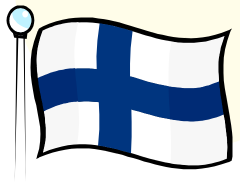
UML in practice

A BRIEF INTRODUCTION
TO THE STANDARD OBJECT MODELING LANGUAGE

ESA SALMIKANGAS JAMK/ICT 29.11.2013

Esa Salmikangas



- Senior Lecturer in Software Engineering
- MSc, DI, MCP
- Jyväskylä University of Applied Sciences, Finland
- Over ten years experience in SW industry before University

The main sources

And preferable reading…

- Martin Fowler : UML Distilled 3rd Edition
- Booch, Rumbaugh, Jacobson: The Unified Modeling Language User Guide 2nd Edition
- www.agilemodeling.com
- http://www.uml.org/

CONTENT

Chapter 1: Introduction

WHAT AND WHY UML?

UML IS A LANGUAGE

- Unified
- Modeling
- Language

LANGUAGES ARE FOR COMMUNICATION.

UML is a Language for Visualizing.

UML is a Language for Specifying.

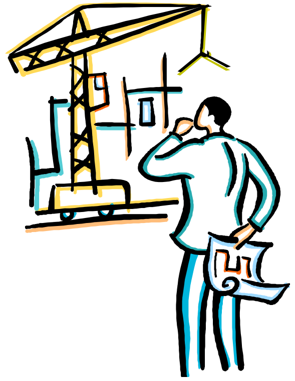
UML is a Language for Constructing.

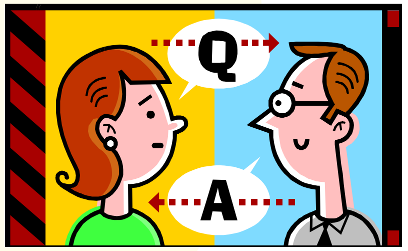
UML is a Language for Documenting.

SPECS

- specification = design documents
- Specs are for:

1. Design a system
2. communication





MODELING

What is a Model?

- A model is a simplification of reality.

Why do we model?

- We build models so that we can better understand the system we are developing.

Principles of Modeling

First

- The choice of what models to create has a deep influence on how a problem is attacked and how to solution is shaped.

Second

- Every model may be expressed at different levels of precision.

Third

- The best models are connected to reality.

Fourth

- No single model or view is sufficient.

What Is the UML?

- The Unified Modeling Language (UML) is a family of graphical notations, backed by single meta-model, that help in describing and designing object-oriented OO style software systems.

http://www.uml.org/

UML is de-facto standard

- The Unified Modeling Language (UML) has quickly become **the de-facto standard** for building Object-Oriented software.

UML is a standard

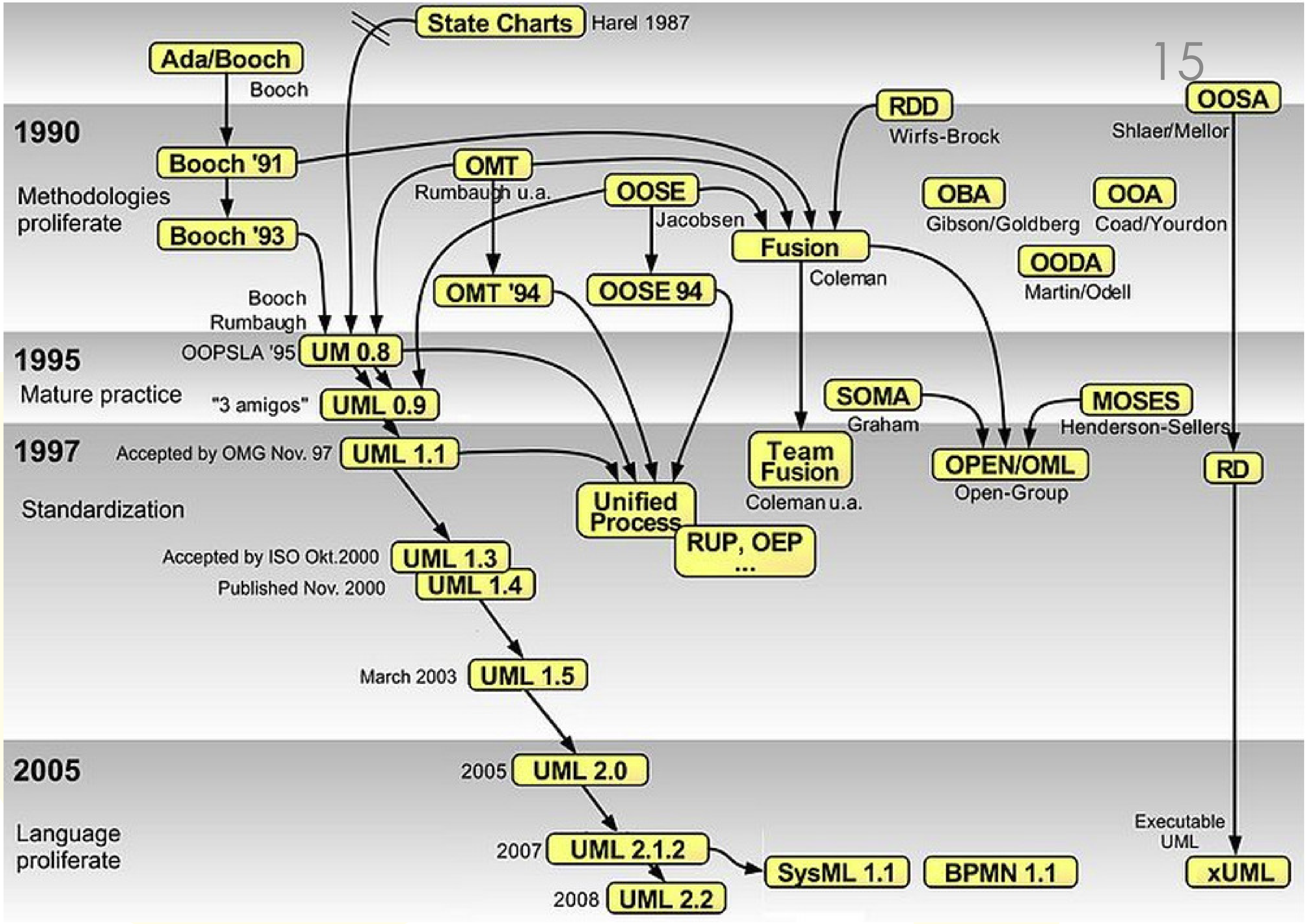
OBJECT MANAGEMENT GROUP
http://www.omg.org/

- Unified Modeling Language (UML) is a standardized (ISO/IEC 19501:2005), general-purpose modeling language in the field of software engineering.
  - The UML is a relatively open standard, controlled by the Object Management Group (OMG),
    an open consortium of companies.
    - The OMG was formed to build standards the support interoperability, specially interoperability of object-oriented systems

UML versions

http://www.omg.org/

- **Appearance in 1997 version 1.0**
- 1998 UML 1.2
- 1999 UML 1.3
- 2001 UML 1.4
- 2002 UML
- **2005 UML 2.0**
- 2009 UML 2.2
- 2010 UML 2.3
- 2011 UML 2.4
- 2012 UML 2.5 “in process”



State Charts
Harel 1987
Ada/Booch
Booch
1990
Booch '91
OMT
Rumbaugh u.a.
OOSE
Jacobsen
RDD
Wirfs-Brock
OOSA
Shlaer/Mellor
Methodologies proliferate
Booch '93
Fusion
Coleman
OBA
Gibson/Goldberg
OOA
Coad/Yourdon
OODA
Martin/Odell
OMT '94
OOSE 94
Booch
Rumbaugh
1995
OOPSLA '95
UM 0.8
Mature practice
"3 amigos"
UML 0.9
SOMA
Graham
MOSES
Henderson-Sellers
1997
Accepted by OMG Nov. 97
UML 1.1
Team Fusion
Coleman u.a.
OPEN/OML
Open-Group
RD
Standardization
Unified Process
RUP, OEP ...
Accepted by ISO Okt.2000
UML 1.3
Published Nov. 2000
UML 1.4
March 2003
UML 1.5
2005
2005
UML 2.0
Language proliferate
2007
UML 2.1.2
SysML 1.1
BPMN 1.1
Executable UML
xUML
2008
UML 2.2

For what?

- Unified Modeling Language (UML) combines techniques from:
  - data modeling (entity relationship diagrams),
  - business modeling (work flows),
  - object modeling,
  - and component modeling.
- It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.

What UML is

- **not** a development method
- **not** a process
- **not** a programming environment
- **not** a programming language (yet)
- **not** ”a silver bullet”

USE OF UML

Ways of using the UML

- There has been (and there is still) long and difficult discussions how the UML should be used…
- three modes using UML:
  - sketch
  - blueprint
  - programming language

1) UML as sketch

- developers use the UML to help communicate some aspects of a system
- forward-engineering
  - UML diagrams before coding
- reverse-engineering
  - UML diagrams after coding
- pretty informal and dynamic, needs only lightweight drawing tools
- emphasis is on selective communication rather than complete specification

2) UML as blueprint

- UML as blueprint is about completeness
- blueprints are developed by a designer, gives a detailed design for a programmer
- more detailed, very specific from all details to a particular area
- require much more sophisticated tools -> CASE-tools
- sketches are more explorative, blueprints are more definetive

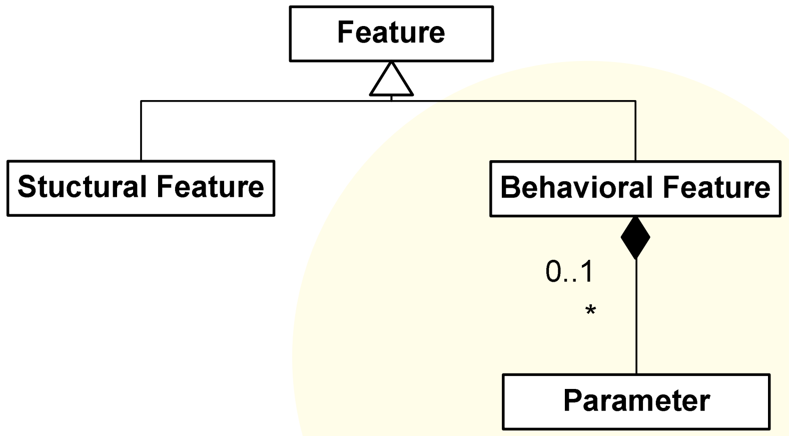
3) UML as programming language

- many CASE tools do some form of code generation
- can reach the point at which all the system can be specified in the UML -> UML as programming language
  - demands particularly sophisticated tooling
  - forward/reverse engineering don't make any sense for this mode, because UML and source code are same thing

Perspectives of the UML

- Software perspective
  - the elements of the UML map pretty directly to elements in a software system
- Conceptual perspective
  - the UML represents a description of the concepts of a domain study
  - we are more building a vocabulary to talk about a particular domain

Notations and meta-models



a small piece of the UML meta-model

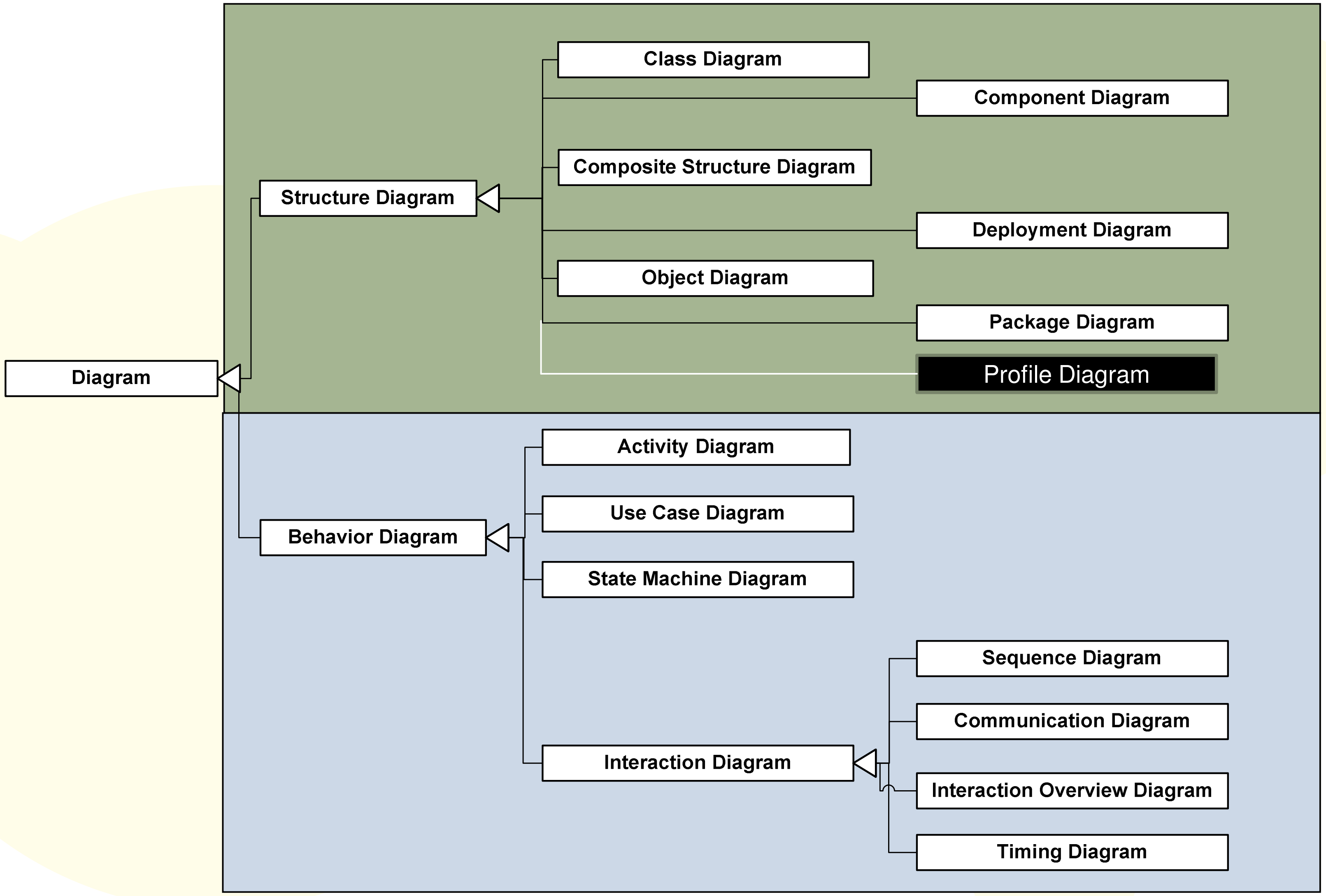
- The UML defines:
  - 1) a notation and
  - 2) a meta-model.
- The notation is the graphical syntax of the modeling language
- meta-model: a diagram defines the concepts of the language

UML Diagrams

The UML 2.2 describes 14 Diagrams

#	Diagram	In Finnish	Purpose	Lineage
1	**Activity**	Toiminto	Procedural and parallel behavior	UML 1
2	**Class**	Luokka	Class, features and relationships	UML 1
3	**Communication**	Yhteistyö, Vuorovaikutus?	Interaction between objects, emphasis on links	in UML 1 collaboration
4	**Component**	Komponentti	Structure and connections of components	UML 1
5	**Composite structure**	Kooste	Runtime decomposition of a class	UML 2
6	**Deployment**	Toteutus	Depolyment of artifacts to nodes	UML 1
7	**Interaction overview**	Vuorovaikutus	Mix of sequence and activity diagrams	UML 1
8	**Object**	Olio	Example configurations of instances	Unoff. UML 1
9	**Package**	Paketti	Compile-time hierarchic structure	Unoff. UML 1
10	**Sequence**	Viestiyhteys	Interaction between objects; emphasis on sequence	UML 1
11	**State machine**	Tila(kone)	How events change an object over its life	UML 1
12	**Timing**	Ajoitus?	Interaction between objects; emphasis on timing	UML 2
13	**Use case**	Käyttötapaus	How user interact wih a system	UML 1
14	**Profile**	Profiili	At Metamodel level show stereotypes and packages	UML 2.x

Classification of UML Diagram types



7 Structure Diagrams

- Class
- Component
- Composite structure
- Deployment
- Object
- Package
- Profile

Structure diagrams emphasize the things that must be present in the system being modeled. These diagrams represent the structure, they are used extensively in documenting the software architecture of software systems.

3 Behavior Diagrams

- Activity
- State Machine
- Use Case

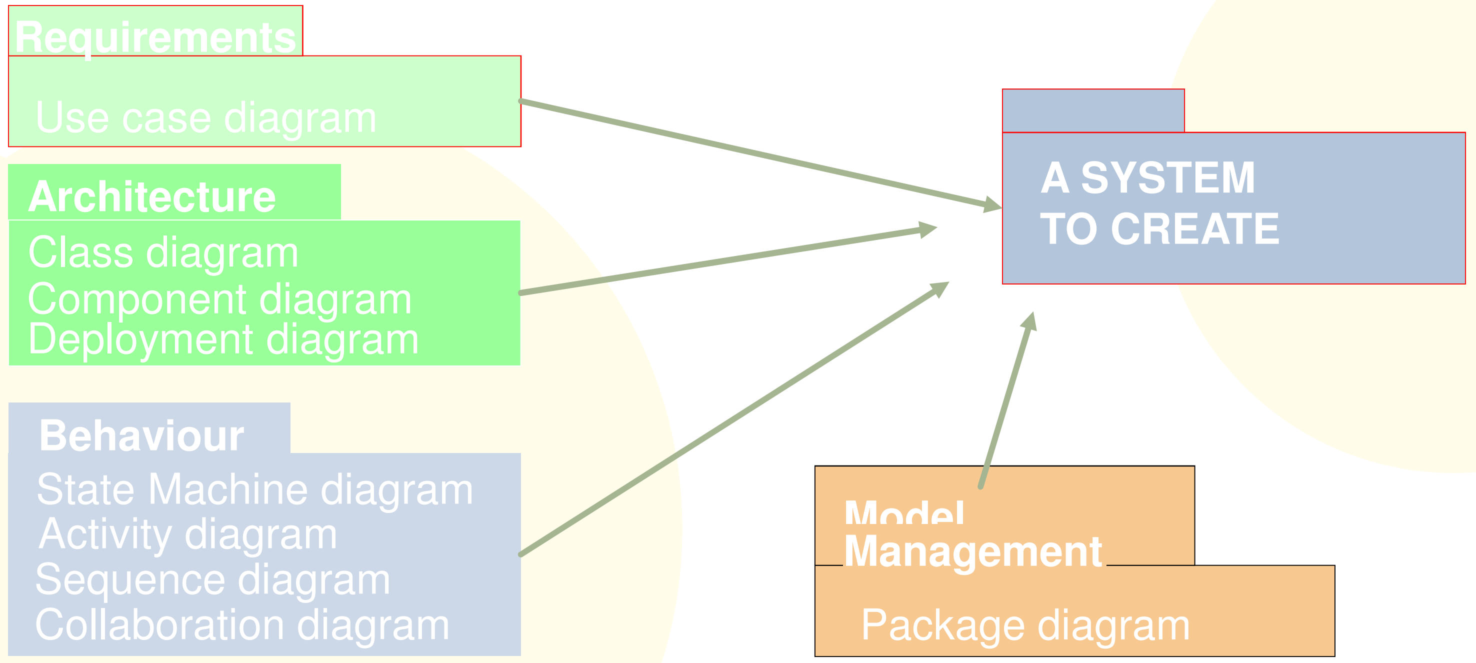
Behavior diagrams emphasize what must happen in the system being modeled. Behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems.

4 Interaction Diagrams

- Communication
- Interaction Overview
- Sequence
- Timing

Interaction diagrams, a subset of behavior diagrams, emphasize the flow of control and data among the things in the system being modeled

Where to use and the meaning of UML



No formal definition exists of how the UML maps
to any particular programming language.
By a UML diagram we can **not** said exactly what equivalent code would look like
We can get rough idea!

UML is Not Enough

- UML provides various diagrams that help to define an application, but it is NOT complete list of all useful diagrams that we might want to use
- Missing:
  - a screen flow diagram,
  - decision tables, etc
- ⇨Don’t hesitate to use a non-UML diagram if no UML diagram suits your purpose.

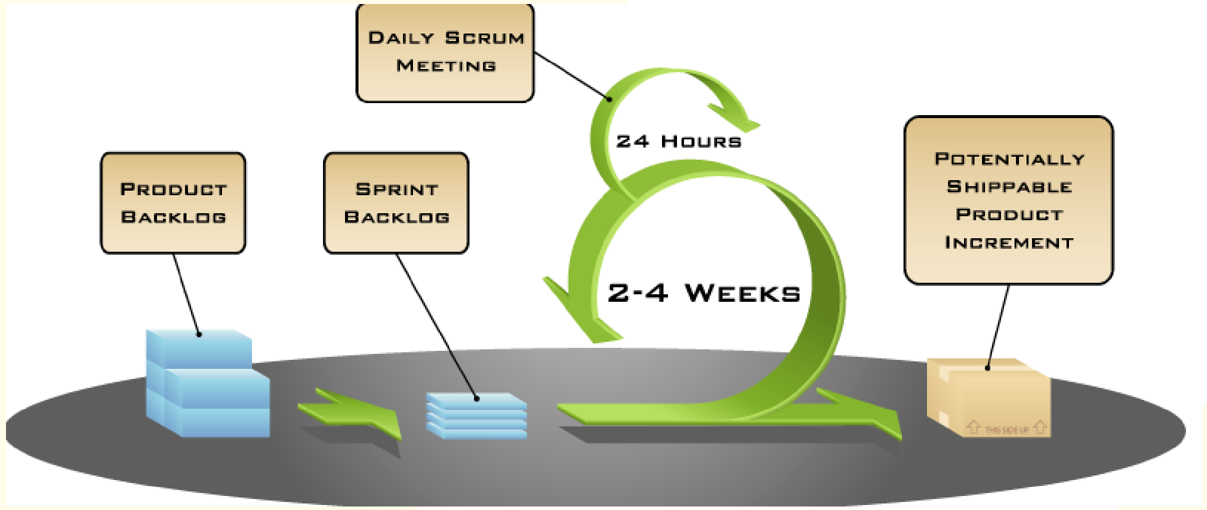
Where to start?

- first:
  - the basic form of class diagram (***structure***)
  - the sequence diagram (***behavior***)
- later:
  - more advanced class diagram notation
  - other behavior diagrams
  - other structure diagrams

Chapter 2: Development Process

UML and development processes

- Modeling techniques does not make any sense without knowing how they fit into a process.
- You can use UML in iterative, agile and waterfall processes



My Three Favorites (and most used Diagrams)

- Use Case
- Class
- Sequence

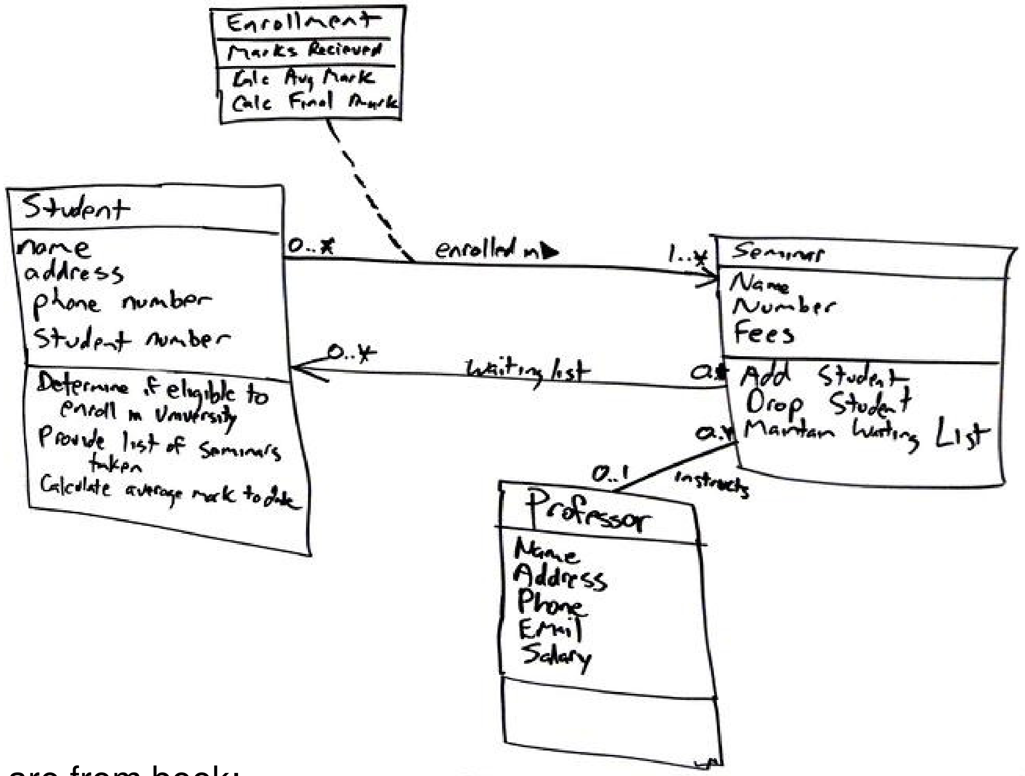
Chapter 3: Class Diagrams / The Essentials

THE MOST USED DIAGRAM

A class diagram

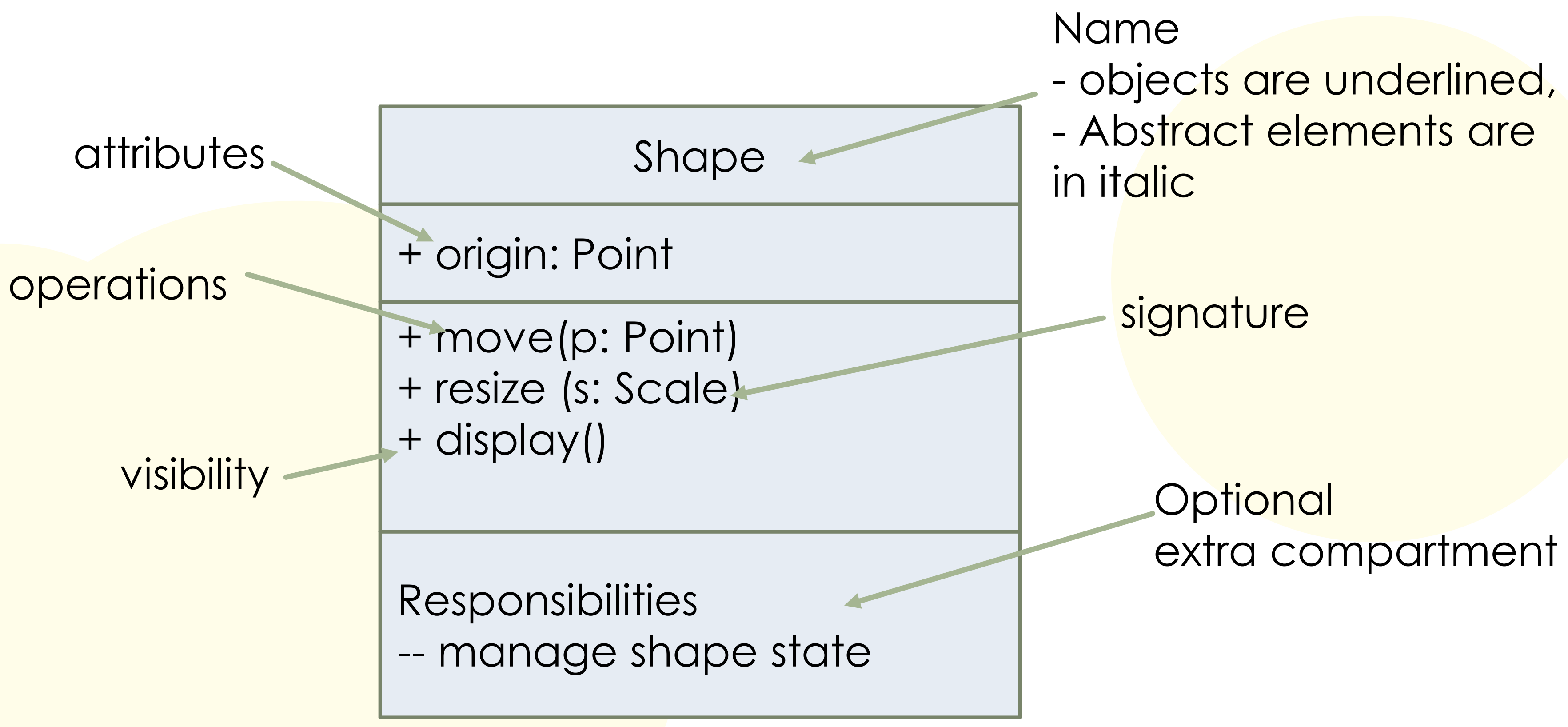
- describes the types of objects in the system and various kind of static relationships that exist among them.
- Class diagram also show the properties and operations of a class and the constraints that apply to the way objects are connected.

Step 1: A Sketch of a conceptual class diagram



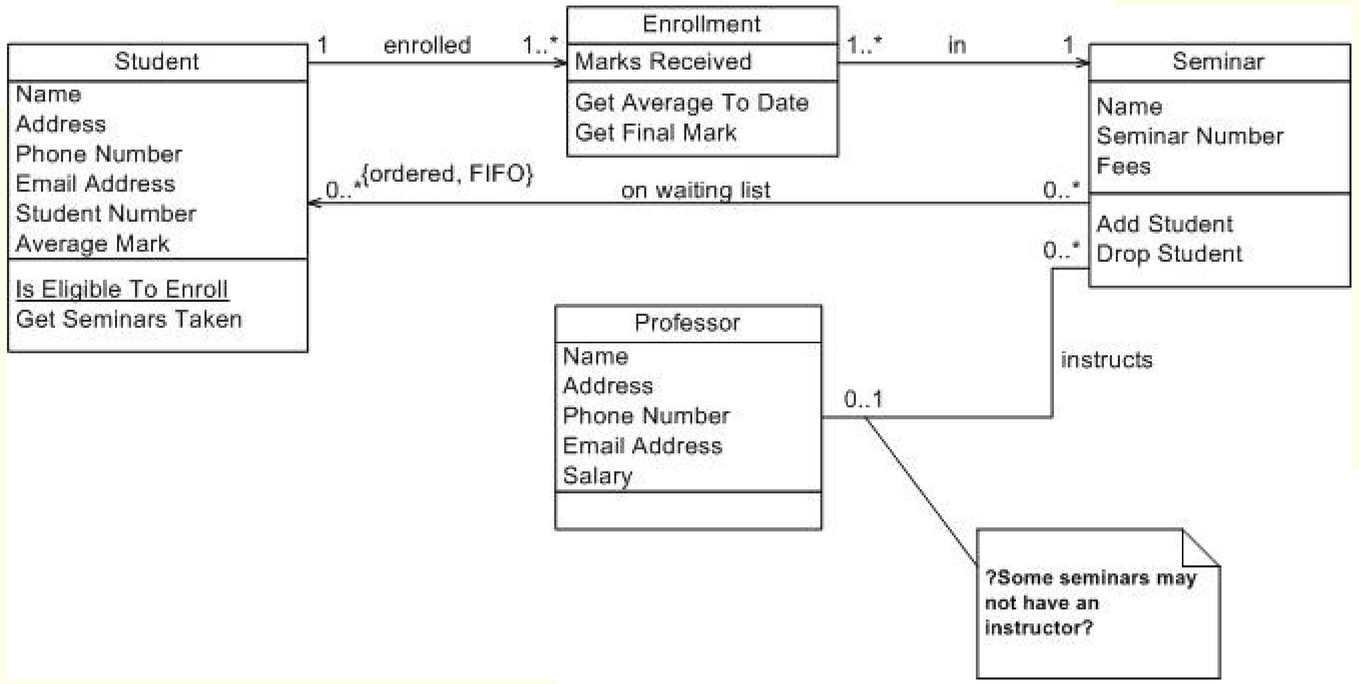
Pictures are from book:
Agile Models Distilled: Potential Artifacts for Agile Modeling

Notation

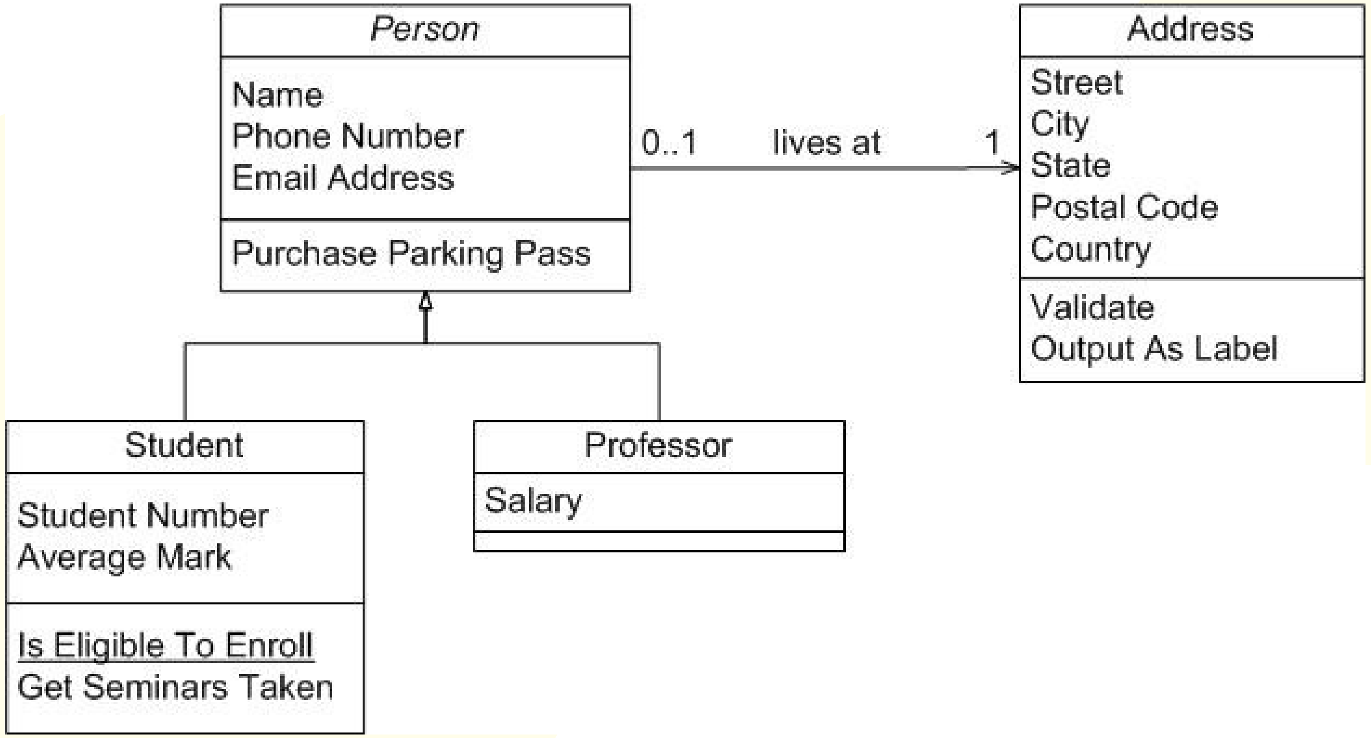


Name
- objects are underlined,
- Abstract elements are in italic
attributes
operations
visibility
signature
Optional extra compartment
Shape
+ origin: Point
+ move(p: Point)
+ resize (s: Scale)
+ display()
Responsibilities
-- manage shape state

Step 2: Initial Conceptual Class Diagram



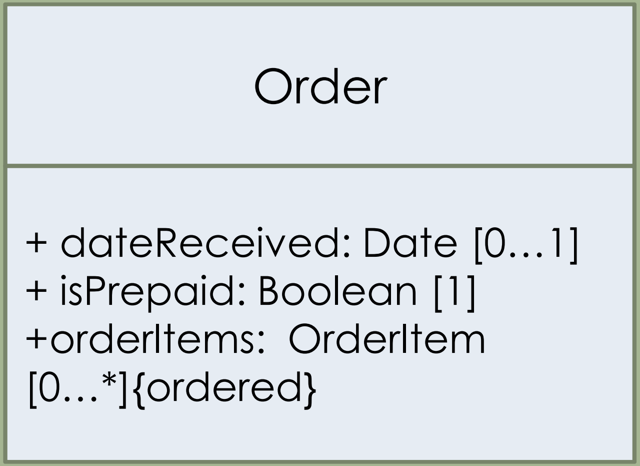
Example of Presenting Inheritance



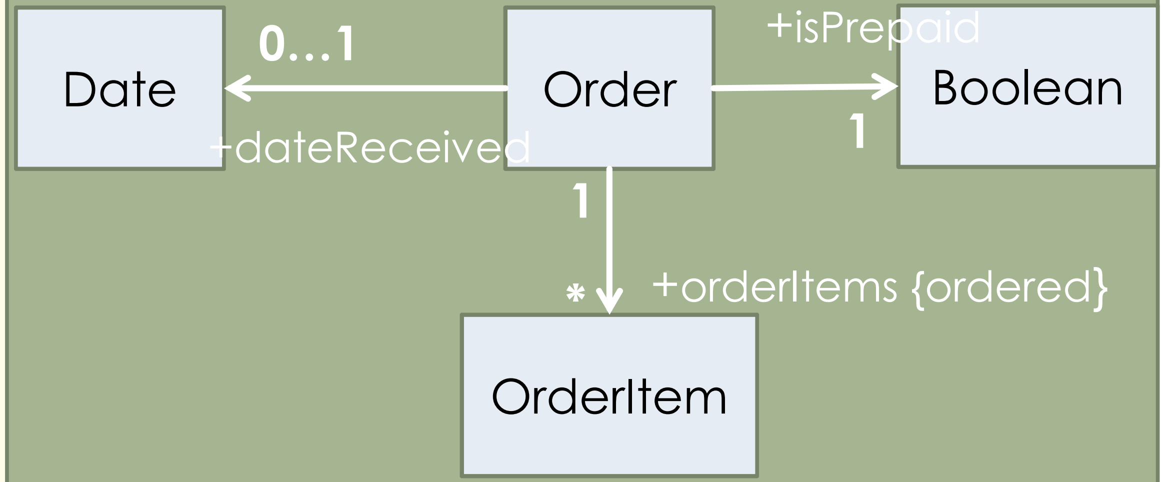
UML gives different ways to present

Example: Showing properties of an order

Option 1:
properties as attributes



Option 2:
properties as associations

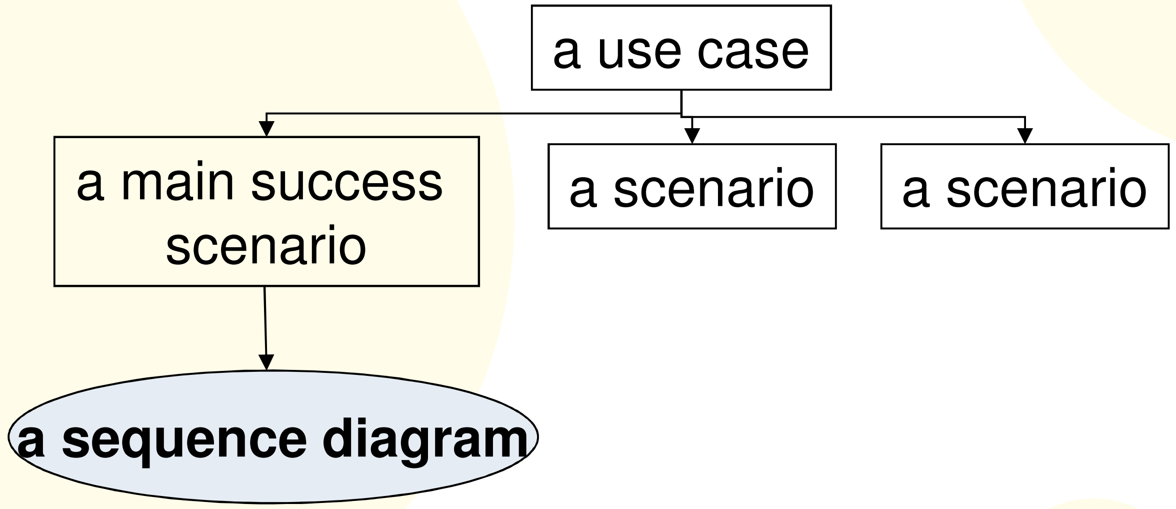


Chapter 4: Sequence Diagrams

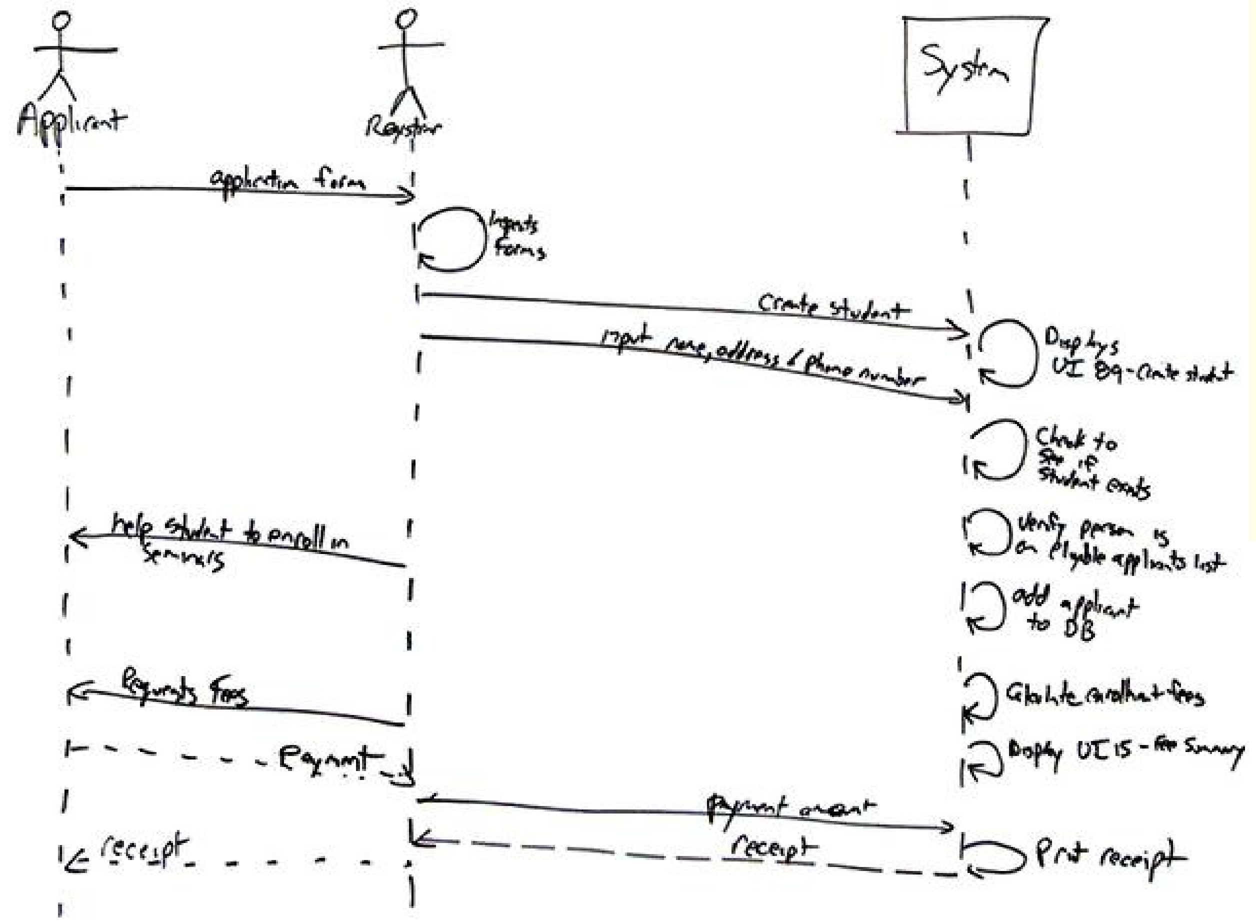
INTERACTION DIAGRAMS DESCRIBES HOW GROUPS OF OBJECTS COLLABORATE IN SOME BEHAVIOR (NOT ALL).

A sequence diagram

- captures the behavior of a single scenario.
- It shows a number of example objects and the messages that are passed between these objects within the use.

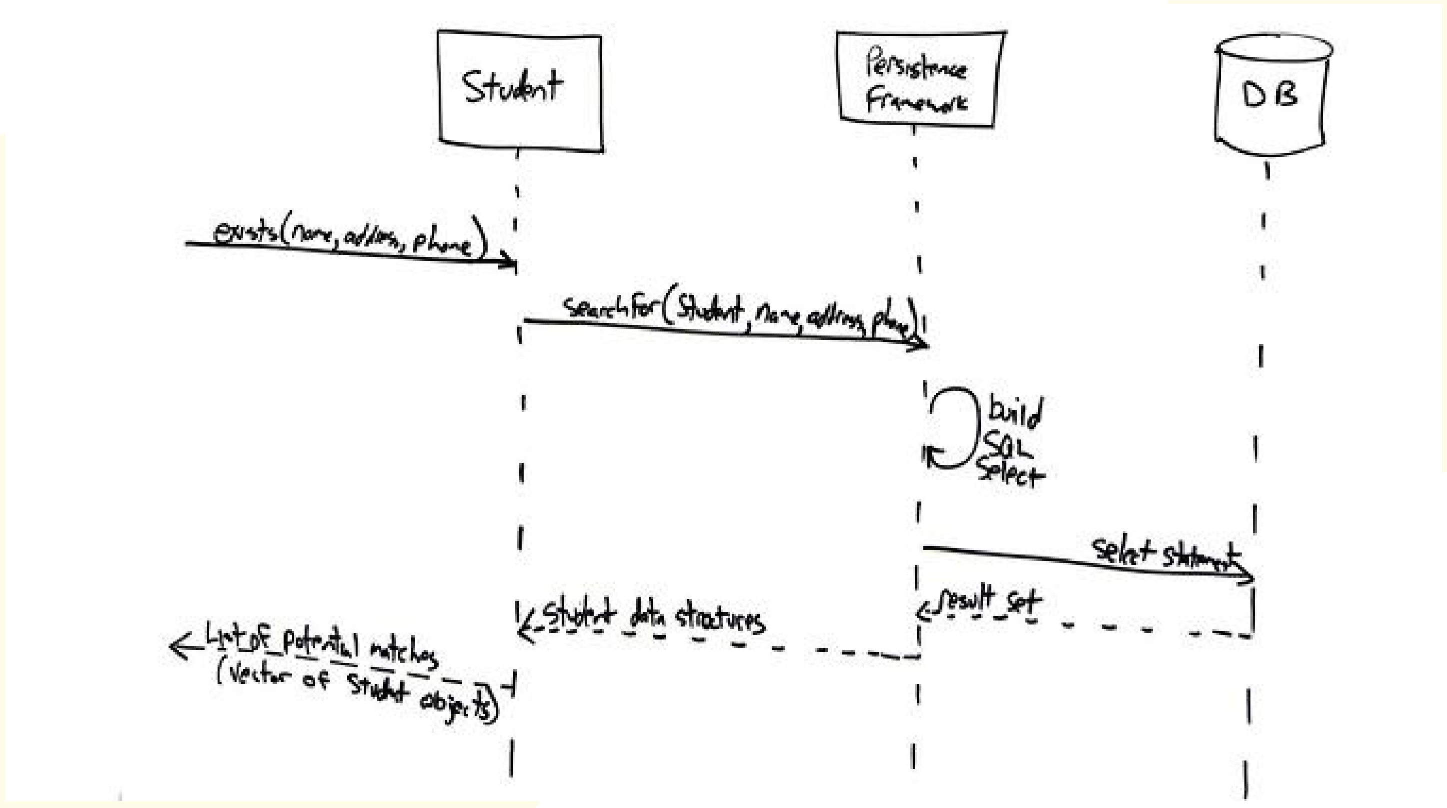


A system-level sequence diagram

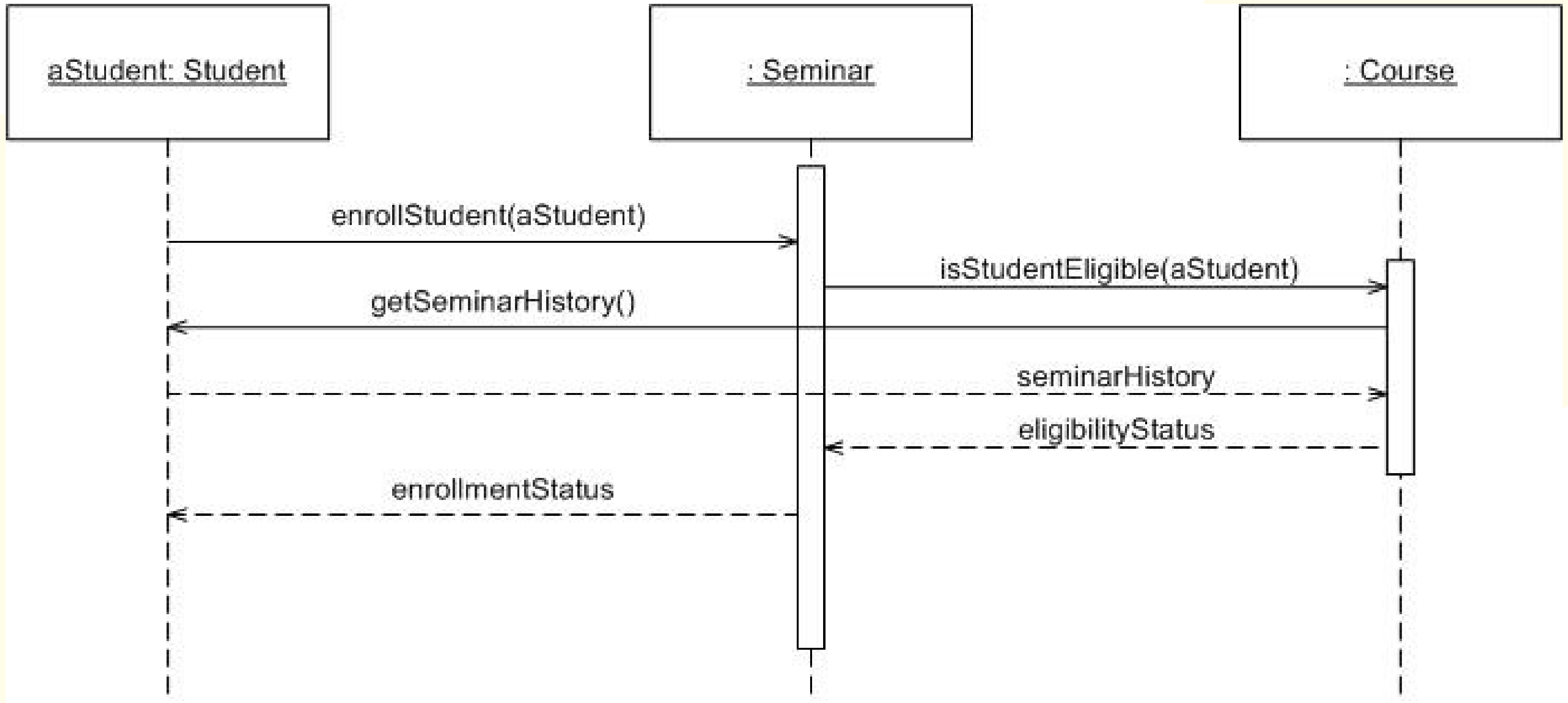

Applicant
Registrar
System
application form
Inputs forms
Create student
Displays UI 89 - Create student
input name, address & phone number
Check to see if student exists
Verify person is on eligible applicants list
add applicant to DB
help student to enroll in seminars
Calculate enrollment fees
Display UI 15 - Fee Summary
Requests fees
Payment
payment amount
receipt
Print receipt
receipt

http://www.agilemodeling.com/artifacts/

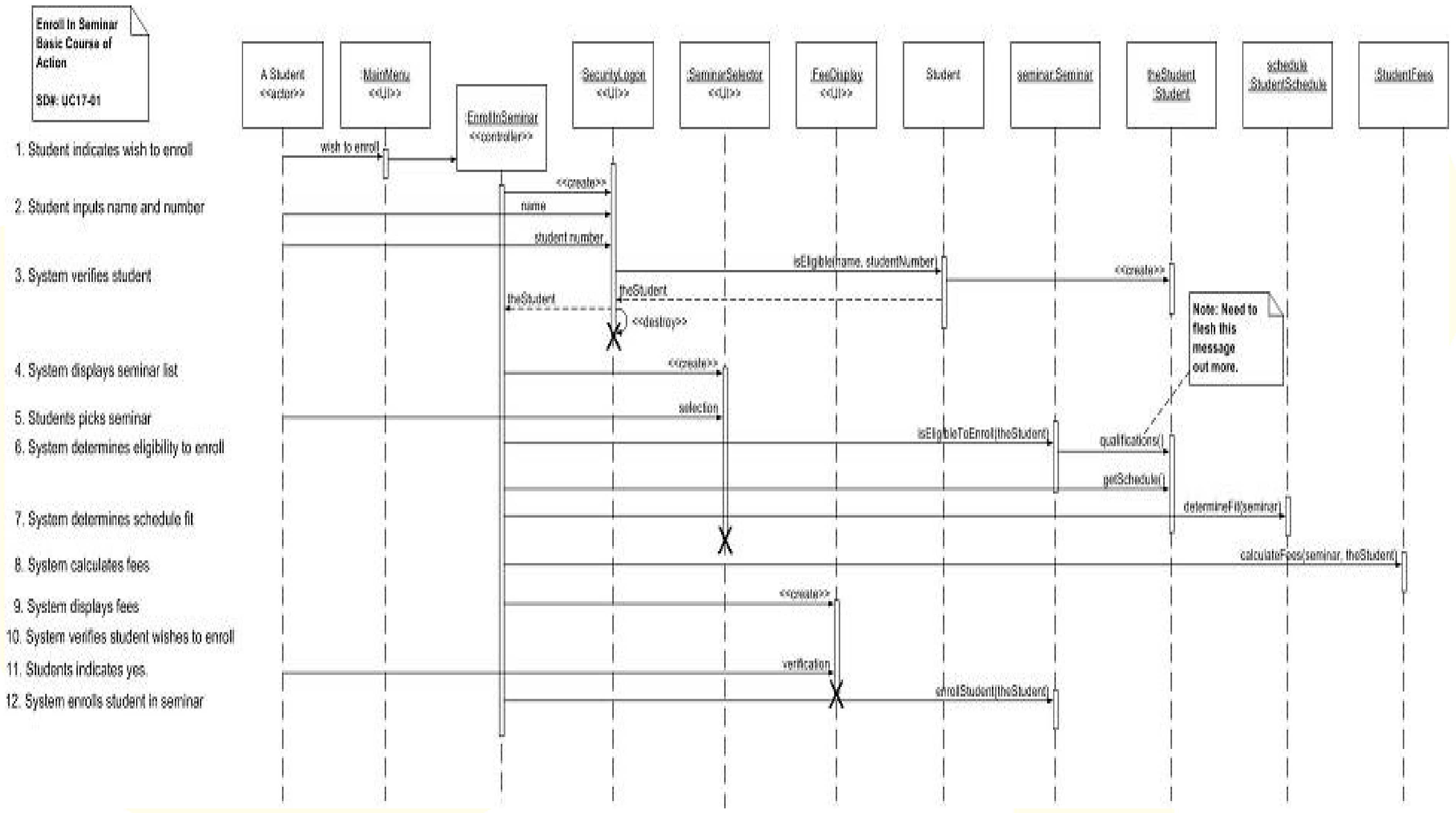
A service-level sequence diagram


Student
Persistence Framework
DB
exists(name, address, phone)
searchFor(Student, name, address, phone)
build SQL Select
select statement
result set
Student data structures
List of potential matches (Vector of Student objects)

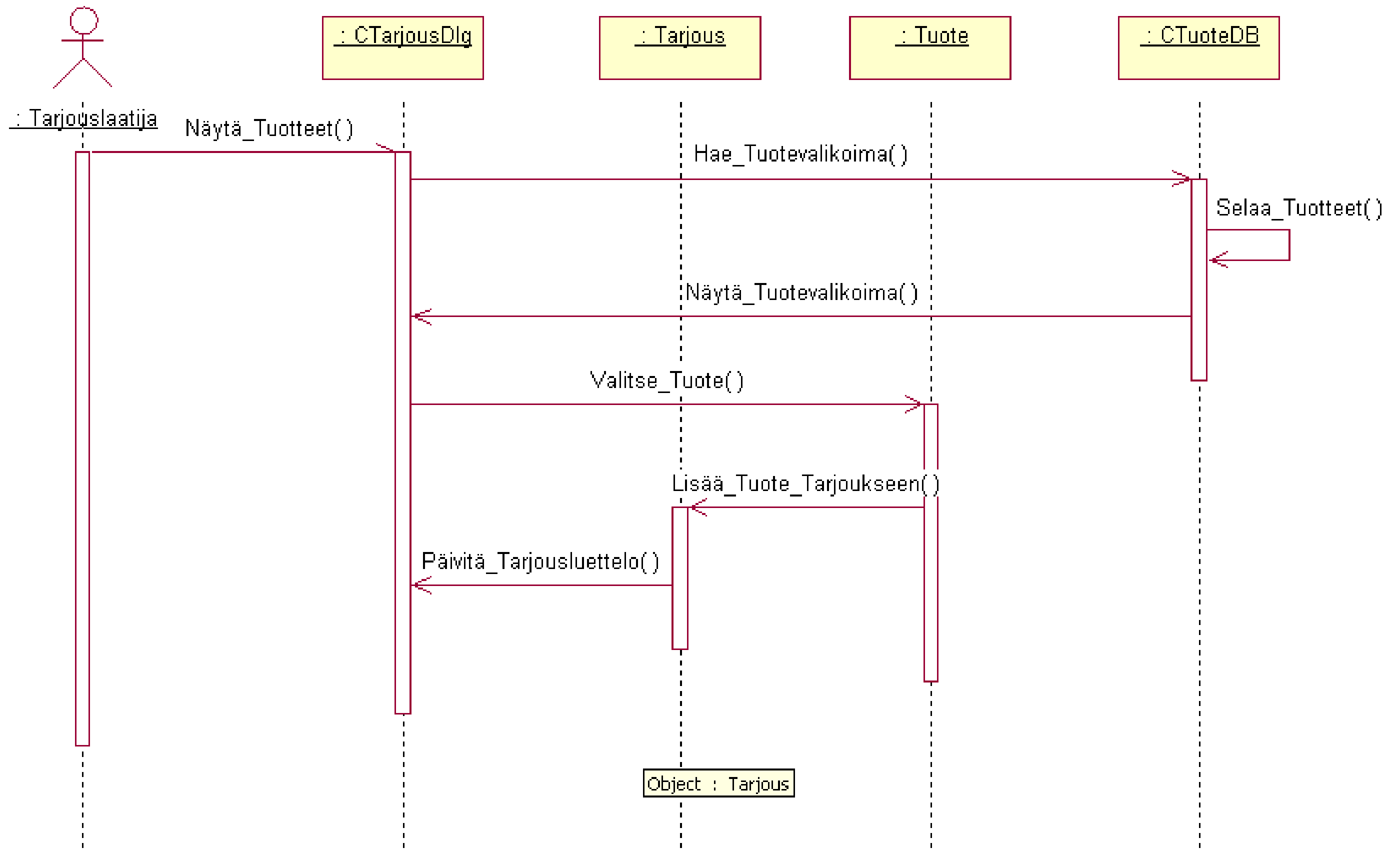
A method in sequence diagram



A use case in a sequence diagram


Enroll in Seminar Basic Course of Action
SD#: UC17-01
A Student <<actor>>
:MainMenu <<UI>>
:EnrollInSeminar <<controller>>
:SecurityLogon <<UI>>
:SeminarSelector <<UI>>
:FeeDisplay <<UI>>
Student
seminar:Seminar
theStudent :Student
schedule :StudentSchedule
:StudentFees
1. Student indicates wish to enroll
2. Student inputs name and number
3. System verifies student
4. System displays seminar list
5. Students picks seminar
6. System determines eligibility to enroll
7. System determines schedule fit
8. System calculates fees
9. System displays fees
10. System verifies student wishes to enroll
11. Students indicates yes.
12. System enrolls student in seminar
wish to enroll
<<create>>
name
student number
isEligible(name, studentNumber)
<<create>>
theStudent
theStudent
<<destroy>>
Note: Need to flesh this message out more.
<<create>>
selection
isEligibleToEnroll(theStudent)
qualifications()
getSchedule()
determineFit(seminar)
calculateFees(seminar, theStudent)
<<create>>
verification
enrollStudent(theStudent)

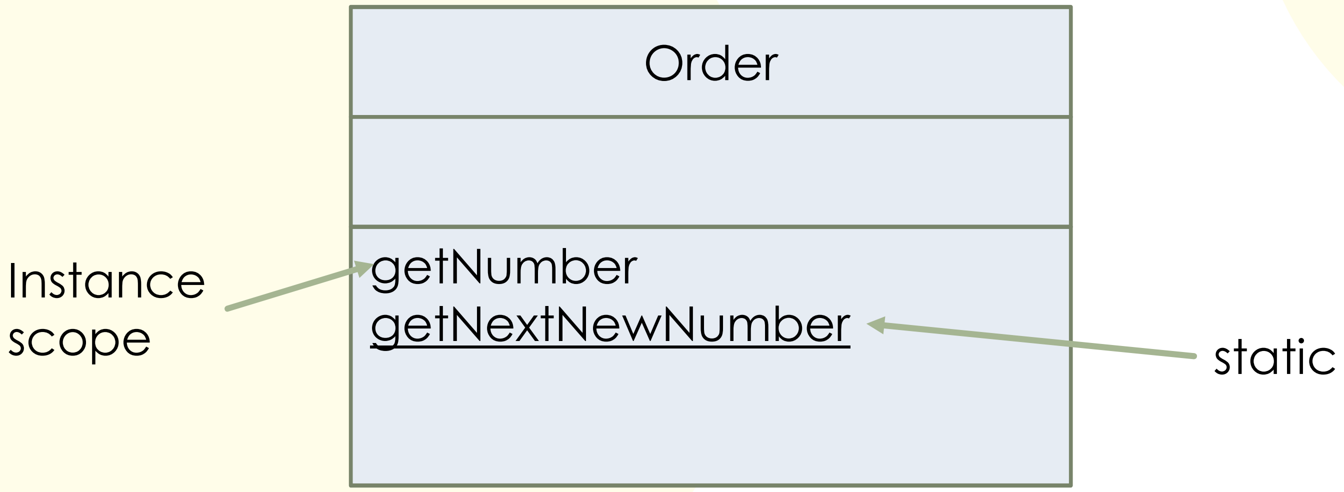
A sequence diagram in The Finnish UML Software Prosa



Chapter 5: Class Diagrams / Advanced Concepts

Static notation

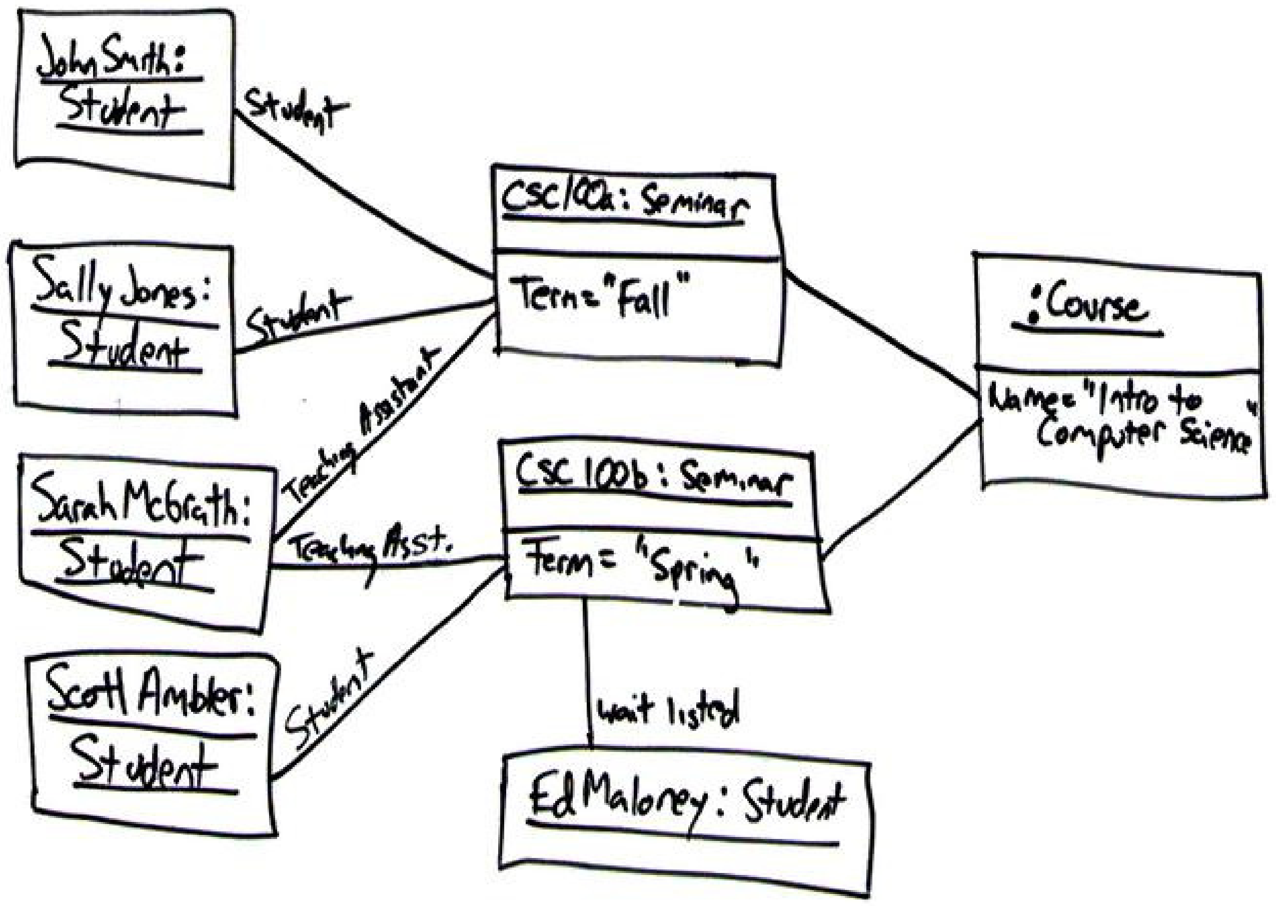
- Static features are <u>underlined</u> on a class diagram



Chapter 6: Object Diagrams

An object diagram

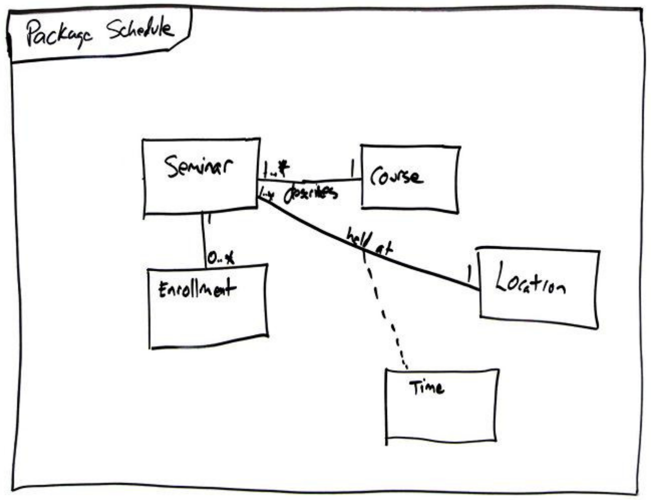
- is a snapshot of the objects in a system at a point in time

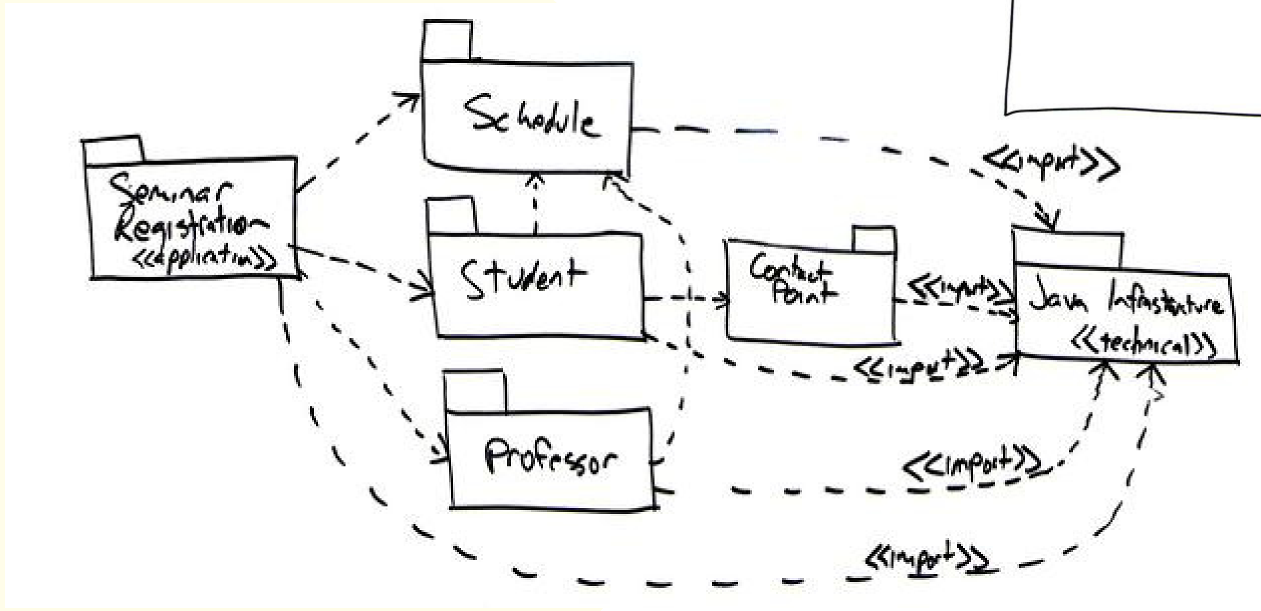


Chapter 7: Package Diagrams

A package diagram

- shows packages and their dependies.

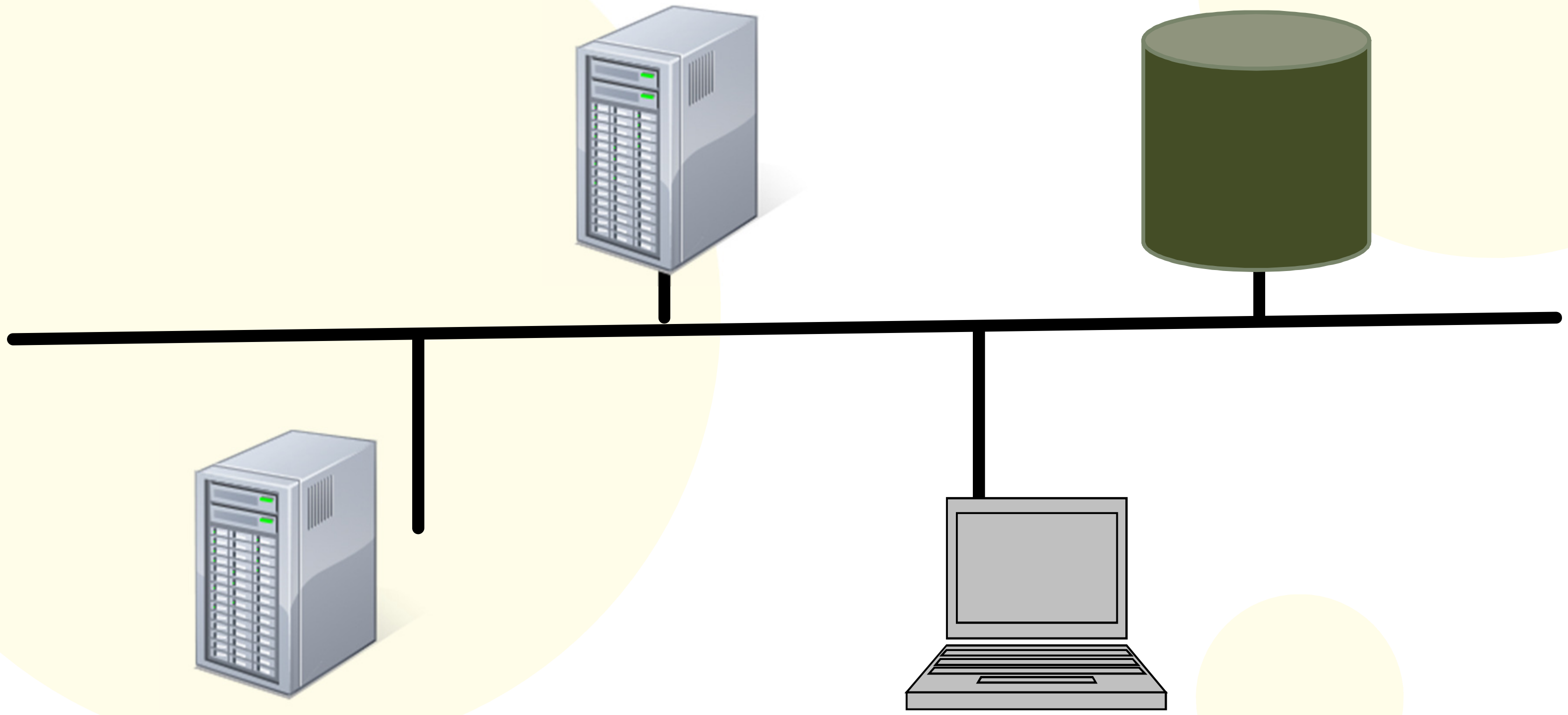




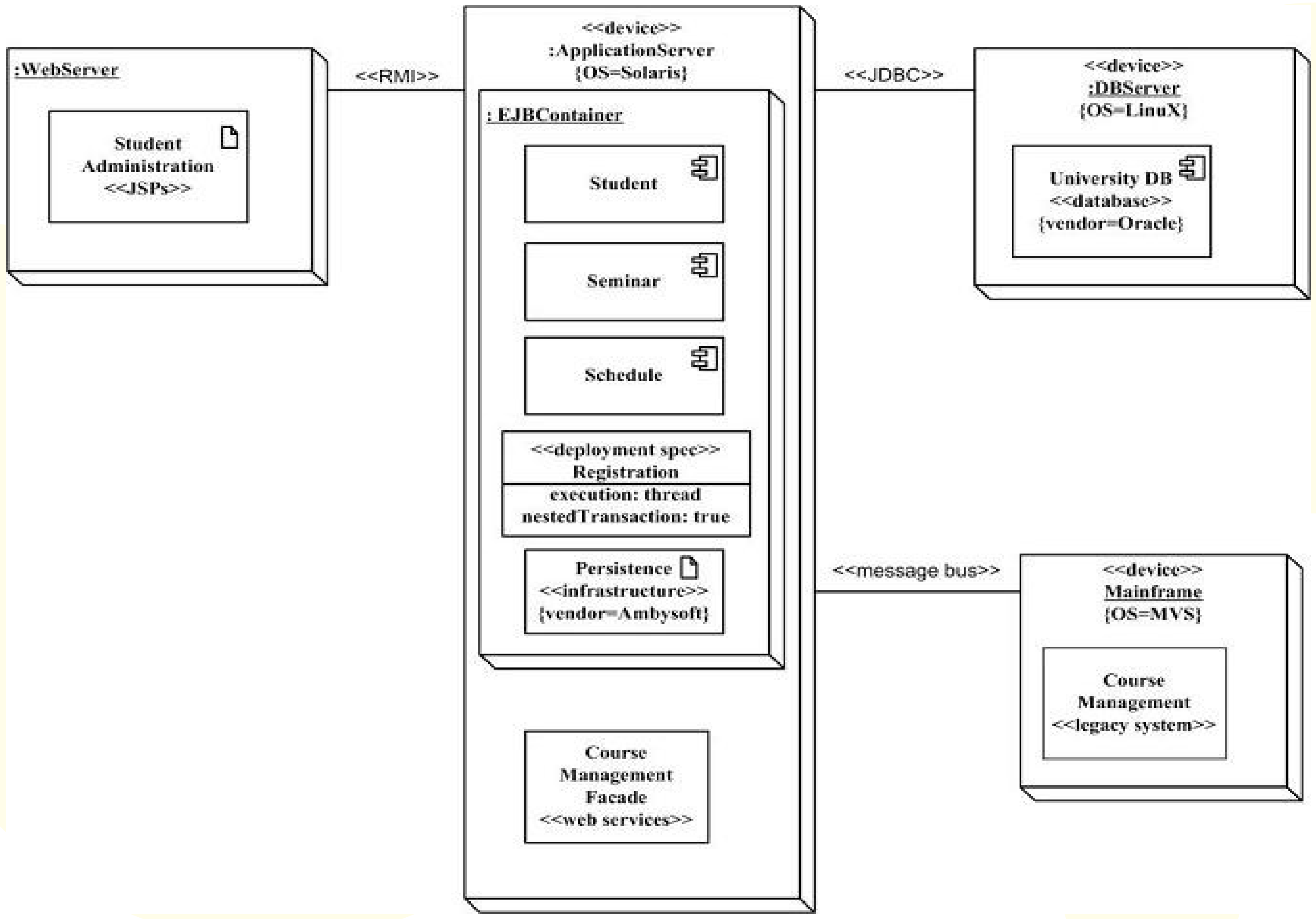
Chapter 8: Deployment Diagrams

A deployment diagram

- shows a system’s physical layout, revealing which pieces of software are running on what pieces of hardware.



Deployment diagram for the university information system


: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>>

Chapter 9: Use Cases

Where and when to use

- use cases are capturing the functional requirements of a system
- describes the typical interaction between the users of a system and the system itself
- an use case diagram itself it is a good start, normally we need written use case stories also

A use case and scenarios

- **a use case** is a set of scenarios tied together by a common user goal
- **a scenario** is a sequence of steps describing an interaction between a user and a system.
- **the main success scenario** is that flow of actions which a user achieves his/her main goal successful

A use case

- a use case is written description, not a diagram (activity diagram or sequence diagram can be used with)

Buy a product
Actor: Customer
Main success scenario:
1) Customer browse catalog and select items to buy
2) …
3) …

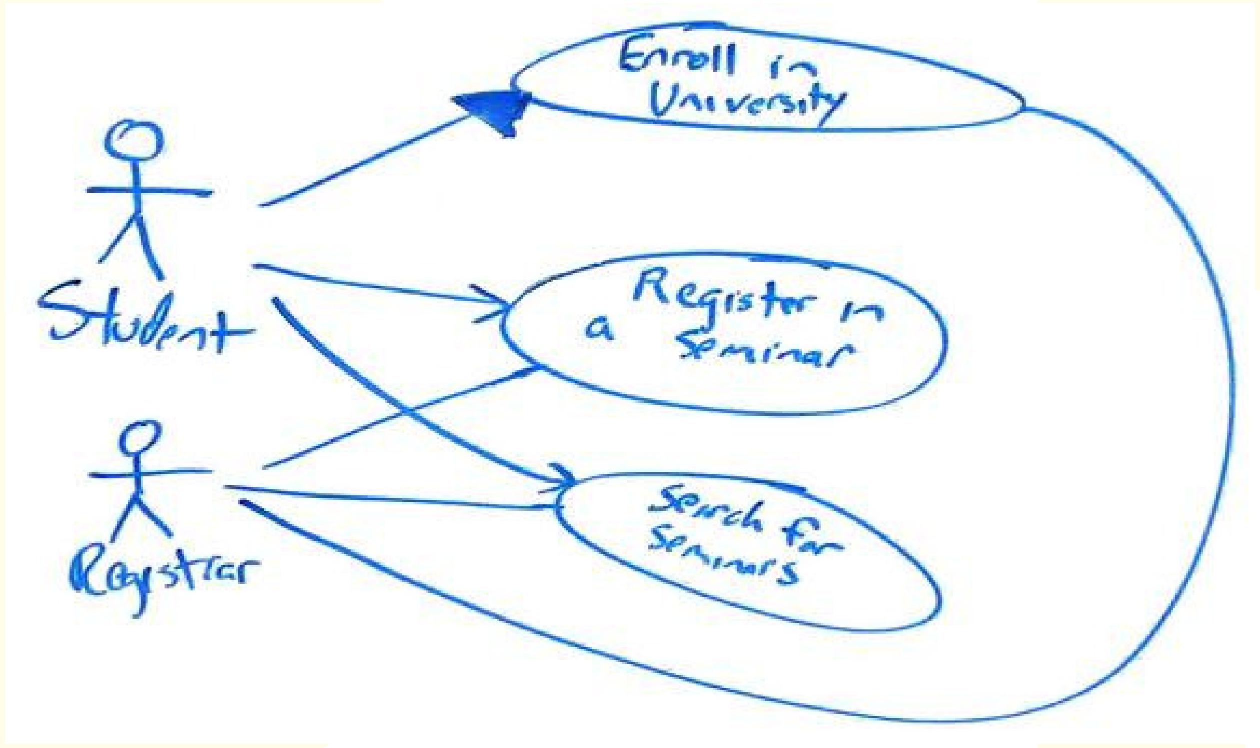
Extensions & exceptions:
3a) Customer can…

Different Use Case

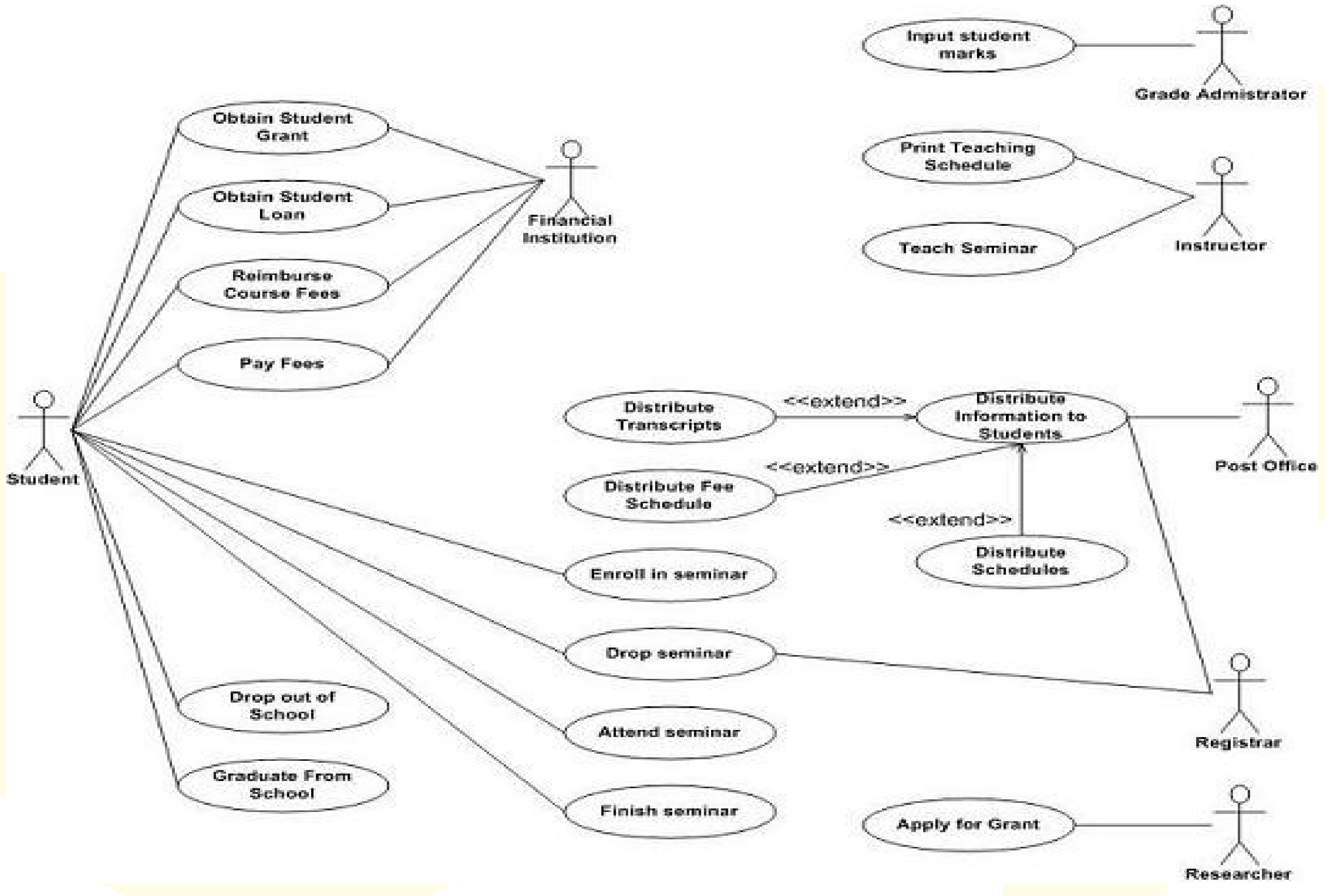
- System use cases
- Business use cases
- Mis use cases

A sketch of a use case diagram

- Diagram shows actors of a system, its use cases and relations of them

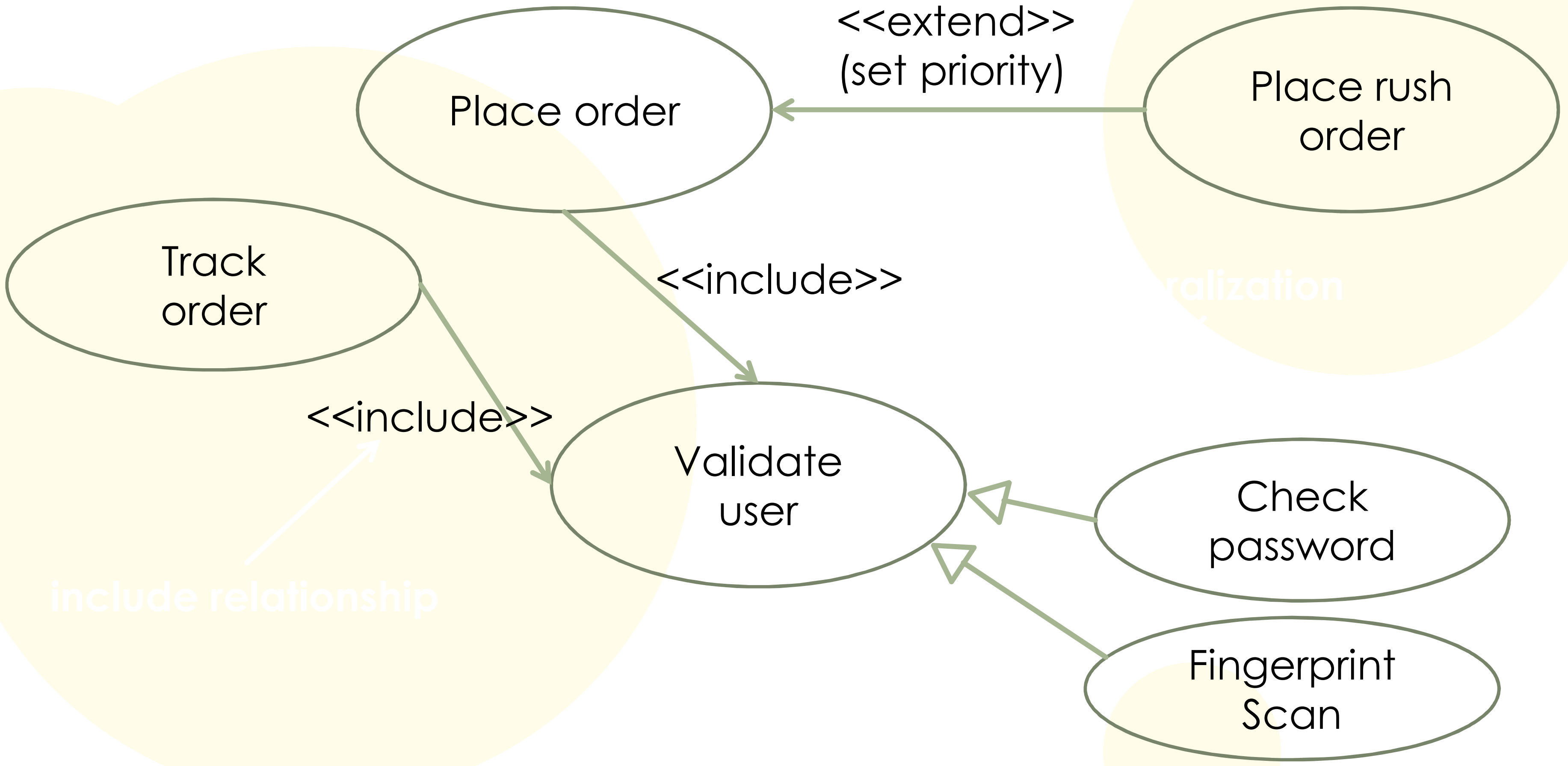


A system use case diagram



Use case reuse

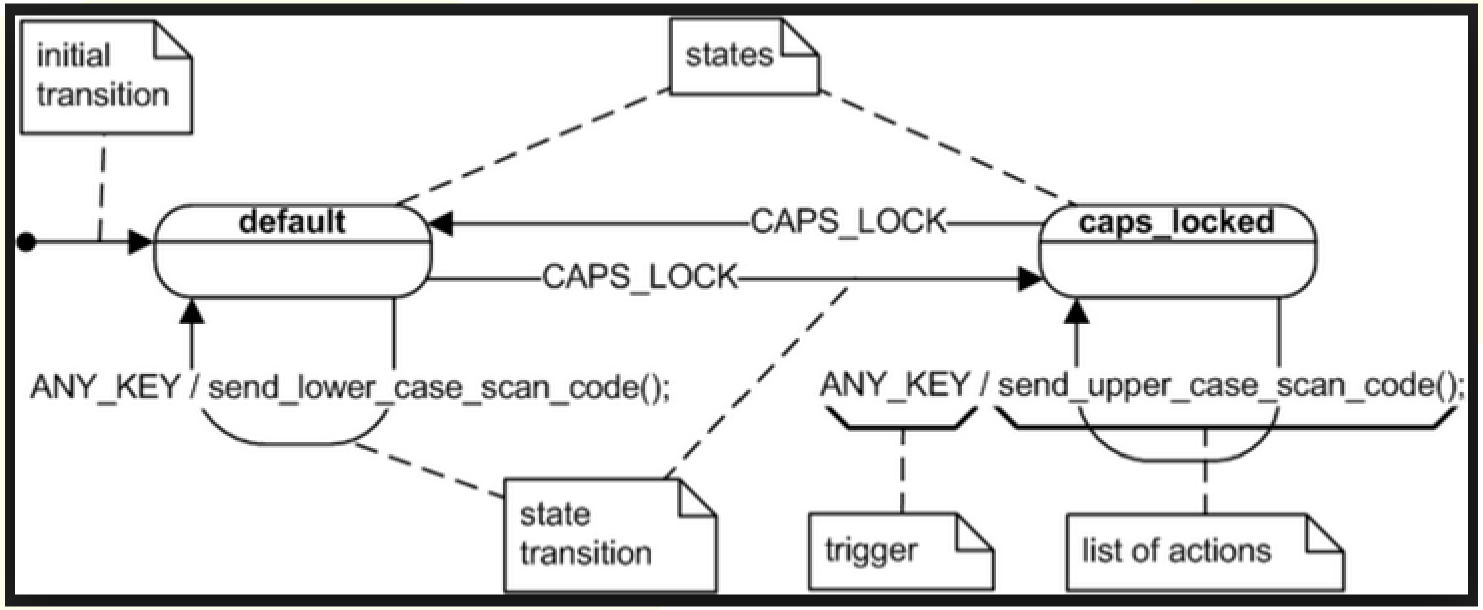
- Extend relation, include relation and generalization



Chapter 10: State Machine Diagrams

State machine Diagram

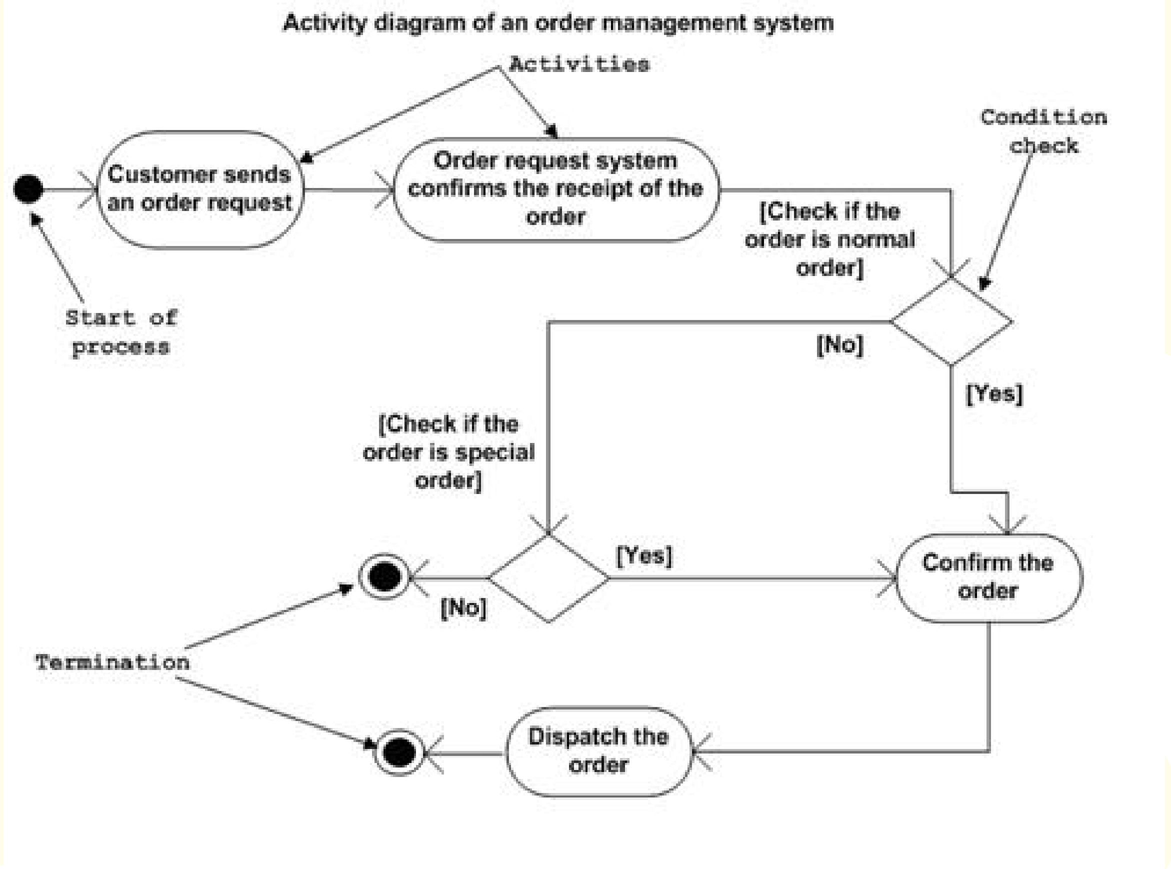
- describes the states and state transitions of the system



Chapter 11: Activity Diagrams

Activity Diagram

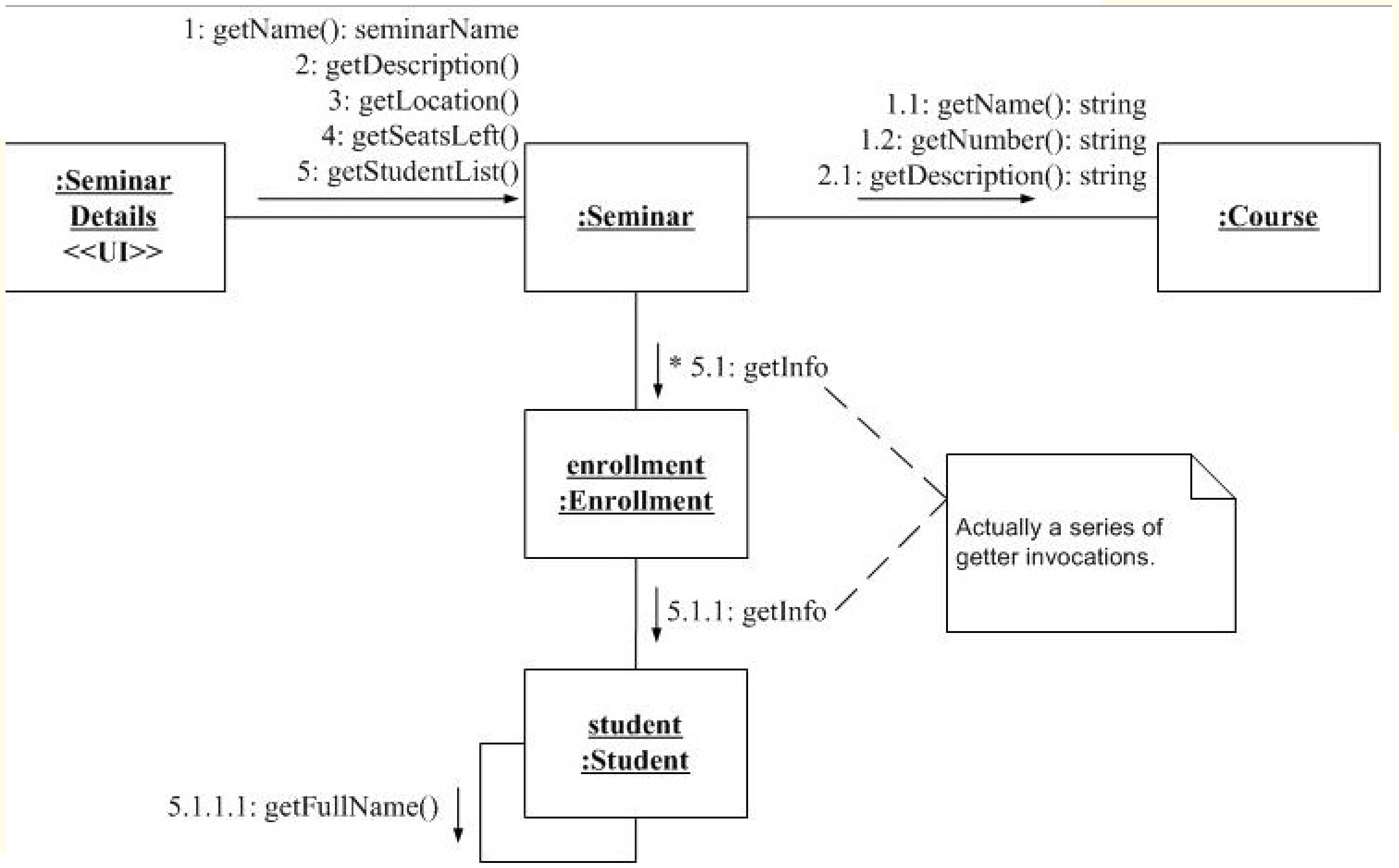
- Describe procedural logic, business process and work flow.


Activity diagram of an order management system
Activities
Condition check
Customer sends an order request
Order request system confirms the receipt of the order
[Check if the order is normal order]
Start of process
[No]
[Yes]
[Check if the order is special order]
[Yes]
[No]
Confirm the order
Termination
Dispatch the order

Chapter 12: Communication Diagrams

Communication Diagram

- Presents the interaction between objects or parts



1: getName(): seminarName
2: getDescription()
3: getLocation()
4: getSeatsLeft()
5: getStudentList()
1.1: getName(): string
1.2: getNumber(): string
2.1: getDescription(): string
:Seminar Details <<UI>>
:Seminar
:Course
* 5.1: getInfo
enrollment :Enrollment
Actually a series of getter invocations.
5.1.1: getInfo
student :Student
5.1.1.1: getFullName()