Probeklausur 1 Januar 2016, Fragen und Antworten

Einführung in die Softwaretechnik (IN0006) (Technische Universität München)

Questions :

1. General Knowledge

What major goal does the technique of decomposition pursue and why?

The main goal of decomposition is to deal with complexity. Since most real world problems are difficult to deal with as a whole, decomposition is used to make it easier to deal and implement the model of this problem. After that, since you broke up the problem into easier and smaller subproblems maintenance has gotten easier too. Detecting and fixing bugs is a trivial task once the size of the systems components are small enough. Decomposition adheres the divide and conquer principle (current buzzwords for further reading: micro- or nanoservices).

Explain the difference between a composition and an aggregation and show how they are represented in a UML diagram.

Composition is a special form of aggregation where the lifetime of the components is controlled by the aggregate. Composition is a filled out diamond shape, where as aggregation is a blank diamond shape in UML.

What are the two major components of an activity diagram? Please explain shortly their purpose.
The Node describes an activity or an object and a Edge is a directed connection between nodes

What sort of UML-diagram do you use, when you want to model the flow and interactions between objects?
Sequence diagram

Explain the difference between a Class and an Object.

Object is instance of Class. Class is "blueprint" for model

How does Abbot's technique in text to model transformation work?
List at least 2 mappings.
Noun to object/class, verb to operation, adjective to attribute

Requirement engineering is a process that is done to raise the quality of the final product. What is URPS standing for?
URPS: Usability, Reliability, Performance, Supportability

Name the two types of concurrency in a system and explain them briefly.
- Physical concurrency: Threads are provided by the hardware (multiprocessors, multi-cores, computer networks)
- Logical concurrency: Threads are provided by the software (usually provided by threads packages)

What is Strict Inheritance?
Strict Inheritance means that methods of the superclass cannot be overridden by subclasses

Which kind of pattern is the bridge-, the observer- and the abstract factory pattern?

Structural, behavioral, and creational pattern.

2. System Model

List the models that the system model is composed of. For each of the models also state its purpose in one sentence or question.

- **Functional Model (Use Case Model)**
  - What are the functions of the system?
  - What does the system do?
- **Static Model (Object Model)**
  - What is the structure of the system?
- **Dynamic Model**
  - How does the system react to (external) events?
  - What is the behaviour of the system?

3. Organization forms

List and shortly explain three forms an organization can have which were discussed in the lecture (in one sentence, no drawings are required).

- **Functional organization:**
  - In a functional organization people are grouped into departments, each addressing an activity ("function")
- **Project-based organization**
  - In a project-based organization people are assigned a project, addressing a problem to be solved within a specified time and budget
- **Matrix organization**
  - In a matrix organization, people from different departments of a functional organization are assigned to work on one or more projects

4. Requirements Review

a) *In the context of a requirements review, explain the following terms in one sentence each:*

correct, complete, consistent, unambiguous, realistic

- *Is the model **correct**? The model represents the client's view of the system*
- *Is the model **complete**? Every scenario is described in the model*
- *Is the model **consistent**? The model does not have components that contradict each other*
- *Is the model **unambiguous**? The model describes one system, not many*
- *Is the model **realistic**? The model can be implemented*

b) *Also shortly explain the terms* verification *and* validation *and how these concepts relate to the term* correctness.

- ***Verification** is an equivalence check between two models, one of them generated from the other one*
- ***Validation** is the comparison of the model with reality (with the client)*
- *Validation shows the correctness of a model. Correctness cannot be shown by means of verification, as there is no formalized model of the client's view of the system.*

5. System Design

List the eight issues of system design and explain four of them by naming at least two aspects relevant to each.

- *Design Goals*
  - *Additional NFRs*
  - *Trade-Offs*
- *Subsystem Decomposition*
  - *(Layers vs Partitions)*
  - *Architectural Style*
  - *Coherence & Coupling*
- *Concurrency*
  - *Identification of parallelism (processes, threads)*
  - *Physical vs. logical concurrency*
- *Hardware/Software Mapping*
  - *Identification of Nodes*
  - *Buy vs. Build*
  - *Network Connectivity*
  - *(Special Purpose Systems)*
- *Persistent Data Management*
  - *Storing persistent objects*
  - *Filesystem vs.Database*
- *Global Resource Handling*
  - *Access Control*
  - *ACL vs. Capabilities*
  - *Security*
- *Software Control*
  - *Monolithic*
  - *Event Driven*
  - *Concurrent Processes*
- *Boundary Conditions*
  - *Initialization*
  - *Termination*
  - *Failure*

6. Testing

Shortly explain the terms black-box and white-box testing. For each type of testing also explain two related problems (one sentence per problem).
Solution:

- Black-box testing
  - *Definition:* Tests the input/output behavior without considering implementation
  - *Problems*:
    - Potential combinatorial explosion of test cases (valid & invalid data)
    - Does not discover extraneous use cases ("features")
    - Erroneous behaviour may still deliver correct results by chance
- White-box testing
  - *Definition:* Tests the internal logic of the subsystem or class
  - *Problems:*
    - Potentially infinite number of paths have to be tested
    - White-box testing often tests what is done, instead of what should be done
    - Cannot detect missing use cases
    - Experts for code needed
    - Time/Cost intensive

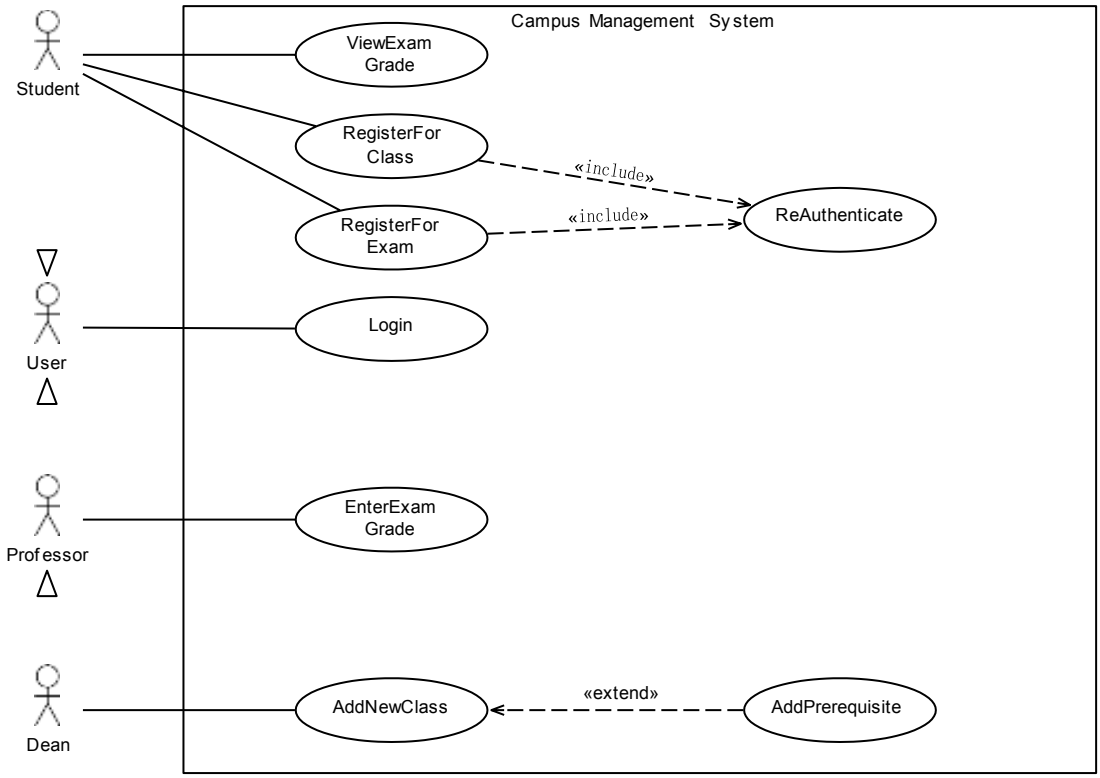
7. Modeling a Campus Management System

The university UVN is planning a new campus management system (UVNOnline). The following text describes the desired functionality:

> The system provides a login mechanism for all users that requires a username and a password. Students shall be able to register for classes and for exams. In order to register for either, the student has to re-authenticate herself by only entering her password. Professors shall be able to enter exam grades into the system, which the students can then view. The dean, who is also a professor, shall be able to add new classes into the system. While adding new classes, she may additionally specify prerequisites if applicable.

c) *Model the functionality of the campus management system described above as a UML diagram.*

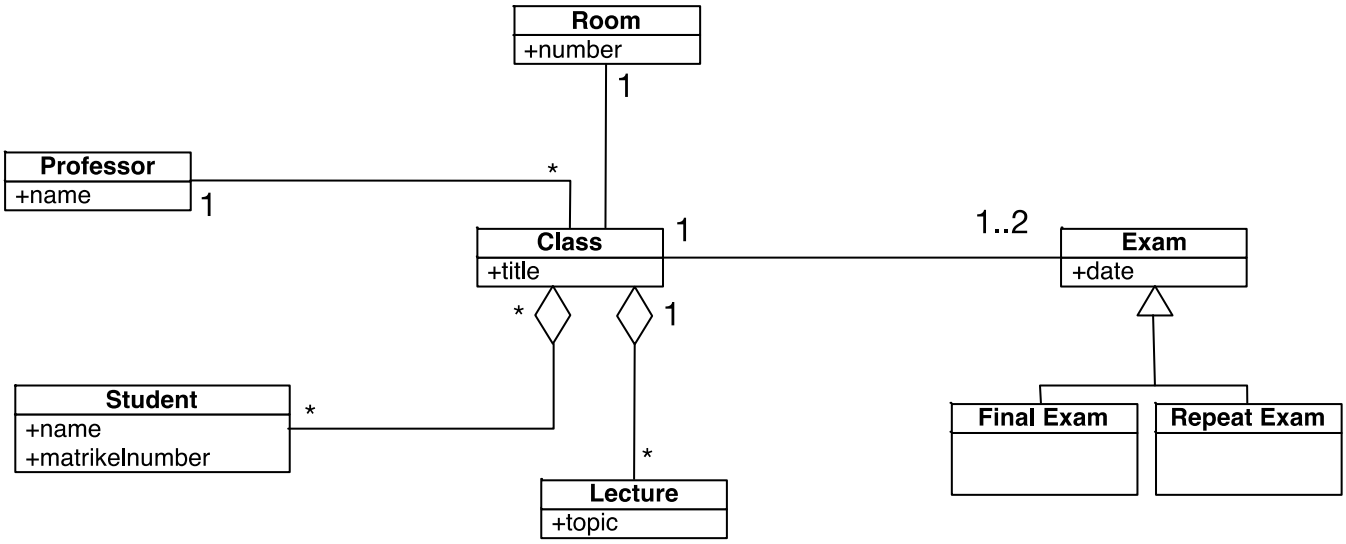
Solution:



Each class in the university is held by a single professor in a single room for many students. Rooms are identified by their number and are used for multiple classes. A professor has a name and teaches many classes. A Student has a name and a matrikelnumber and can take part in many classes. Each class has a title and consists of many lectures, each with a different topic. At the end of the semester, a class has one or two exams, each on a certain date. Exams can either be final exams or repeat exams.

d) *Use Abbot's technique on the text above to create a static analysis model in UML (including attributes and multiplicities).*

Solution:



8. Hardware/Software Mapping

The car manufacturer ALV is making plans for a new production management system with two main components, an order component and a production component. Clients should be able to access the order component via a web browser on their PC or with a native client on their mobile device. For complexity reasons, the web interface should not directly be integrated into the order component. Both the order component and the production component are backed by separate stand-alone database servers. The production component accesses the order component to receive new orders, and provides this information to an assembly line controller that is located on a production machine.

Take the above description and map the necessary components to as few nodes as possible in the form of a deployment diagram.

Solution: