2021-12-16-Software Systems - Design

Cursus: B-CS-MOD02-1B-202001024 B-CS Software Systems Core 202001024

Tijdsduur: 3 uur

Gegenereerd op: 20 dec. 2021

Inhoud:

Pagina's:

2021-12-16-Software Systems - Design

Cursus: B-CS Software Systems Core 202001024

Welcome to the Software Systems Design **digital exam**:

- This is an open book exam. You are allowed to use the slides. The slides are accessible on your Chromebooks under the whitelisted URL (http://grammarware.net/slides/2021/ss/) **Note:** A tab with the slides is already open by default.
- **Smartphones or personal notebooks are not allowed. Put those in your bag now (with the sound switched off)**
- For **technical questions** concerning the chromebooks, Remindo, log-in issues etc.: raise your hand.
- For **content questions**: use the BBB chat.
- Do not forget to **save** your answers once you are done.
- There are **12 questions** and **60 points**.

Good luck and enjoy the test!

Aantal vragen: 12

1 "All models are wrong" -- This sentence was used repeatedly throughout the course.

3 pt. Briefly explain the meaning of the statement in your own words.

2 Recall UML State Machine Diagrams.

Can you use them as (a.) descriptive, (b.) predictive, and (c.) prescriptive models?

For each of them give an answer and shortly motivate it.

2 pt. **a.** Descriptive:

2 pt. **b.** Predictive:

2 pt. **c.** Prescriptive:

3 Within the framework of Test Driven Development (TDD), what is the consequence of specifying too

2 pt. few test cases? Sketch at least two issues.

4 Given below are pairs of classes (eg. "House" and "Room").

For each pair:

1. define a sensible association between the classes (eg. "lives in" or "composition"),
2. specify the multiplicity/cardinality of the association if appropriate (eg. "1 - 1" or "1..* - 0.. 5").

2 pt. **a.** "Laptop"

"Device"

2 pt. **b.** "House"

"Landlord"

2 pt. **c.** "Java Program"

"Class"

2 pt. **d.** "Key"

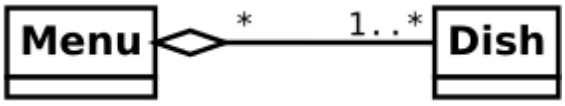
"Keyboard"

2 pt. **e.** "Woman"

"Human"

5 Consider the following class diagram:

4 pt.

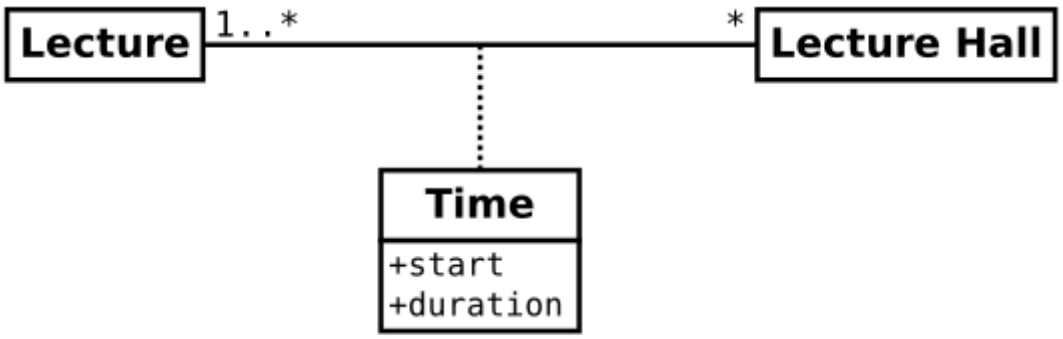


Specify two meaningful attributes for each class. Include visibility and type.

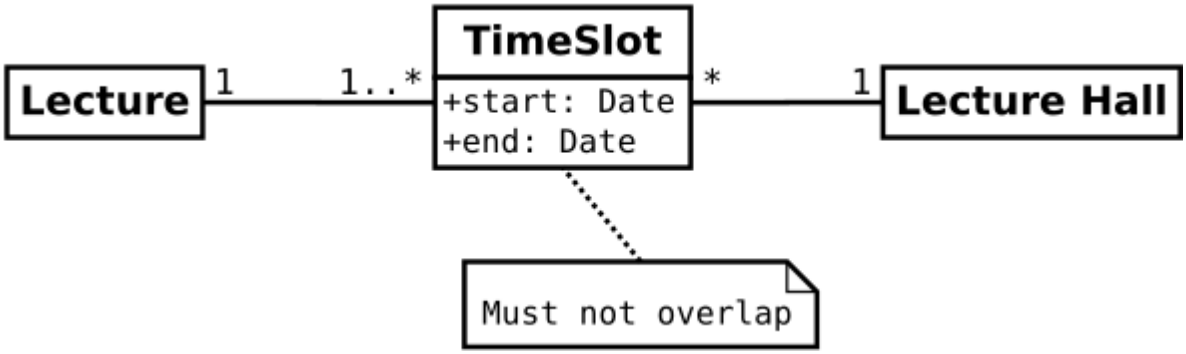
6 Consider the following two class diagrams, that model the schedule of lectures:

6 pt.

Class Diagram 1:



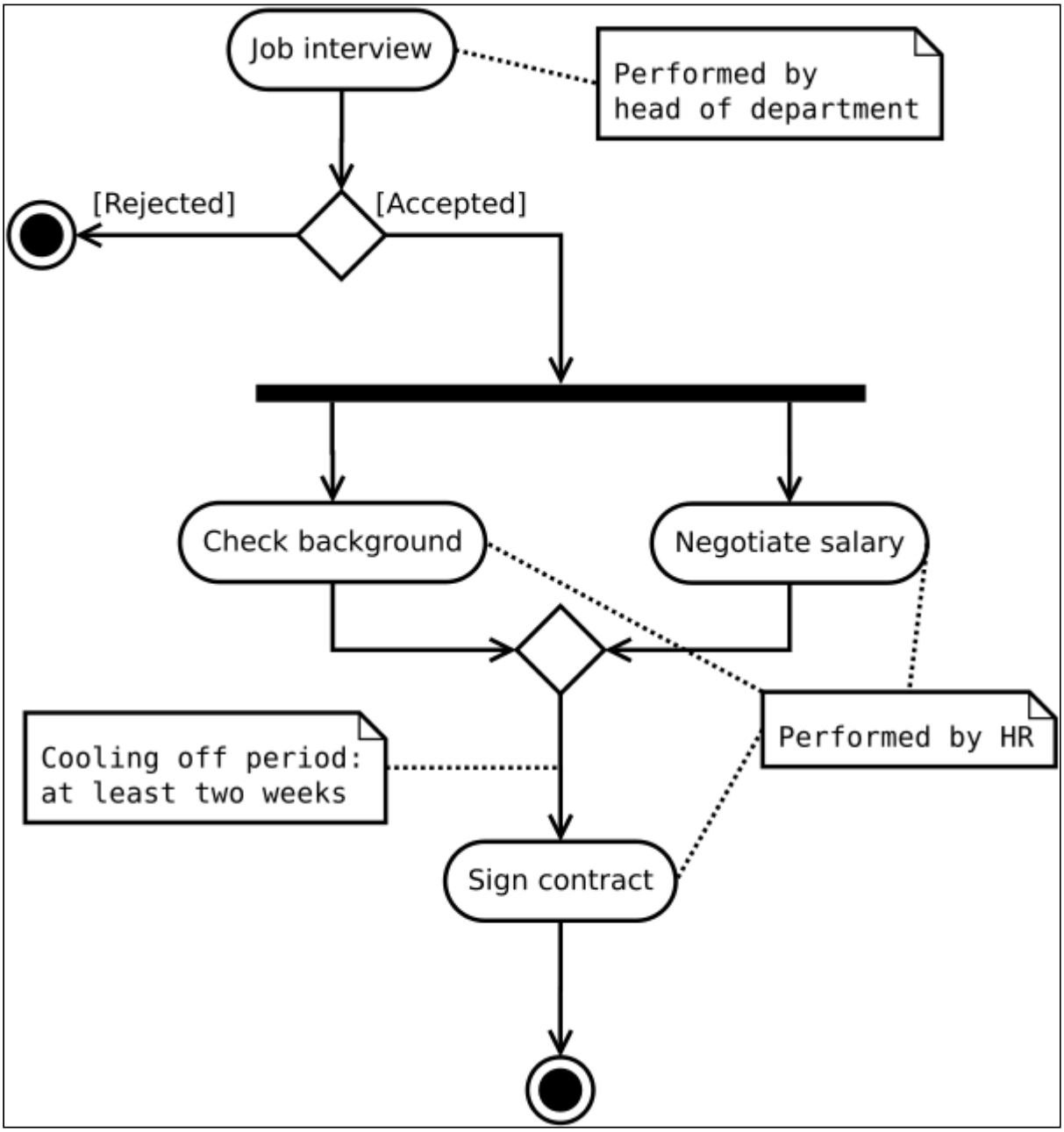
Class Diagram 2:



Compare the two diagrams. Point out at least three advantages/disadvantages of the models.

7 Given is the following Activity Diagram:

4 pt.



Name two properties of the diagram you would improve, and shortly explain how you would improve them.

8 A developer in your project group has the habit of committing once per day, no matter the state of their local copy.

2 pt.

As commit message they always use "today's work :)".

Why is this a bad idea?

9 Are the following good Software Metrics?

Explain your answer.

2 pt. **a.** Number of methods in a class.

2 pt. **b.** Average number of methods per class in the project.

2 pt. **c.** Number of (other) classes that a class depends on.

2 pt. **d.** Number of commits per developer.

2 pt. **e.** Number of methods in a project whose names contains the letter 'b'.

10 You are a software developer at a company, developing and maintaining an application built in Java.
3 pt. Your team lead asks you to migrate from Swing to JavaFx, so that instead of JButton you'll use Button, instead of JLabel you'll use Label, etc.

Your colleague proposes to start with a global search-and-replace, automatically removing all occurrences of the letter J from the codebase.

Give at least three examples of what could possibly go wrong.

11 The following guidelines were given to students that want to write good code:

1. Write short methods
2. Give meaningful names
3. Write simple methods
4. Write code once
5. Keep signatures small
6. Separate concerns in classes
7. Keep components balanced
8. Keep the code clean
9. Automate everything
10. Keep improving the code

For each of them find a legitimate excuse to break the rule locally, or explain why the guideline is actually a universally unbreakable rule.

For example:

- A legitimate excuse to break rule 9: "This program will definitely only be run once with these parameters, so there is no need to add them to a makefile."
- A reason why rule 10 is a universally unbreakable law: "Software that is not being improved becomes legacy".

1 pt. **a.** Write short methods.

1 pt. **b.** Give meaningful names.

1 pt. **c.** Write simple methods.

1 pt. **d.** Write code once.

1 pt. **e.** Keep signatures small.

1 pt. **f.** Separate concerns in classes.

1 pt. **g.** Keep components balanced.

1 pt. **h.** Keep the code clean.

12 As it was discussed in the lecture, the five essential elements of modular software design are

2 pt.

1. purpose,
2. connection,
3. interface,
4. encapsulation, and
5. implementation.

Pick one, briefly describe what it means, and argue why it is important.

Thank you, your answers were saved. You will be informed about your grade after the exam correction.

Take care!