KTH Microelectronics
and Information Technology

Exam in ID2207 Modern Methods in Software Engineering, 2013-10-31, 09:00-13:00

Rules
This exam is "closed book" and you are not allowed to bring any material or equipment (such as laptops, PDAs, or mobile phones) with you. The only exceptions are English to "your favorite language" dictionary and pencils.

Instructions
- Please read the entire exam first!
- Write clearly
- Each sheet of paper must contain your name, "personnummer", Problem number and a unique sheet number
- Write only on one page of a sheet. Do not use the back side
- Sort your sheets according to the problem's numbering
- Only one Problem must be reported on each sheet
- If more than one sheet is needed the continuation should be clearly noted on the beginning of each sheet and the sheet numbers used should be consecutive
- Always motivate your answers. Lack of clearly stated motivation can lead to a reduction in the number of points given
- The tasks are not necessarily sorted in order of difficulty. If you get stuck it might be a good idea to go on to the next task.

Grading
If n is amount of your exam points and m is your bonus points earned in autumn 2013 then:

$n+m < 50$ fail (F)
$50 \leq n+m < 60$ grade E
$60 \leq n+m < 70$ grade D
$70 \leq n+m < 80$ grade C
$80 \leq n+m < 90$ grade B
$90 \leq n+m$ grade A

GOOD LUCK!

Mihhail Matskin, mobile 0704614269

Problem I. General questions

a) What are basic ways to deal with complexity?

Abstraction: ignore non-essential details, modeling
Decomposition: break problem into sub.problems
Hierarchy: simple relationship between chunks

(4p)

b) Which of the following are models?

Yes - a UML class diagram (need to motivate?)
Yes - a set of UML class diagrams describing the classes in a software system
Yes - a 1:100 scale clay replica of a new sports car that will be used to test its aerodynamics in a wind tunnel
No - a full-scale, working prototype of a new sports car

(4p)

c) Why do we apply principle of Falsification in software engineering?

- Because it is impossible to show that a model is correct or correctly represents reality.
- By falsifying we can remove certain obvious mistakes in order to make the model "more correct" or "less wrong". (5p)

Problem II. Software Life Cycle

a) Unified Software Development Process (UP) considers phases, iterations and workflows in software life cycles. Briefly explain each of them and how they are related. Use figures where appropriate.

Cycles consist of the four Phases: Inception (establish business case), Elaboration (specification of pruduct cases, design of architecture), Construction (product is built), Transition (period when product moves to beta phase). Each phase consists of a number of iterations. Each iteration is a set of use cases or considers certain risks. For each iteration, several workflows are performed parallelly (supporting workflows: Management, Environment, ... engineering workflows: requirements, design, ...) (5p)

b) Explain what is common and what is different between transformational and deductive synthesis.

Explain what is common and what is different between transformational synthesis and Model Driven Architecture (MDA)

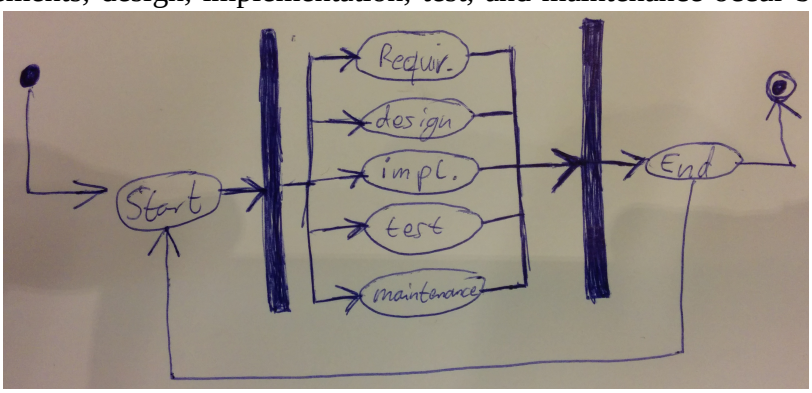
Both base on deduction but they use different deduction methods: Synthesis is based on inference (axioms and rules are expressed as implications) Transformation is based on replacement (axioms are expressed as equations or rewrite rules). MDA: guidelines expressed as models for structuring of specifications, used by imposing a series of transformations in MDA, you ick a platform independent model (like pseudocode) and then port it to the specific platforms. (both use transformations) (5p)

Problem III. UML and OOP

a) What is/are difference(s) between Sequence diagrams and Communication diagrams? When each of them is more appropriate to use?

Both depict the same information; CD represents the sequence by numbers, SD by "timeline". CD is more compact but might become hard to read when it gets complex. SD can become very large but always stays easy to follow.
Use SD when it should be easy to follow and space consumtion is not important. Use CD vice versa. (4p)

b) Draw a UML activity diagram describing the dependency between activities for a life cycle in which requirements, design, implementation, test, and maintenance occur both concurrently and iteratively.



(5p)

c) Consider the object model in Figure 1. Given your knowledge of the Gregorian calendar, modify the model such that a developer unfamiliar with the Gregorian calendar could deduce the number of days in each month, the number of weeks in each month, the number of months in a year and the number of days in a week? Identify additional classes if necessary.

(6p)

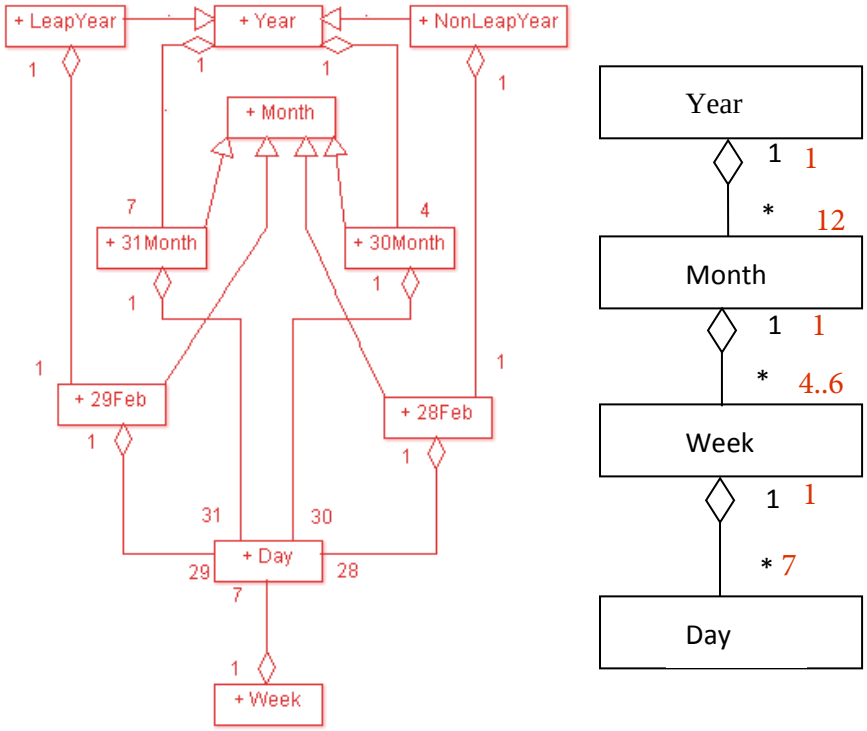


Figure 1

Problem IV. Requirements Elicitation

a) What are main types of requirements? Briefly explain each of them

1) Functional requirements: interaction between system and environment
2) Nonfunctional requirements: Usability, Reliability, Performance, Supportability,
(implementation related, user visible aspects (see homework))

(4p)

b) Specify which of these requirements are verifiable and which are not. Explain why.

- *"The system must be usable."*
  NO: Not verifiable, because not meisurable. What does "usable" mean?
- *"The system must provide visual feedback to the user within 1 second of issuing a command."*
  YES: Can be meisured. Whether the time of feedback is higher or lower than 1 second
- *"The availability of the system must be above 95%."*
  NO: We cannot predict the furute. We dont know if in 2 years the system might not work e.g. for 3 consecutive months.
- *"The user interface of the new system should be similar enough to the old system such that users familiar with the old system can be easily trained to use the new system."*
  NO: We cannot meisure "ease" or "similariry". We have to rely on user's feedback but cannot verify.

(4p)

Problem V. Requirements Analysis

a) There are different approaches to class identification. In practice, the process of class discovery is likely to be guided by different approaches at different times. Give a scenario of a mixed approach to class discovery involving at least 4 other approaches.

Discover initial set of classes (generic knowledge), common class pattern approach provides additional guidance, more classes from high-level-descriptions analysis, noun-phrase-approach provides more classes, use-case driven approach used to verify existing classes, CRC approach allows brainstorming of the discovered classes. (5p)

b) Give an example of a problem where Aspect-Oriented Programming is useful to apply. Your example must be different from the examples presented in the Lecture Notes of the course.

Some problems are not attached to any particular domain but they are spread across them

(5p)

Example[illegible]

Problem VI. System Design

a) Assume that we classified design goals into five categories: performance, dependability, cost, maintenance, and end user. Assign one or more categories to each of the following goals: (need to motivate?)

WHAT ABOUT COST?!?

1. Users must be given a feedback within 1 second after they issue any command. performance, end-user
2. The TicketDistributor must be able to issue train tickets, even in the event of a network failure. dependability, end-user
3. The housing of the TicketDistributor must allow for new buttons to be installed in the event the number of different fares increases. maintenance
4. The AutomatedTellerMachine must withstand dictionary attacks (i.e., users attempting to discover an identification number by systematic trial). dependability
5. The user interface of the system should prevent users from issuing commands in the wrong order. end-user, dependability

(4p)

b) What are benefits of layered architecture? What are main problems with this architecture?

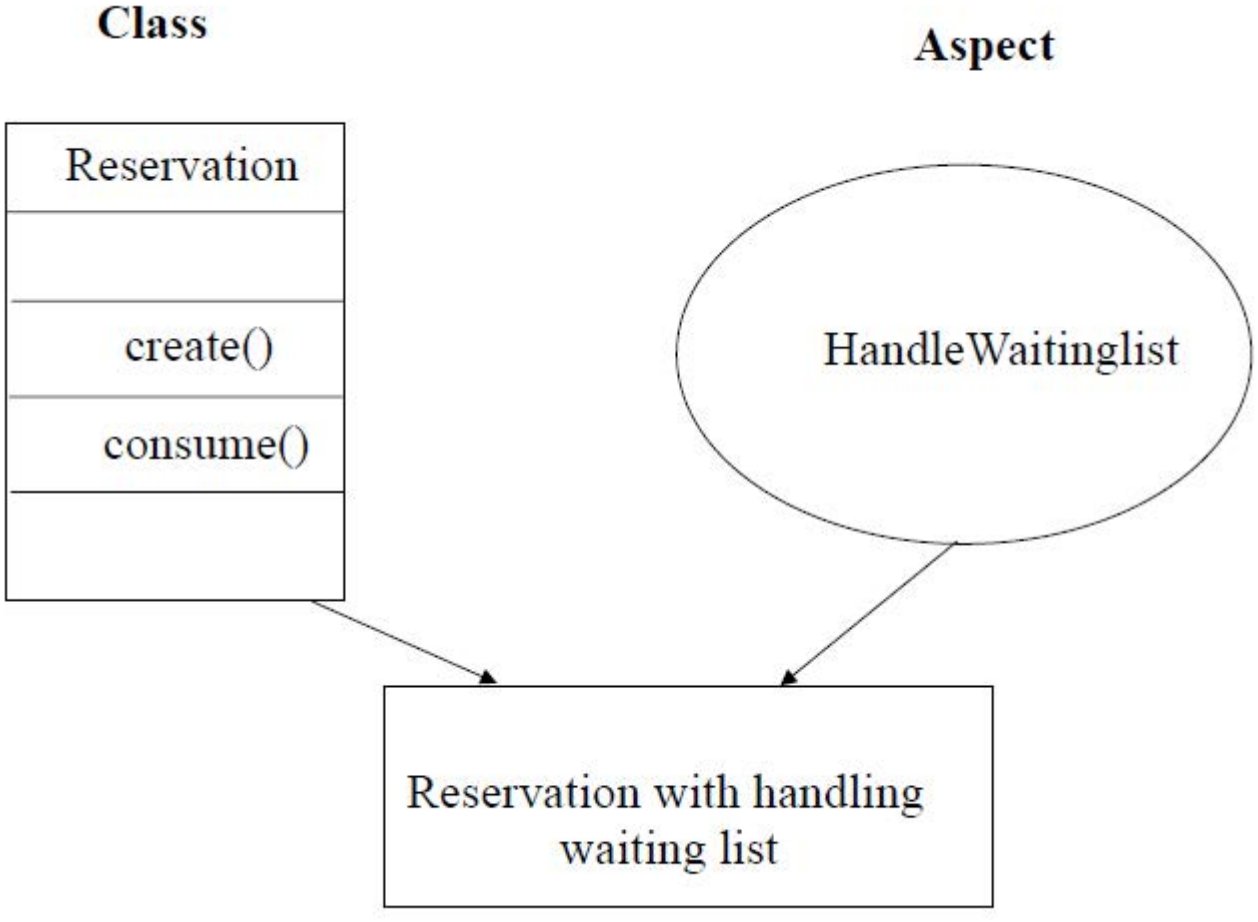
they are hierarchical --> reduced complexity; independent layers (decoupling) enable easier change of implementation (because of independance); a very effective abstaction; entire system might become inefficient --> slide 05 page 26 (open/closed?) (4p)

Can be both (transparent and non-transparent layers: drawback (see slides) + overhead (storage + performance)

c) Older compilers were designed according to a pipe and filter architecture, in which each stage would transform its input into an intermediate representation passed to the next stage. Modern development environments, including compilers integrated into interactive development environments with syntactical text editors and source-level debuggers, use repository architecture. Identify the design goals that may have triggered the shift from pipe and filter to repository architecture.

More convenient to use, higher flexibility, loose coupling,

(5p)


Class
Aspect
Reservation
create()
consume()
HandleWaitinglist
Reservation with handling waiting list

Problem VII. Object Design - Reuse

a) Consider an application that must select dynamically an encryption algorithm based on security requirements and computing time constraints. Which design pattern would you select? Draw a UML class diagram depicting the classes in the pattern and justify your choice.

Strategy--> see sketch

(5p)

b) Explain how delegation, implementation and specification inheritance are shown in UML class diagrams.

delegation (line), specification (arrow w/ closed arrowhead), implementation (also arrow like inheritance)

(4p)

Problem VIII. Object Design – Interface design

a) Consider a class diagram in the Figure 2.

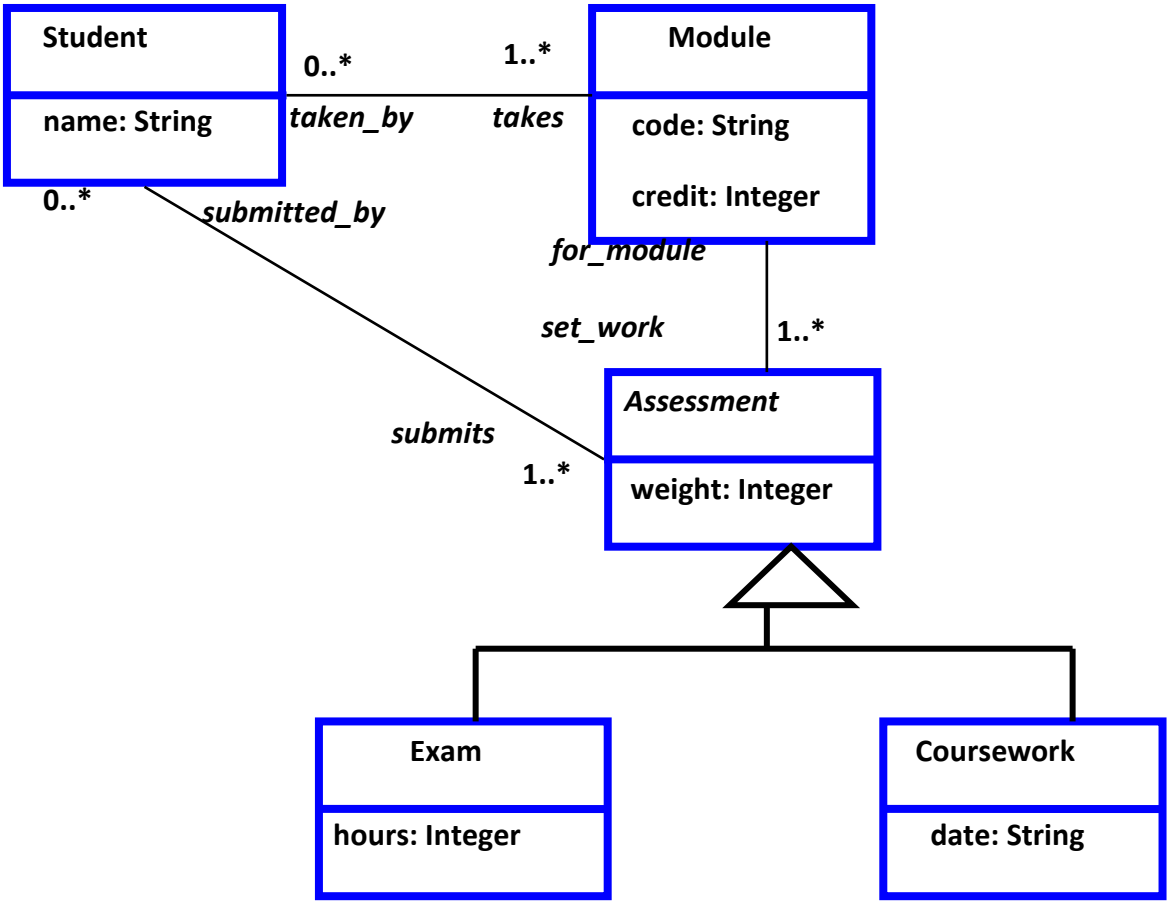


Figure 2.

Your task is to define in OCL the following constraints:

a) Modules can be taken if they have more than seven students registered

b) The assessments for a module must total 100%

c) Students must register for 120 credits each year

(6p)

Problem IX. Moving to Code

a) Explain the following mapping concepts: model transformation, forward engineering, reverse engineering and refactoring. Draw a figure showing their relations to model space and source code space.

(4p)

b) Explain realization of a unidirectional, one-to-one association in source code.

(4p)

Problem X. Testing.

a) Explain the following testing concepts: Reliability, Fault, Erroneous state, Failure.

(4p)

Problem XI. Agile Software Development.

a) What are main ideas of the Manifesto of Agile Software Development?

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over construnct negotiation
- responding to change over following a plan

(4p)

----------End of Exam----------