Software Engineering with Objects and Components

Massimo Felici

Room 1402, JCMB, KB

0131 650 5899

mfelici@inf.ed.ac.uk

Administration

- Look at the SEOC1 course webpage:
  http://www.inf.ed.ac.uk/teaching/courses/seoc1/
- **Tutorials** begin in **week 3**; Frequency: once a week; Maximum 12 people per tutorial group
- **Course Resources**:
  - **Main Course Book**: UML, Schaum's Outline Series, Simon Bennett, John Skelton and Ken Lunn, McGraw-Hill, 2001, ISBN 0-07-709673-8.
  - Lecture Notes and **References**
  - **Software**: Argo/UML and Java
- **Coursework**:
  - in small teams (approx 3-4 people);
  - two deliverables equally weighted
    - 1st deliverable: **Monday, 31st October**
    - 2nd deliverable: **Monday, 5th December**
- **Assessment**:
  - 25% coursework; 75% degree examination

Software Engineering

- Software Engineering Institute (SEI) motto:
  - *The right software. Delivered defect free, on time and on cost, every time.*
- Software Engineering studies:
  - How to make software that is *"fit for purpose".*
  - **"fit for purpose"**:
    - good enough – functionally, non-functionally, meets constraints of the environment, law, ethics and work practice.
  - How to meet **time** and **financial constraints** on delivery.
- We still fail too often
  - see a **Collection of Software Bugs** by Prof. Thomas Hackle **[SEOC1 Resource webpage]**

An Example: Patriot Missile

- **Accident Scenario**: On February 25, 1991, during the Gulf War, an American Patriot Missile battery in Dharan, Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people.
- A report of the General Accounting office, <u>GAO/IMTEC-92-26</u>, entitled *Patriot Missile Defense: **Software Problem Led to System Failure** at Dhahran, Saudi Arabia* reported on the cause of the failure.

An Example: Patriot Missile continued...

- **Fault**: inaccurate calculation of the time since boot due to computer arithmetic errors.
  - The time in tenths of second as measured by the system's internal clock was multiplied by 1/10 to produce the time in seconds.
  - This calculation was performed using a 24 bit fixed point register. In particular, the value 1/10, which has a non-terminating binary expansion, was chopped at 24 bits after the radix point.
- **Error**: The small chopping error, when multiplied by the large number giving the time in tenths of a second, lead to a significant error. Indeed, the Patriot battery had been up around 100 hours, and an easy calculation shows that the resulting time error due to the magnified chopping error was about **0.34 seconds**.
  - the binary expansion of 1/10 is

  0.0001100110011001100110011001100....
  - Now the 24 bit register in the Patriot stored instead

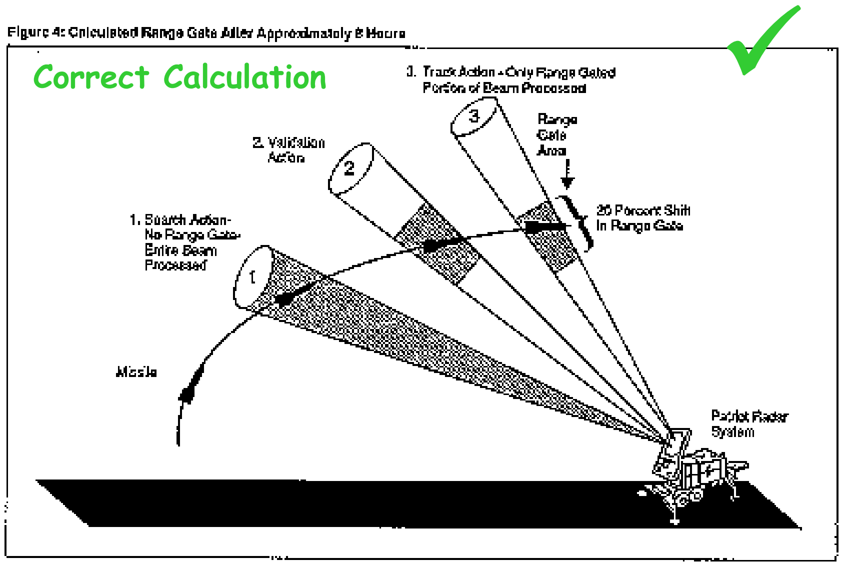
  0.00011001100110011001100 introducing an error of

  0.0000000000000000000000011001100... binary, or about **0.000000095** decimal.
  - Multiplying by the number of tenths of a second in 100 hours gives 0.000000095×100×60×60×10=0.34.

An Example: Patriot Missile continued...

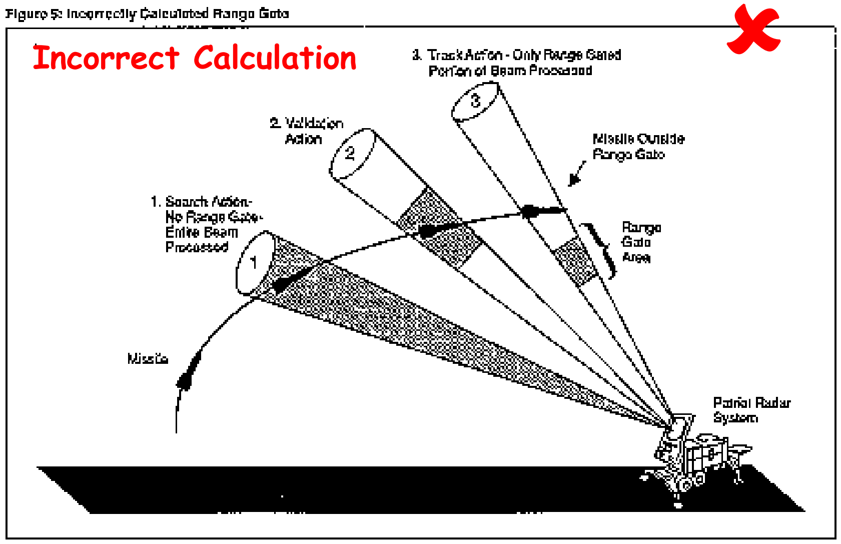
- **Failure**: A Scud travels at about 1,676 meters per second, and so travels more than 500 meters in this time.
- This was far enough that the incoming Scud was outside the **"range gate"** that the Patriot tracked.
- Ironically, the fact that the bad time calculation had been improved in some parts of the code, but not all, contributed to the problem, since it meant that the inaccuracies did not cancel.

Figure 4: Calculated Range Gate After Approximately 8 Hours



✓
Correct Calculation
3. Track Action - Only Range Gated Portion of Beam Processed
Range Gate Area
2. Validation Action
20 Percent Shift In Range Gate
1. Search Action - No Range Gate - Entire Beam Processed
Missile
Patriot Radar System

Figure 5: Incorrectly Calculated Range Gate

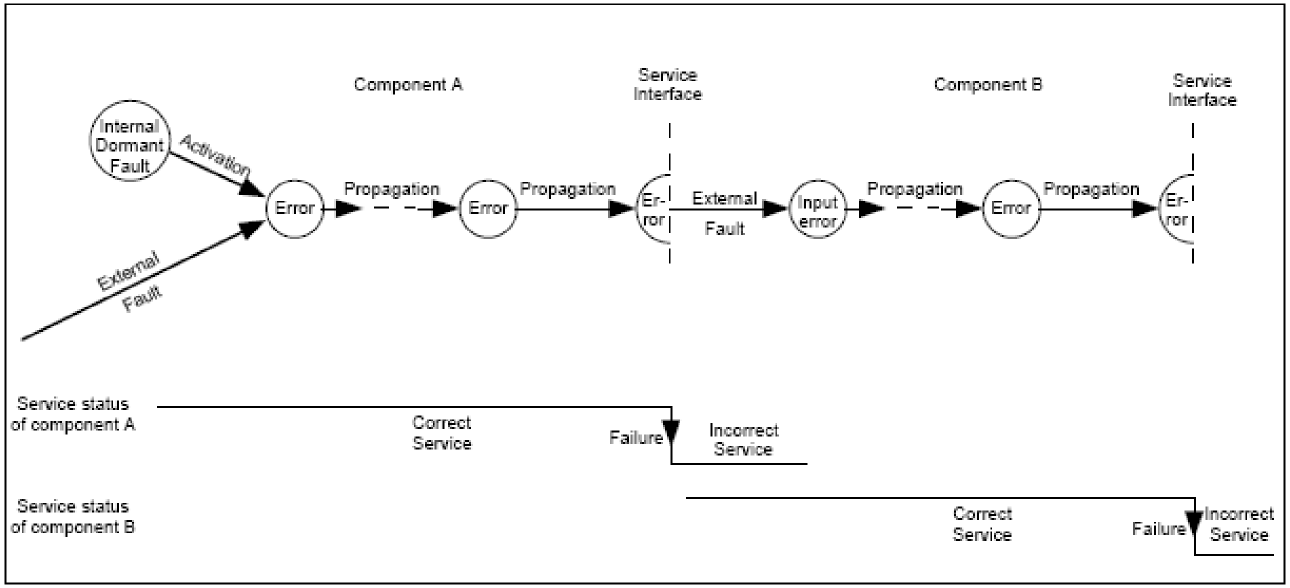


✗
Incorrect Calculation
3. Track Action - Only Range Gated Portion of Beam Processed
2. Validation Action
Missile Outside Range Gate
1. Search Action - No Range Gate - Entire Beam Processed
Range Gate Area
Missile
Patriot Radar System

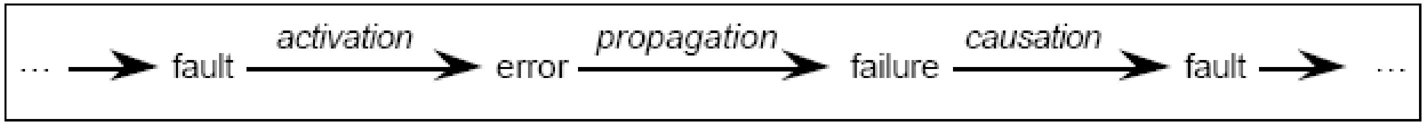
An Example: Patriot Missile...conclusions

- Containing coding errors is very hard
  - seemingly insignificant errors result in major changes in behaviour
- Original fix suggested a change in procedures
  - reboot every 30 hours – impractical in operation
- Patriot is atypical
  - coding bugs rarely cause accidents alone
- Maintenance failure
  - failure of coding standards and traceability

The Pathology of Failure



- Relationship between **Faults**, **Errors** and **Failures**


$$\cdots \longrightarrow \text{fault} \xrightarrow{\textit{activation}} \text{error} \xrightarrow{\textit{propagation}} \text{failure} \xrightarrow{\textit{causation}} \text{fault} \longrightarrow \cdots$$

- The fundamental chain of dependability threats

Other Case Studies - Readings

- *Ian Sommerville. Software Engineering Case Studies, 2004.*
  - *The Ariane 5 Launcher Failure*
  - *The London Ambulance fiasco*
  - *Airbus Flight Control System*

[SEOC1 Resource webpage]

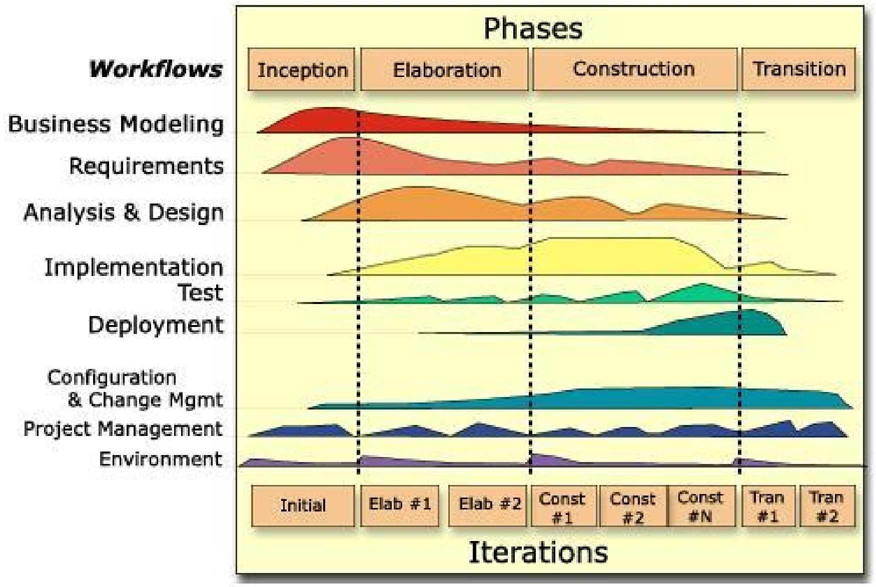
- *Medical Devices: The Therac-25*
  - Nancy Leveson. *Safeware: System Safety and Computers.* Addison-Wesley, 1995

[SEOC1 Resource webpage]

Software Engineering

- We will study the following areas:
  - **Software Requirements**: the activities involved in gaining an accurate idea of what the users of the system want it to do.
  - **Software Design**: the design of a system to meet the requirements.
  - **Software Construction**: the realisation of the design as a program.
  - **Software Testing**: the process of checking the code meets the design.
  - **Software Configuration, Operation and Maintenance**: major cost in the lifetime of systems.
- These are the essential activities
- How we deploy effort and arrange these activities is part of Software Engineering Processes

(Rational) Unified Process - RUP



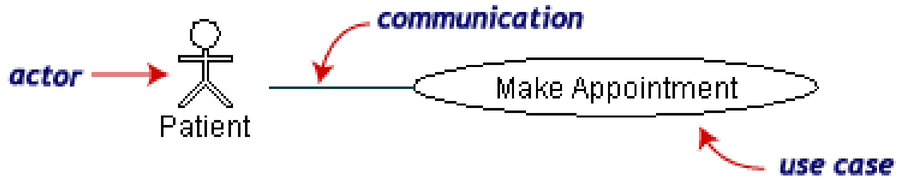
References

- **Software Engineering**
  - Ian Sommerville. Software Engineering. 7th Edition, Addison-Wesley, 2004.
- **Safety-critical Systems**
  - Nancy G. Leveson. Safeware: System Safety and Computers. Addison-Wesley, 1995
  - Neil Story. Safety-Critical Computer Systems. Addison-Wesley, 1996.

Models Supporting SE

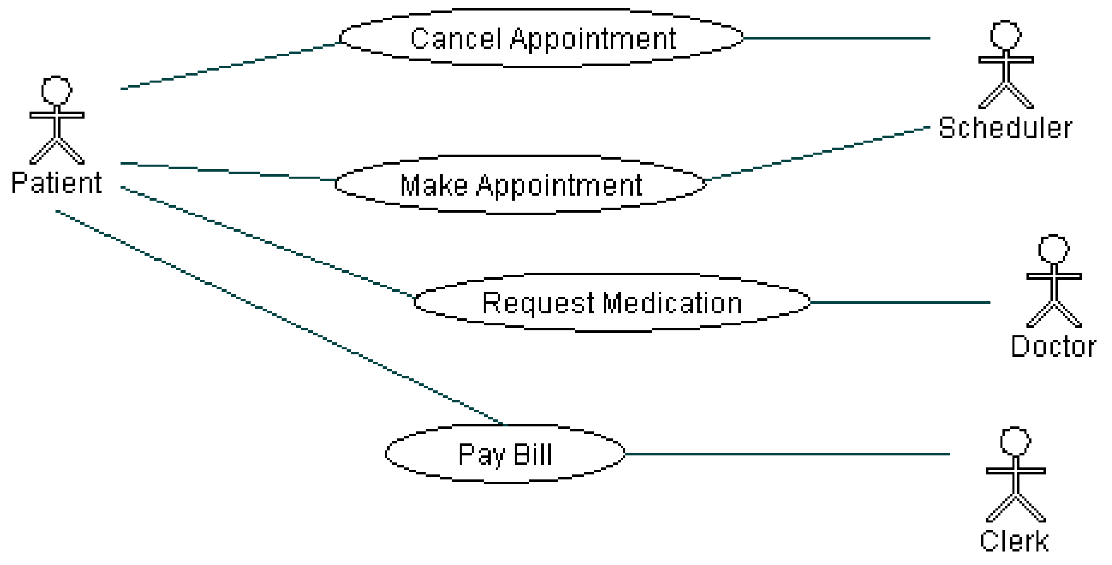
- UML provides a range of **graphical notations** that capture various aspects of the engineering process.
- Provides a common notation for various different facets of systems.
- Provides the basis for a range of consistency checks, validation and verification procedures.
- Provides a common set of languages and notations that are the basis for creating tools.

UML: Use Case Diagrams



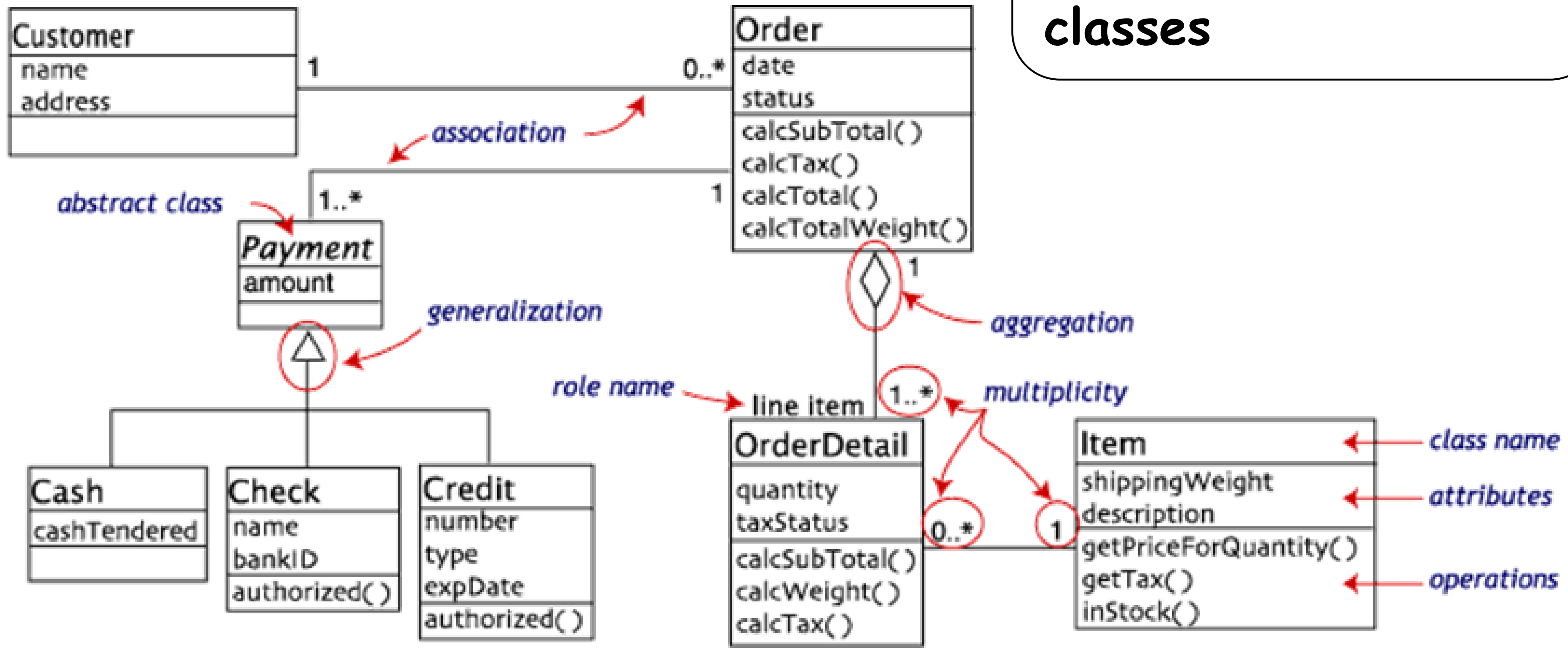
Used to support requirements Capture and analysis

Show the actors' Involvement in System activities



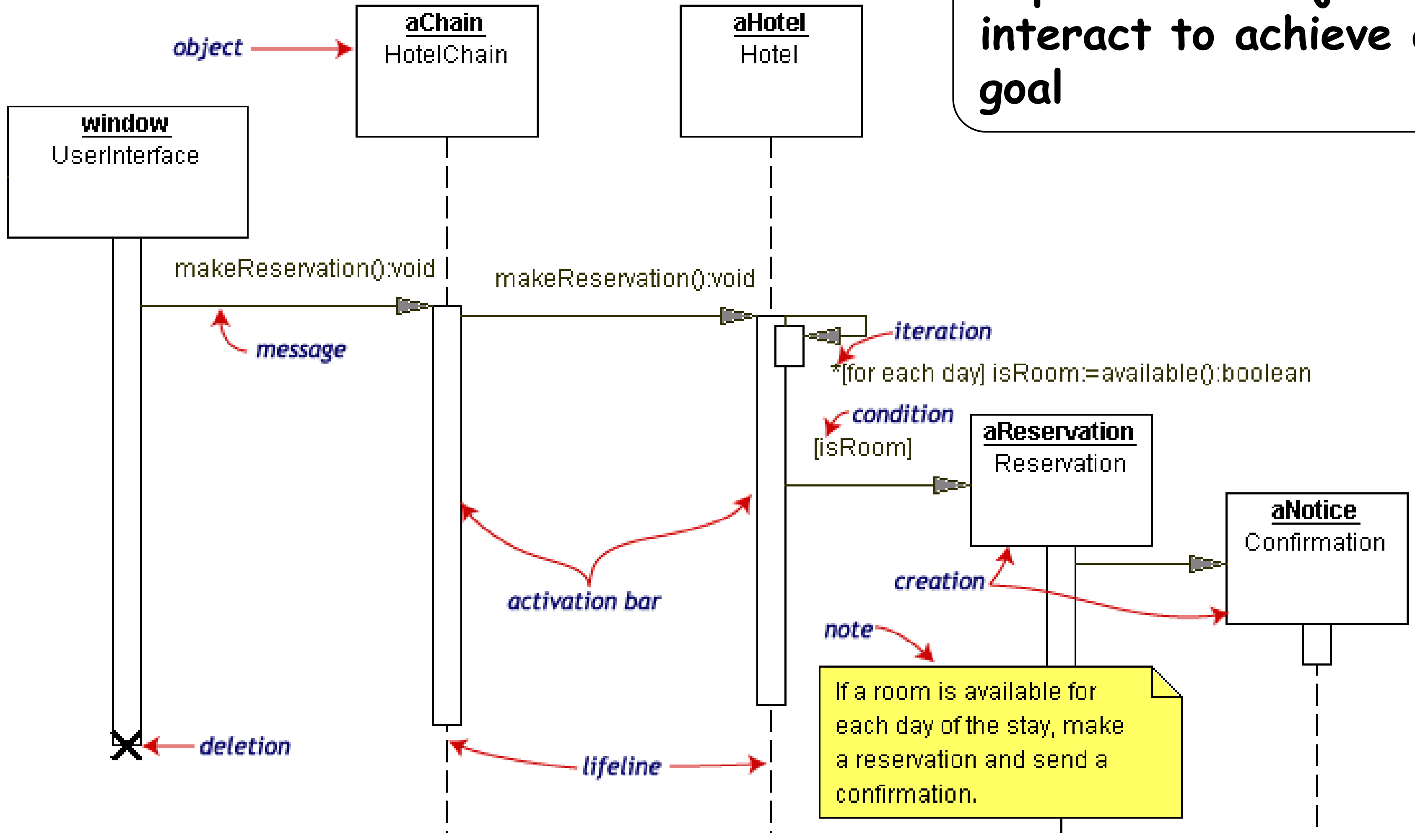
UML: Class Diagrams

Capture the static structure of systems associations between classes



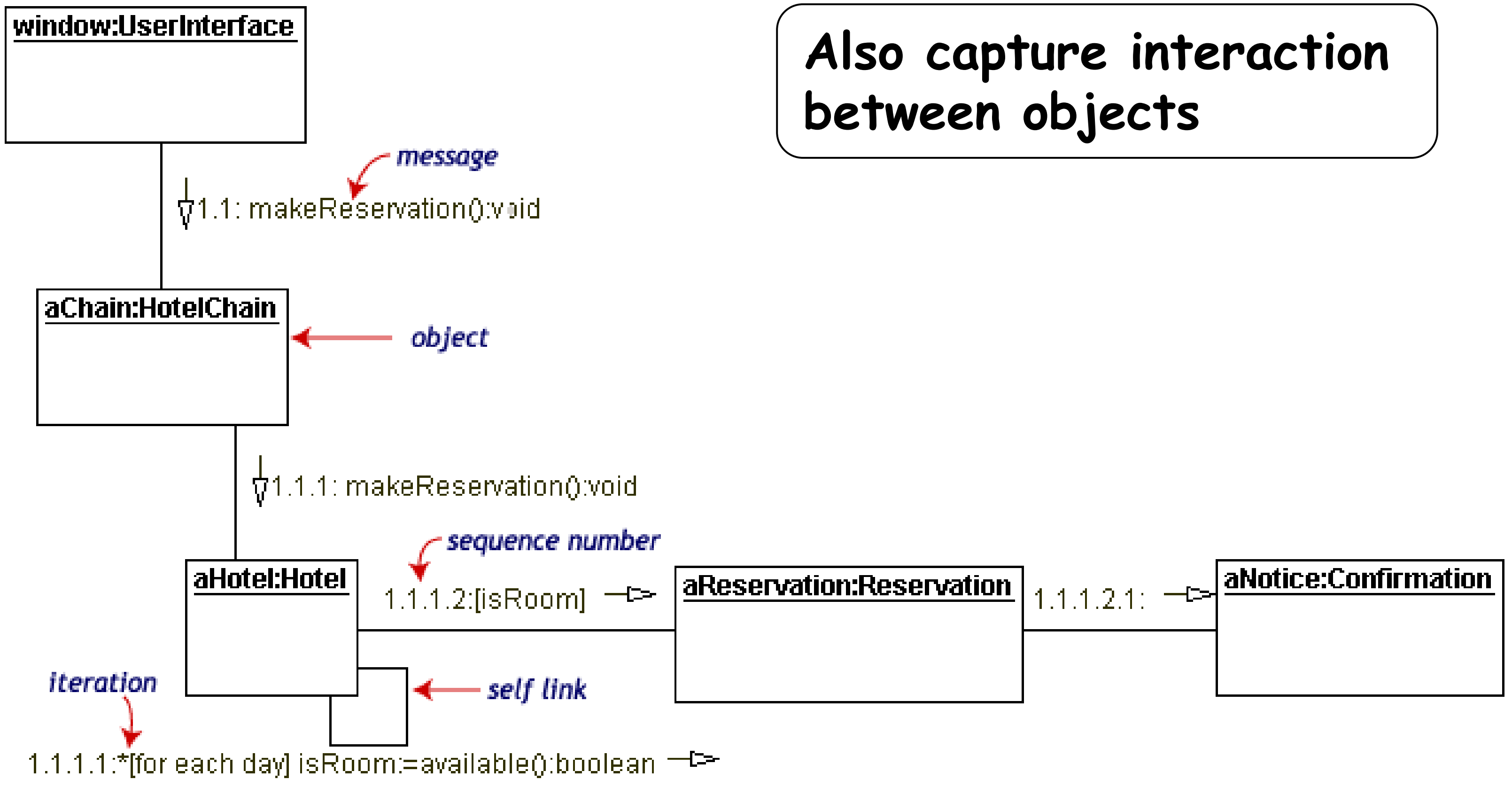
UML: Sequence Diagrams

Capture how objects interact to achieve a goal



If a room is available for each day of the stay, make a reservation and send a confirmation.

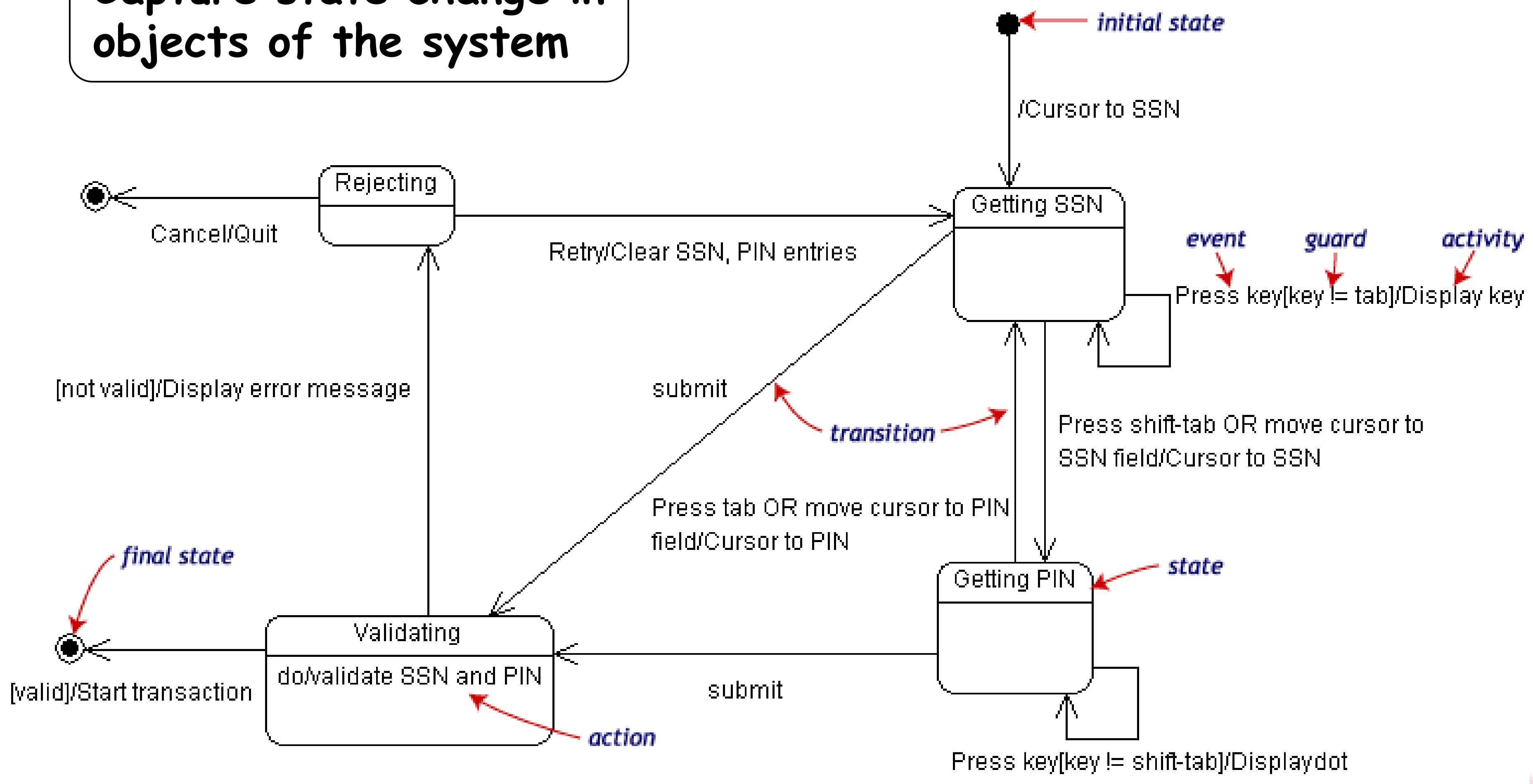
UML: Collaboration Diagrams



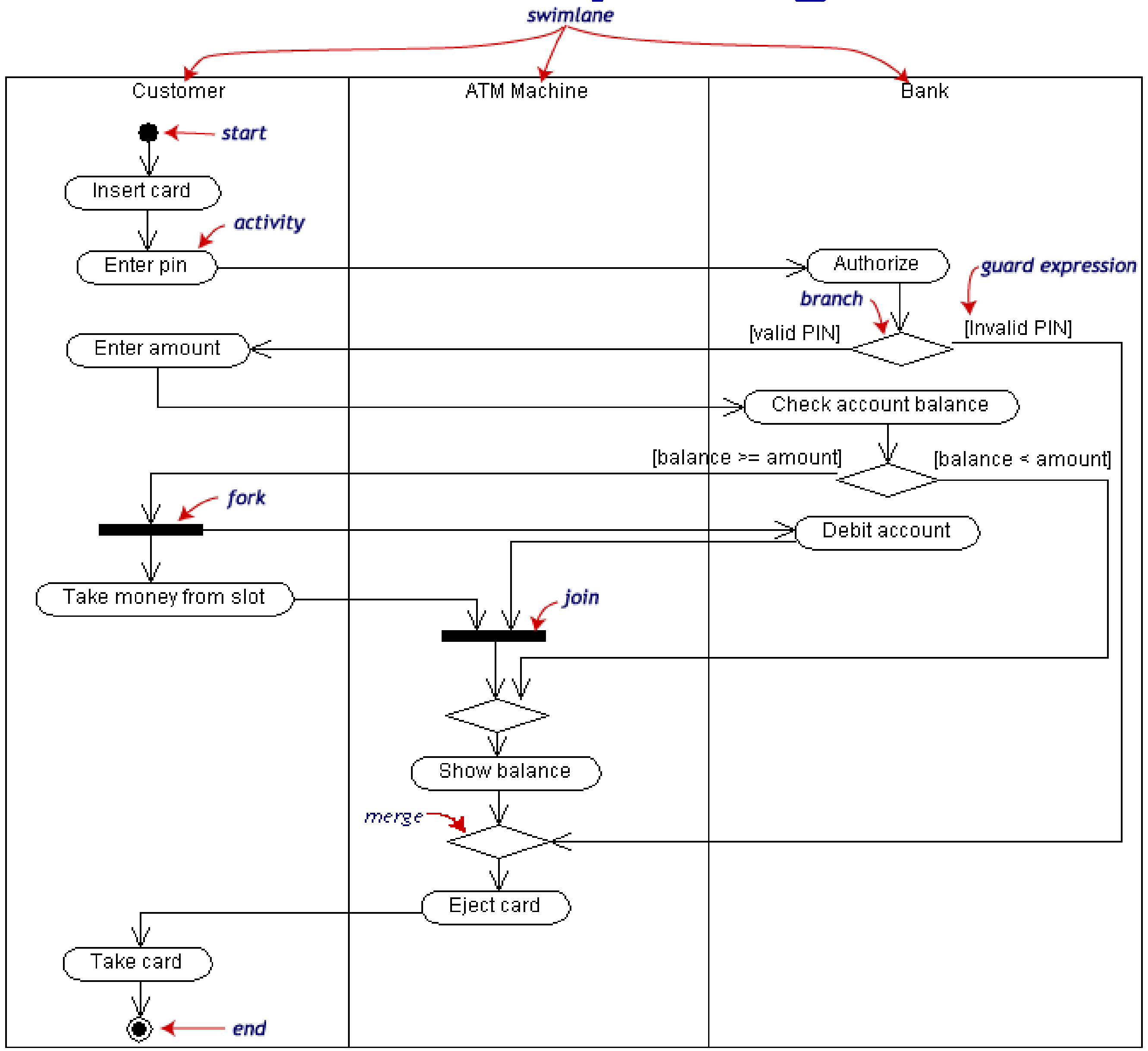
Also capture interaction between objects

UML: Statechart Diagrams

Capture state change in objects of the system

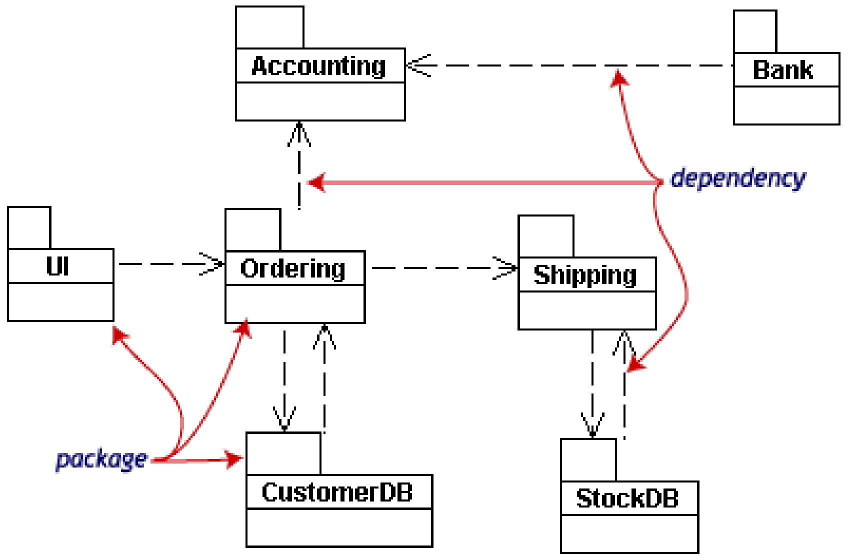

initial state
/Cursor to SSN
Rejecting
Getting SSN
Cancel/Quit
Retry/Clear SSN, PIN entries
event
guard
activity
Press key[key != tab]/Display key
[not valid]/Display error message
submit
transition
Press shift-tab OR move cursor to SSN field/Cursor to SSN
Press tab OR move cursor to PIN field/Cursor to PIN
final state
Getting PIN
state
Validating
do/validate SSN and PIN
[valid]/Start transaction
submit
action
Press key[key != shift-tab]/Displaydot

UML: Activity Diagrams

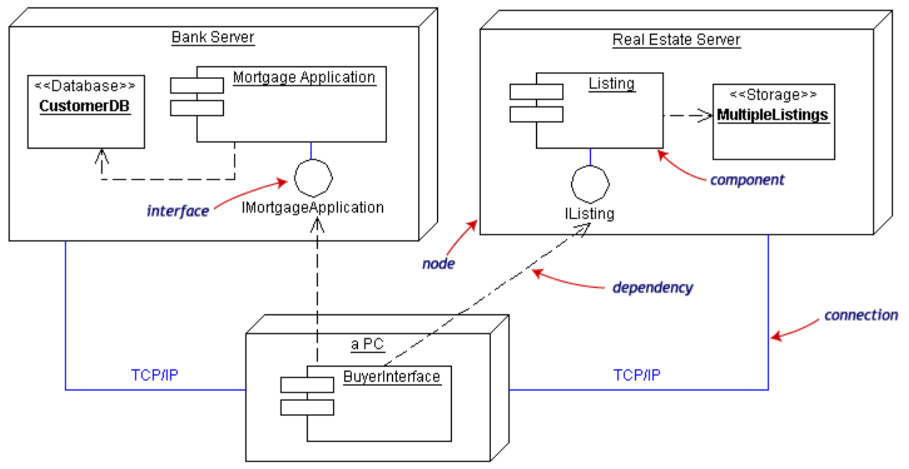


Capture the workflow in a situation

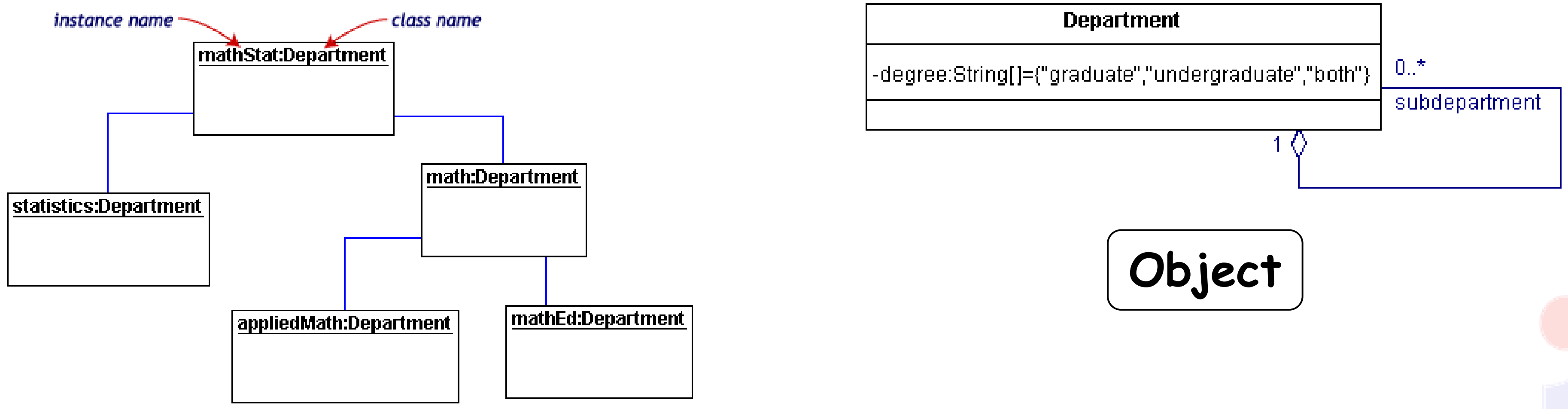
UML: Other Diagrams



Package



Component and Deployment



Object

Things to Do

- Read the other software engineering case studies.

  [SEOC1 Resource webpage]

- Read the *Practical UML: A Hands-On Introduction for Developers*. The Borland intro to UML is a quick intro to UML.

  [SEOC1 Resource webpage]

- Buy the main course book:
  - UML, Schaum's Outline Series, Simon Bennett, John Skelton and Ken Lunn, McGraw-Hill, 2001, ISBN 0-07-709673-8.
- Read chapters 1 and 2 of the UML book
- Further Readings:
  - Bertrand Meyer. Software Engineering in the Academy. In IEEE Computer, May 2001, pp. 28-35.
  - A. Avizienis, J.-C. Laprie and B. Randell, Fundamental Concepts of Dependability. UCLA CSD Report no. 010028, LAAS Report no. 01-145, Newcastle University Report no. CS-TR-739.

  [SEOC1 Resource webpage]