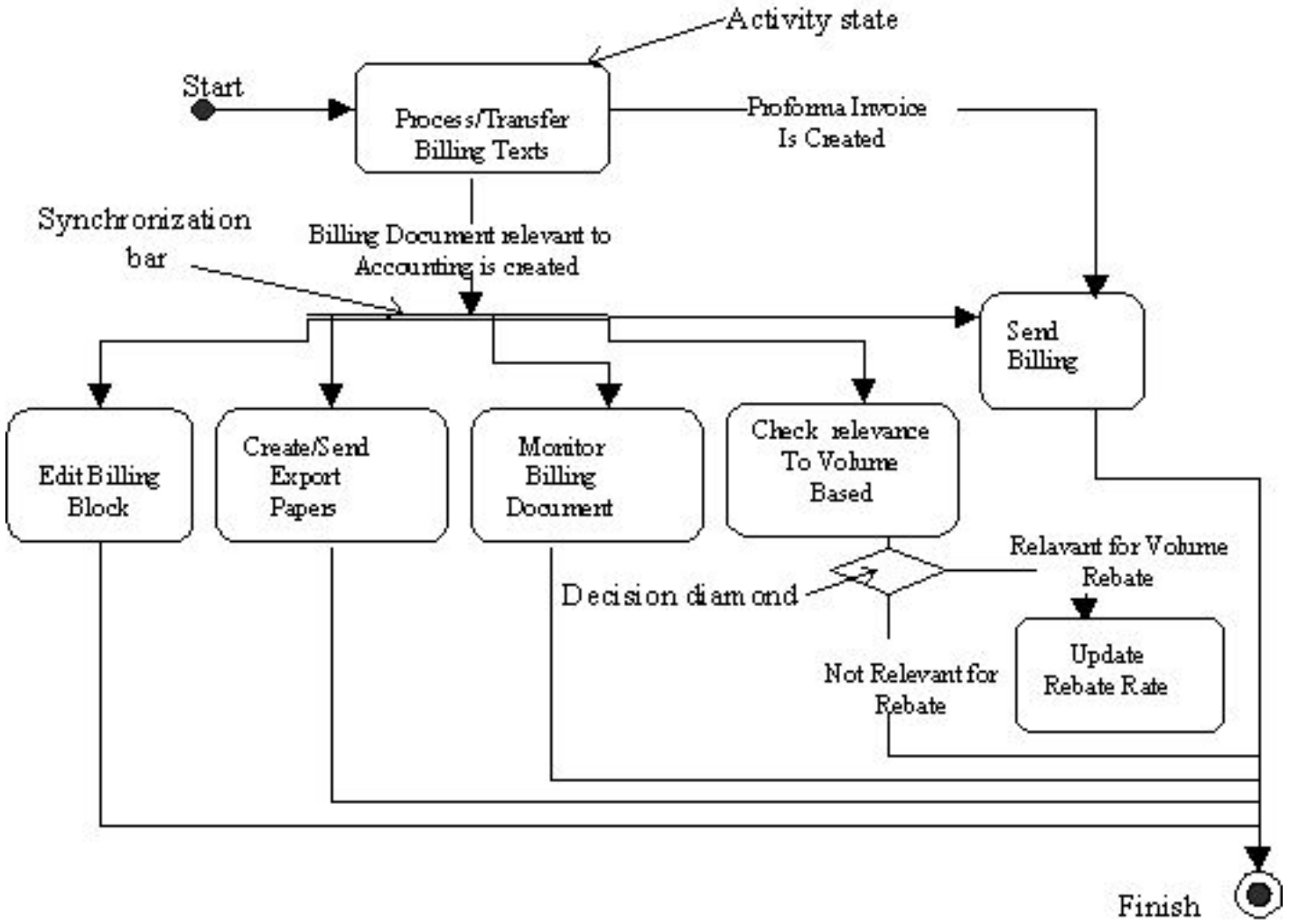
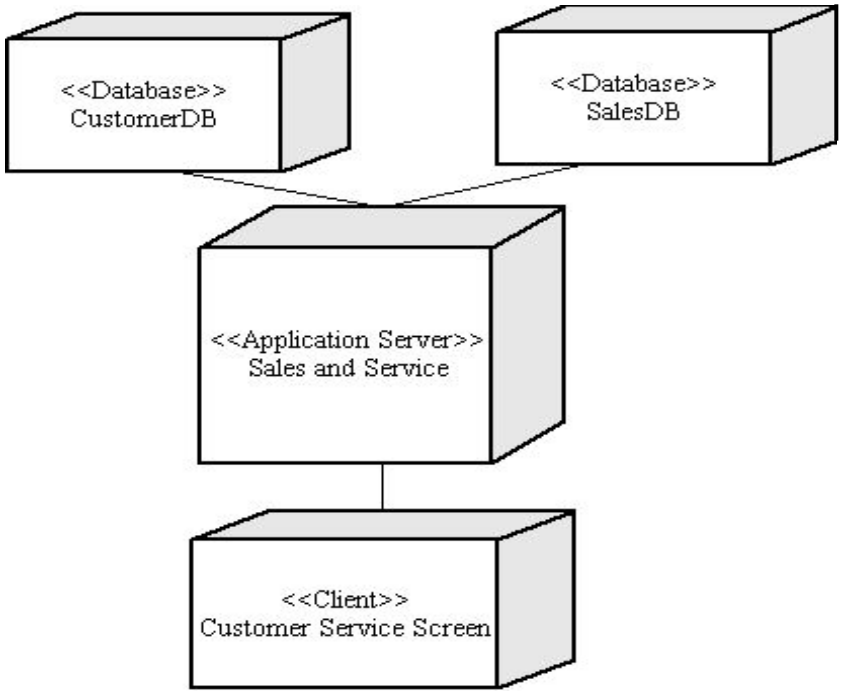
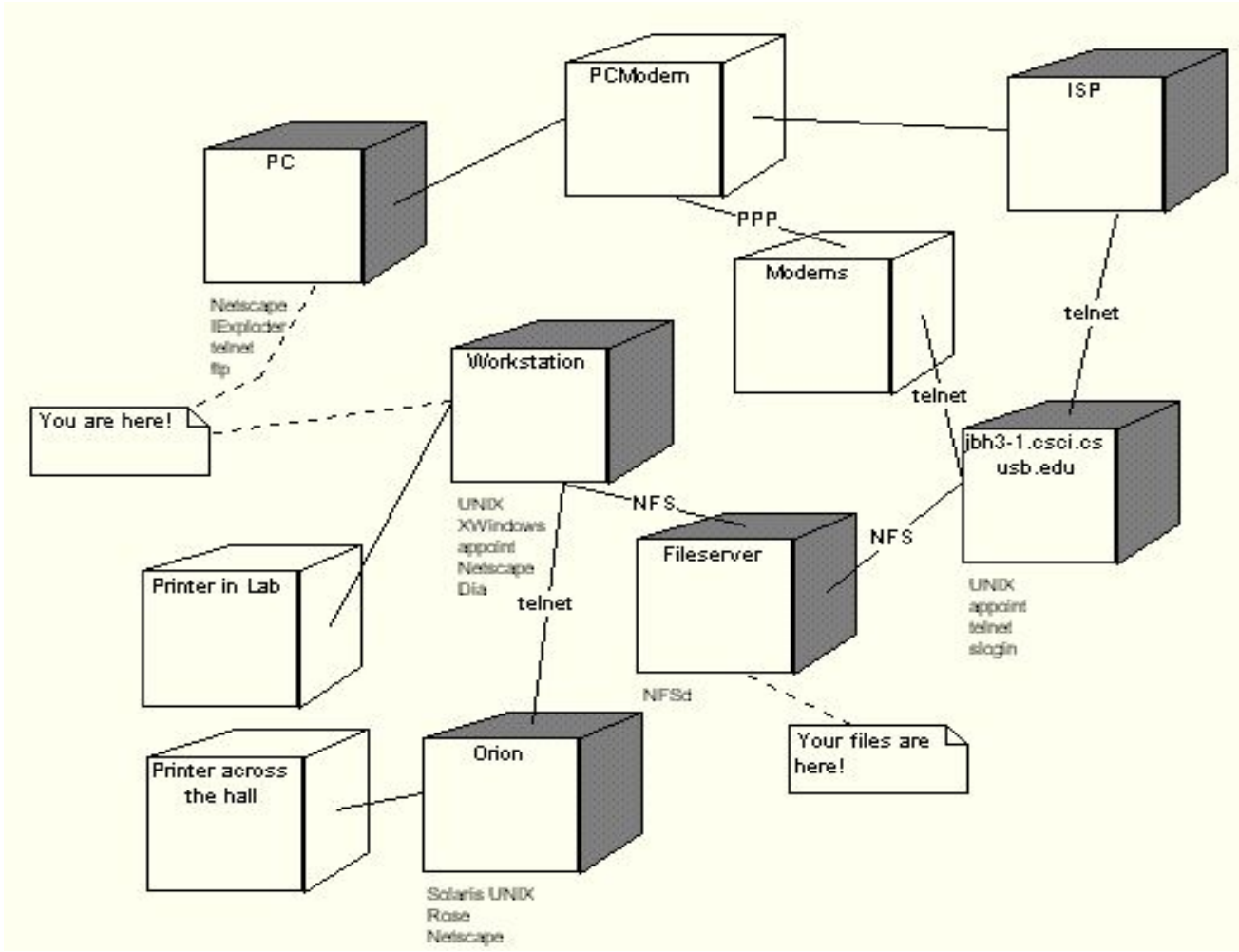

Activity state
Start
Process/Transfer Billing Texts
Proforma Invoice Is Created
Synchronization bar
Billing Document relevant to Accounting is created
Send Billing
Edit Billing Block
Create/Send Export Papers
Monitor Billing Document
Check relevance To Volume Based
Relavant for Volume Rebate
Decision diamond
Not Relevant for Rebate
Update Rebate Rate
Finish

Диаграмма развертывания (deployment diagram)

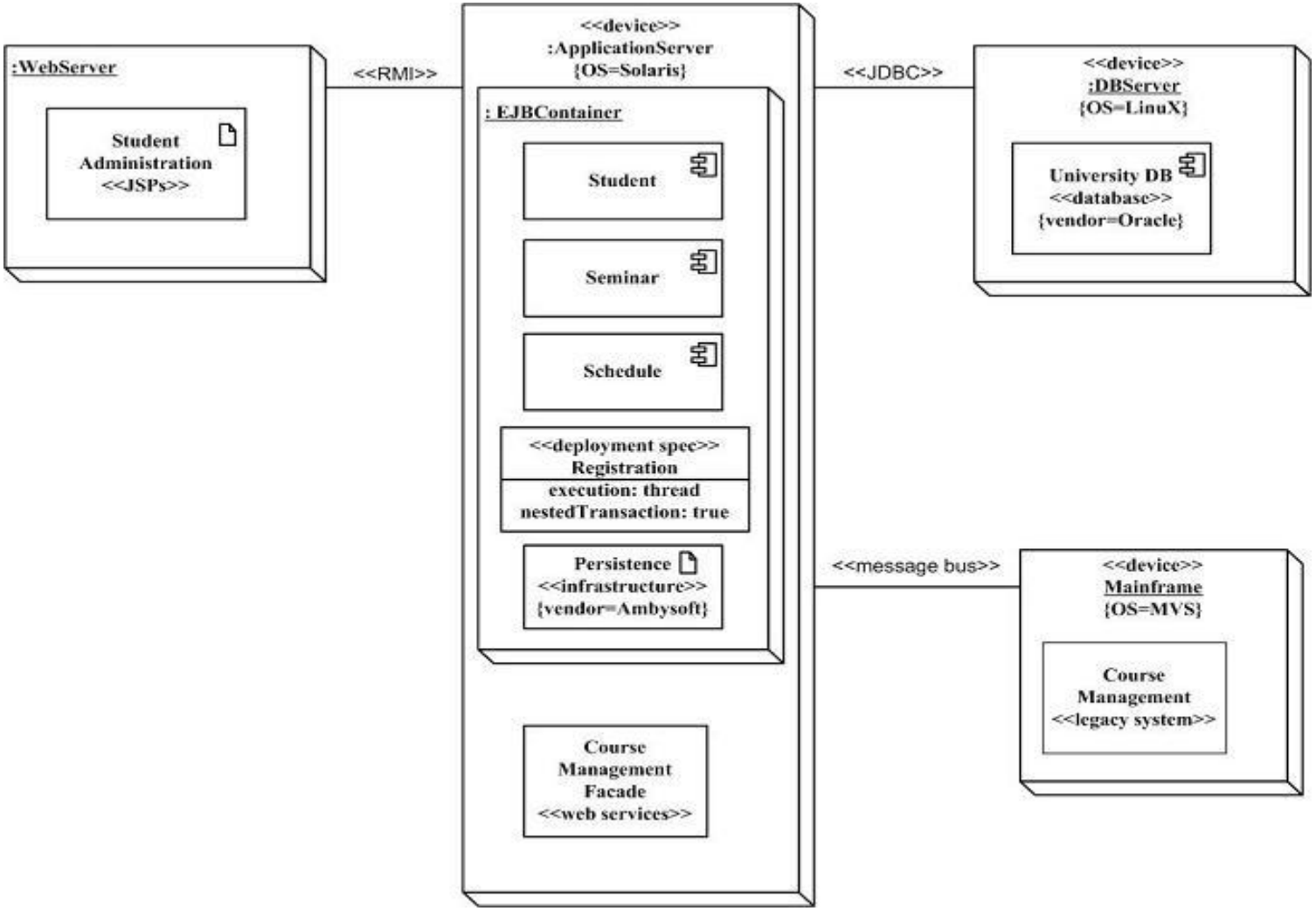
- Диаграммы развертывания предназначены лишь для распределенных компьютерных систем
- Диаграмма развертывания показывает физическую архитектуру компьютерной системы
- Пример:



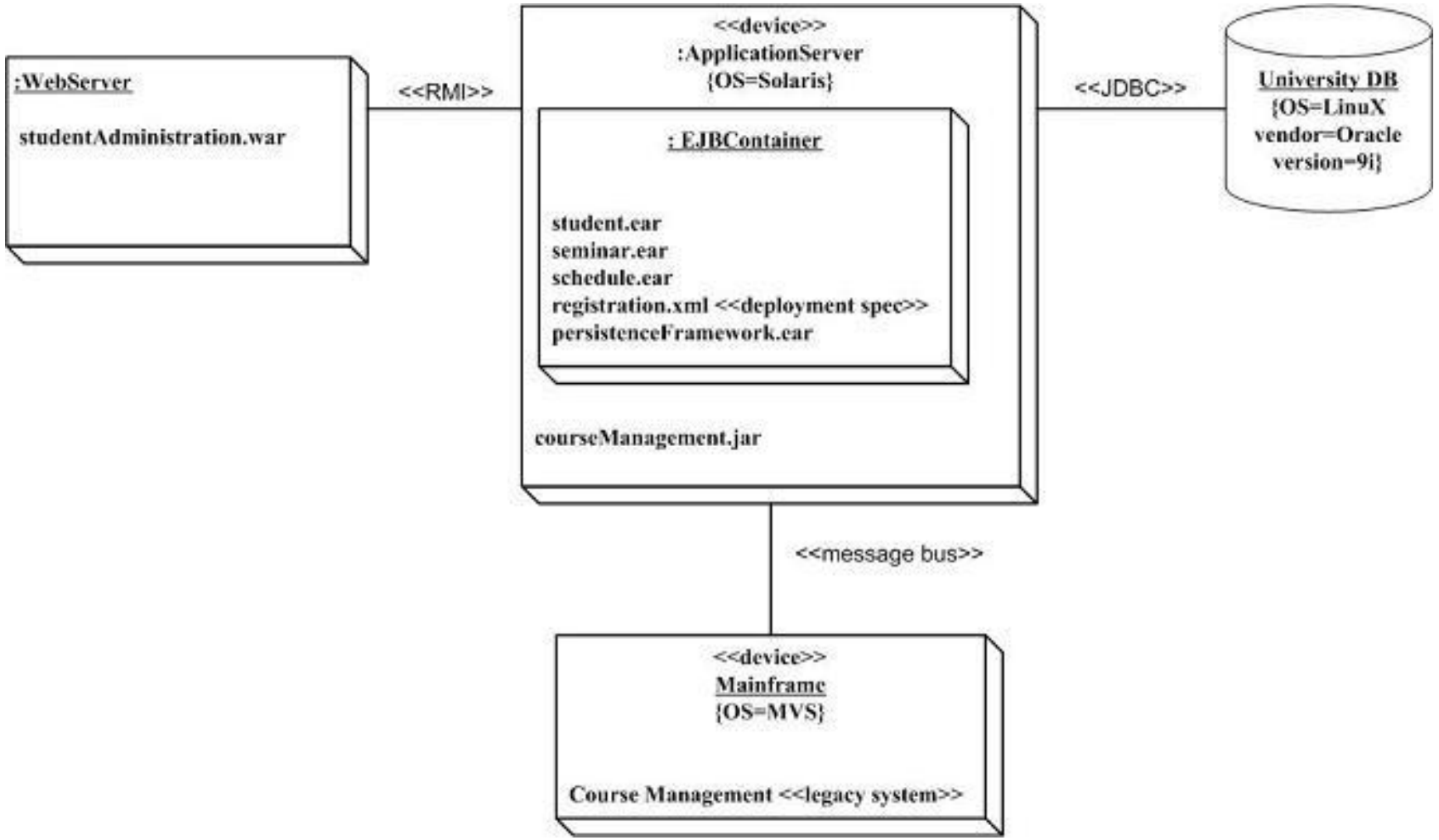
Пример диаграммы развертывания



Еще пример


:WebServer
Student Administration
<<JSPs>>
<<RMI>>
<<device>>
:ApplicationServer
{OS=Solaris}
: EJBContainer
Student
Seminar
Schedule
<<deployment spec>>
Registration
execution: thread
nestedTransaction: true
Persistence
<<infrastructure>>
{vendor=Ambysoft}
Course Management Facade
<<web services>>
<<JDBC>>
<<device>>
:DBServer
{OS=LinuX}
University DB
<<database>>
{vendor=Oracle}
<<message bus>>
<<device>>
Mainframe
{OS=MVS}
Course Management
<<legacy system>>

Тот же пример в краткой форме



ООП и последовательность построения диаграмм

Выбор необходимых диаграмм (Г. Буч)

- Какие именно виды диаграмм лучше всего отражают архитектуру системы и возможный технический риск, связанный с проектом?
- Какие из диаграмм удобнее всего превратить в инструмент контроля за разработкой системы?
- На всякий случай сохраняйте даже забракованные диаграммы!

Последовательность построения диаграмм

Если у вас *нет опыта* объектно-ориентированной разработки, воспользуйтесь следующими рекомендациями:

- Начинайте применять идею абстрагирования в отношении конкретных моделей. Очень полезны коллективные упражнения с анализом прецедентов
- Постройте модель простой статической части задачи с помощью классов, зависимостей, обобщений и ассоциаций
- Примените простые диаграммы последовательностей или кооперации для моделирования динамической части задачи. Хорошо начать с построения модели взаимодействия пользователя с системой.

Последовательность построения диаграмм (продолжение)

- В соответствии с принципами ООП, диаграммы можно строить в такой последовательности:
  - диаграмма прецедентов
  - диаграмма классов
  - диаграмма объектов
  - диаграмма последовательностей
  - диаграмма взаимодействия
  - диаграмма состояний
  - диаграмма активности
  - диаграмма развертывания
- Существуют разные подходы к последовательности построения диаграмм.
- Не всегда нужно строить все диаграммы!

Вопросы ?

Вопросы и упражнения

- Вопросы:
  - Почему нужно строить разные диаграммы при моделировании системы?
  - Какие диаграммы соответствуют статическому представлению о системе?
  - Какие диаграммы представляют систему в динамике?
- Упражнения:
  - Вы разрабатываете компьютерную программу для игры в шахматы. Какая диаграмма UML была бы полезной в этом случае? Почему?
  - Составьте список вопросов потенциальному пользователю такой программы? Объясните, почему вы хотели бы задать именно их?

Использованные материалы

При разработке представленных материалов с разрешения авторов или правообладателей использовались следующие источники:

- Г.Буч, А. Джекобсон, Дж. Рамбо.
  UML: Руководство пользователя
  http://alice.stup.ac.ru/~dvn/uproc/books/uml_user_guide/index.htm
- Унифицированный язык моделирования ПО. Scott W. Ambler Copyright © 1998 Software Development magazine
  http://zone1c.narod.ru/
- Леоненков. Самоучитель по UML
  http://khpi-iip.mipk.kharkiv.edu/library/case/leon/index.html
- Дж. Шмулер. Освой самостоятельно UML за 24 часа, 2-е издание: пер. с англ. – М.: Изд. Дом «Вильямс», 2002.
- Sample: The UML © Dr. Richard J. Botting, California State University
  http://www.csci.csusb.edu/dick/samples/uml.html
- OMG Unified Modeling Language Specification. Copyright © 2000, Object Management Group http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf

О проекте ВИРТУОЗ

Цель проекта "Виртуоз" - подготовка преподавателей к внедрению в ВУЗах России и других стран СНГ новой специальности – "Информационные технологии". Основная задача проекта - совершенствование методологии преподавания программной инженерии. Реализация этой программы направлена на создание надёжной образовательной основы для дальнейшего развития отечественной ИТ-индустрии.

В рамках проекта (август-декабрь 2004 г.) - обучение современным методикам программной инженерии, передовым информационным технологиям, получение опыта работы по созданию учебных курсов, соответствующих международным образовательным стандартам IEEE/ACM Computing Curricula 2001: Computer Science и Software Engineering, а также прохождение стажировки в корпорации Intel.

Проект "Виртуоз" выполняется в Нижегородском государственном университете им. Н.И. Лобачевского по инициативе компании Intel при поддержке Microsoft, IBM, Borland, Лаборатория Касперского и др.