Exam 2021

Chapters in the text-book that are less relevant to the exam

- Chapter 3 – Project organization and Communication
- Chapter 12 – Rational Management
- Chapter 13 – Configuration Management
- Chapter 14 – Project Management

Grading

1. In order to pass exam (grade E) you have to get at least 38 points from questions from Part I (this includes your bonus points).
2. In order to get grade D you have to pass exam (see item 1) and get 49 points from all questions (this includes your bonus points).
3. In order to get grade C you have to pass exam (see item 1) and get 63 points from all questions (this includes your bonus points).
4. In order to get grade B you have to pass exam (see item 1) and get 74 points from all questions (this includes your bonus points).
5. In order to get grade A you have to pass exam (see item 1) and get 85 points from all questions (this includes your bonus points).

Part I (48p).

<u>General questions</u>

1. What is the purpose of modelling?

(2p.)

2. In the following description, explain when the term account is used as an application domain concept and when as a solution domain concept:

"Assume you are developing an online system for managing bank accounts for mobile customers. A major design issue is how to provide access to the accounts when the customer cannot establish an online connection. One proposal is that accounts are made available on the mobile computer, even if the server is not up. In this case, the accounts show the amounts from the last connected session."

(3p.)

Part I

Software Life-cycle

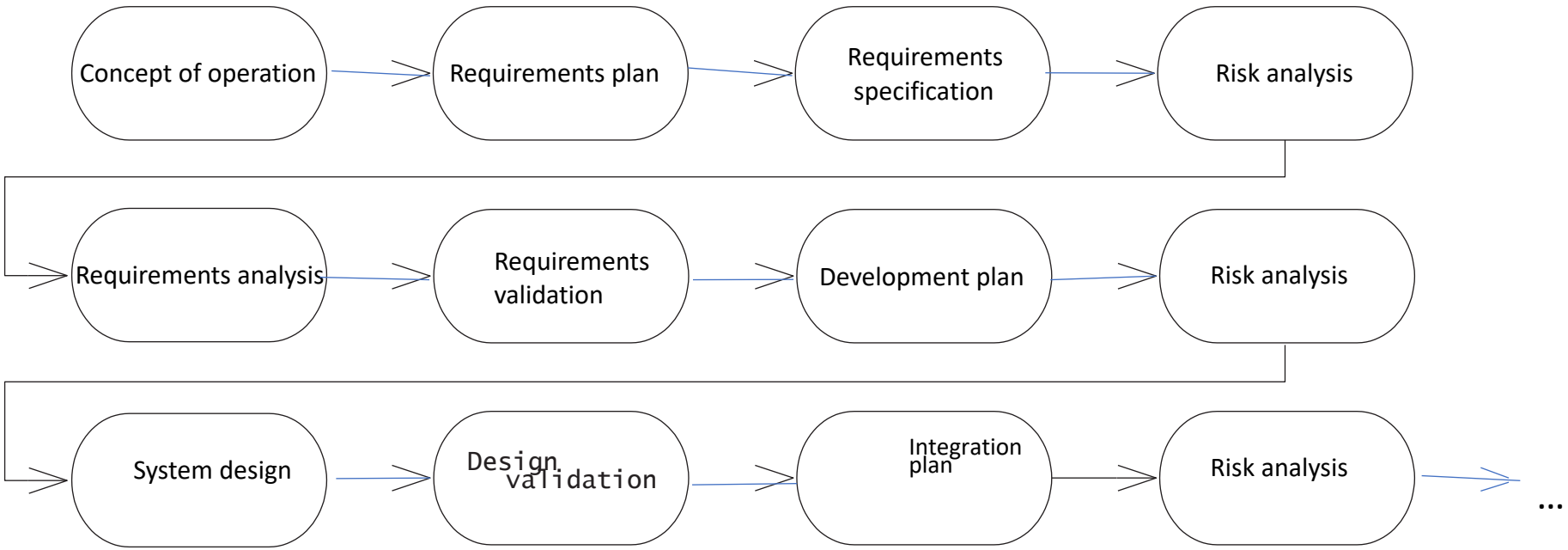
3. Redraw Boehm's spiral model as a UML activity diagram. Compare the readability of the original figure with the activity diagram.

(5p.)

Modelling with UML

4. Consider an ATM (Automated Teller Machine - cash machine) system. Identify at least three different actors that interact with this system and at least one that can interact with system but shouldn't.

(3p.)


Concept of operation
Requirements plan
Requirements specification
Risk analysis
Requirements analysis
Requirements validation
Development plan
Risk analysis
System design
Design validation
Integration plan
Risk analysis
...

Part I

Requirements Elicitation

5. From your point of view, describe the strengths and weaknesses of users during the requirements elicitation activity. Describe also the strengths and weaknesses of developers during the requirements elicitation activity. (4p.)

System Analysis

6. What are inputs and outputs of the System Design activity? (4p.)

7. Which approaches to class identification do you know? Briefly explain them. (4p.)

Part I

System design

8. In many architectures, such as the three- and four-tier architectures, the storage of persistent objects is handled by a dedicated layer. In your opinion, which design goals have led to this decision?

(4p)

9. Decomposing a system into subsystems reduces the complexity developers have to deal with by simplifying the parts and increasing their cohesion. If cohesion is the guiding principle driving developers to decompose a system into small parts, which competing principle drives them to keep the total number of parts small?

(3p.)

Part I

Object Design

10. Explain difference between usage of inheritance during System Analysis and Object Design activities. Give examples.

(5p.)

11. Which types of visibility are defined during object design? Explain why do we have different visibility?

Part I

Testing

12. Which types of unit testing you know? Explain difference between them.

(4p)

Agile methods

13. What is a meaning of Spike in eXtreme Programming?

(4p.)

Part II

14. What is meant by “knowledge acquisition is nonlinear”? Provide a concrete example of knowledge acquisition that illustrates this. (5p)

15 For each of the following documents, indicate in which phase(s) of the software life-cycle it is produced:

- *final user manual, -architectural design,*

-module specification, -source code,

-test plan, -preliminary user manual, - detailed design,

- project plan, - cost estimate,

- test report, -documentation (6p.)

- *final user manual, - implementation*
- *architectural design, - system design*
- *module specification, - system/object design*
- *source code, - implementation*
- *test plan, - requirements*
- *preliminary user manual, - requirements*
- *detailed design, - object design*
- *project plan, - project planning*
- *cost estimate, - project planning*
- *test report, - testing*
- *documentation.- implementation*

Part II

17. Explain realization of a bidirectional, many-to-many association in source code. (6p)

18. Discuss how the MVC (Model-View-Controller) architecture helps or hurts the following design goals:

- Extensibility (e.g., the addition of new types of views).
- Response time (e.g., the time between a user input and the time all views have been updated)
- Modifiability (e.g., the addition of new attributes in the model)
- Access control (i.e., the ability to ensure that only legitimate users can access specific parts of the model)

(5p.)

Bidirectional, many-to-many association

Object design model before transformation

Source code after transformation

```
public class Tournament {
   private List players;
   public Tournament() {
        players = new ArrayList();
   }
   public void addPlayer(Player p) {
        if (!players.contains(p)) {
              players.add(p);
              p.addTournament(this);
        }
   }
}
```

```
public class Player {
   private List tournaments;
   public Player() {
        tournaments = new ArrayList();
   }
   public void addTournament(Tournament t)
   {
        if (!tournaments.contains(t)) {
              tournaments.add(t);
              t.addPlayer(this);
        }
   }
}
```

Part II

18. Explain different ways of mapping inheritance to relational database. When each of them is more preferable?

(5p)

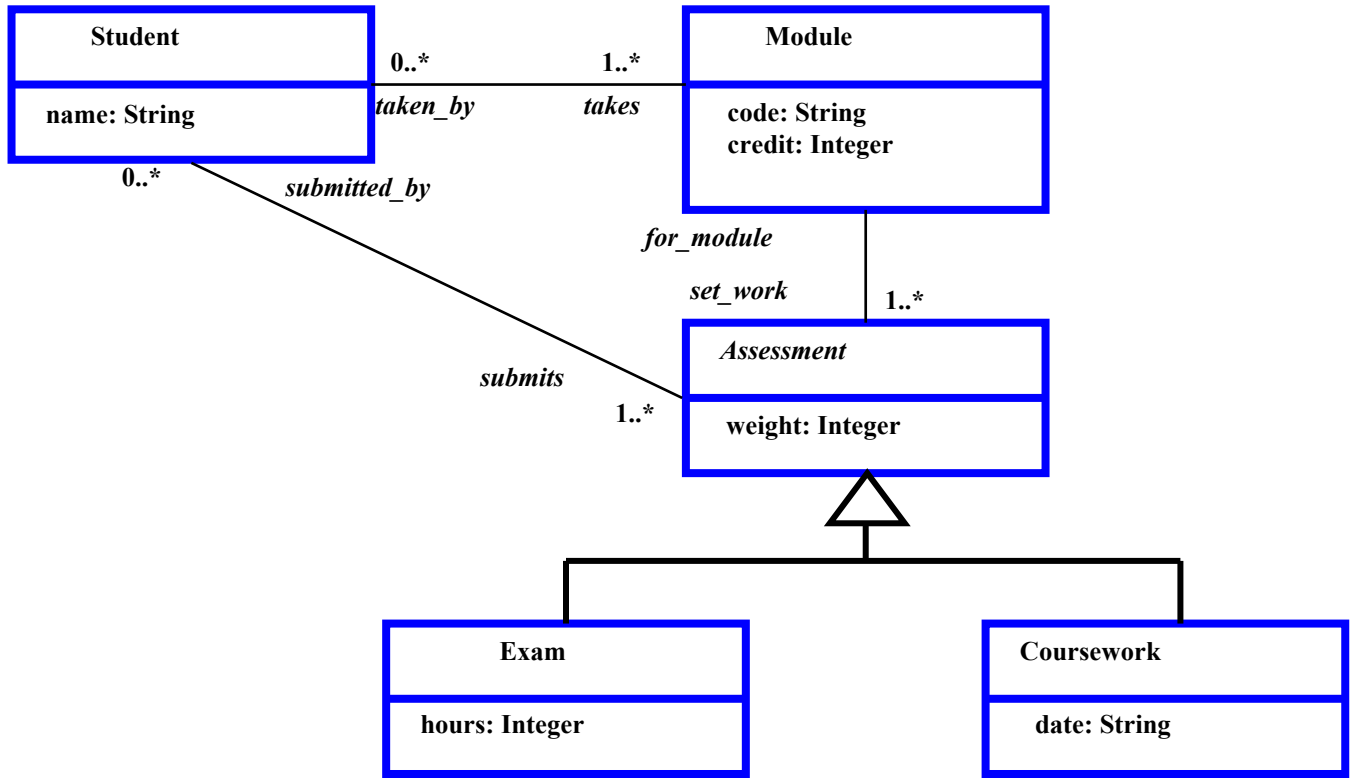
Part III

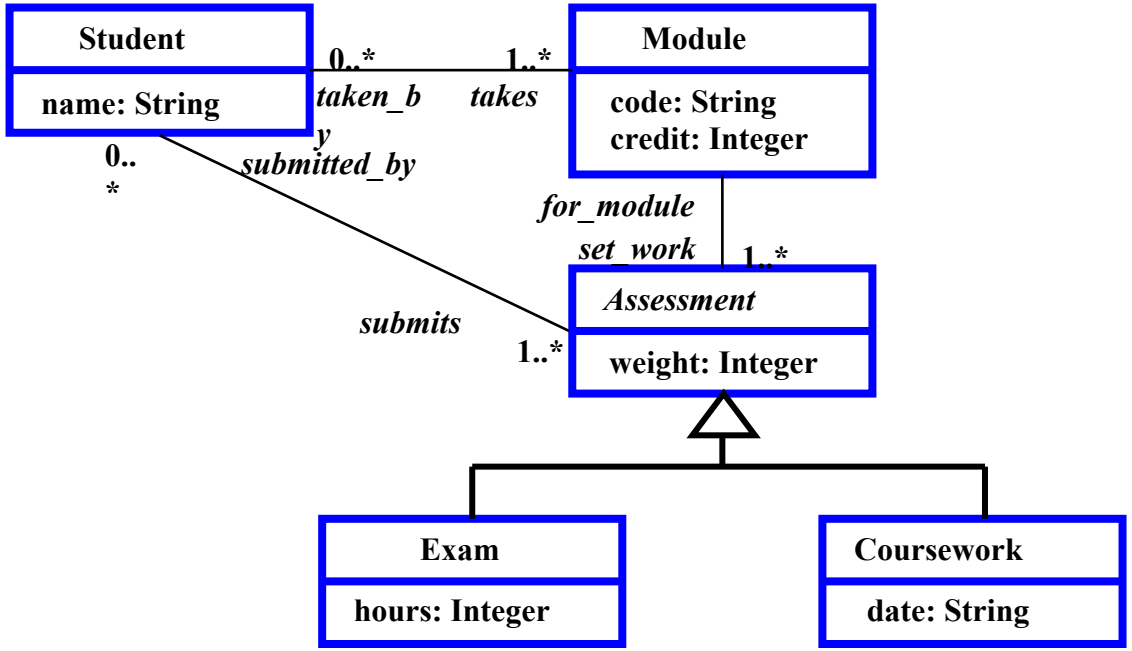
20. Having the class diagram in Figure 1, you task is define in OCL the following constraints:

a) Students must take at least 90 credits of CS modules each year

b) All modules must have at least one assessment worth over 50%

c) Students can only have assessments for modules which they are taking





- All modules must have at least one assessment worth over 50%

context *Module* **inv**: *set_work*→*exi*sts(s | s.*weight > 50*)

- Students can only have assessments for modules which they are taking

context *Student* **inv**: *takes*→ *forall (s* | *submits->include* **(***s))*

Students must take at least 90 credits of CS modules each year

context *Student* **inv**: *takes* →*select*(*code = 'CS'*).credit→sum( ) >= 90

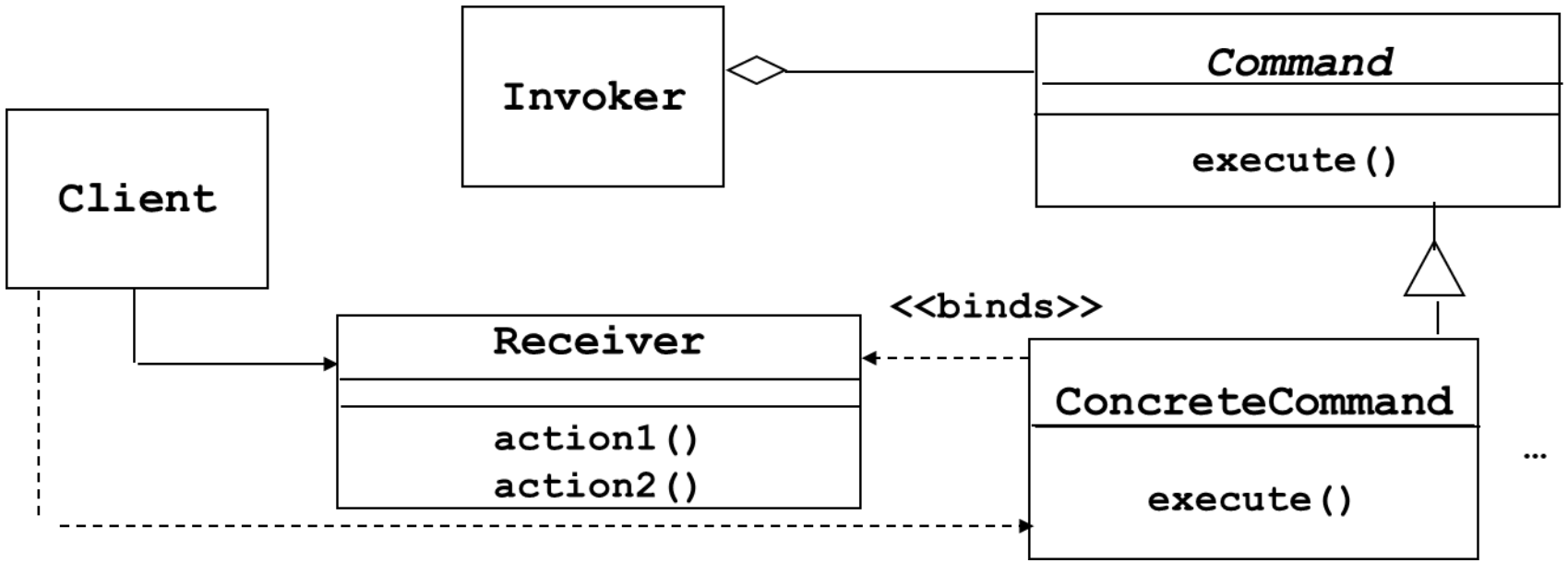
Part III

20. We considered 3 formal software development models: inductive, deductive and transformational. Briefly (without technical details) explain main features of each of these models.

(5p)

21. Explain the Command design patter. When is it applied? Illustrate your answer with a figure.

(7p.)



Client creates a **ConcreteCommand** and binds it with a **Receiver.**

Client hands the **ConcreteCommand** over to the **Invoker** which stores it.

The **Invoker** has the responsibility to do the command (“execute” or “undo”)**.**