Cover page

Department of (department): Department of Computer Science

Examination paper for (course code) (course title): (TDT4242) (Advanced Software Engineering)

Academic contact during examination: Jingyue Li
Phone: 9189 7446
Examination date: 7 June 2018
Examination time (from-to): 9.00 – 13.00
Permitted examination support material: C
Other information:

Students will find the examination results on the Studentweb. Please contact the department if you have questions about your results. The Examinations Office will not be able to answer this.

Introduction

In this exam, you can score a maximum of 50 points. The remaining 50 points for the semester comes from the compulsory exercises.
If you feel that any of the problems require information that you do not find in the text, then you should
• Document the necessary assumptions
• Explain why you need them
Your answers should be brief and to the point.

1. Requirements prioritization

(2 points)
Explain how to prioritize software requirements using the "cumulative voting" approach, and benefits and limitations of this approach.

Answers:
(1/2 points) Stakeholders are given many imaginary units (100 dollars, 1000 points, etc.). Points are distributed among requirements to prioritize

- 100 points
  Req1: 40
  Req2: 20

Req3: 30

Req4: 10

(1/ 2 points) Benefits and limitations

- Easy to use and fast
- Eligible to small number of requirements, not scalable

2. Requirements case study

(16 points)

Company A is going to development a robot to deliver objects, e.g. mail, to employees' office. The requirements that Company A receive from its customer are as follows:

We are going to develop a robot to help deliver objects, e.g. mail, to office to our employees. We want a robot that is efficient, safe, and energy saving. We hope that after we give a map to the robot, and mark out the office that we want to the robot to go, the robot can reach there autonomously. In addition, we would like the robot to have sensors to avoid collisions with obstacles, because there are always people walking around in the office area, and people can also leave something on the floor. The robot should move autonomously without our intervention. However, we sometimes want to interrupt the robot if necessary. We want the robot to move with a default speed. The robot should leave the object outside the door of the office. We hope that the robot use battery to save energy and can charge by itself. When the robot is close to people, the robot should go to the charge site to charge its battery.

Task 1: Make requirement boilerplates and re-express the requirements using the boilerplates you make (2 points)

Task 2: Categorize the textual requirements according to the requirements quality metrics: *ambiguity*, *inconsistency*, *forward referencing*, and *opacity*. If one requirement has several quality issues, list and explain all of them. Then, try to fix the requirements quality issues of each requirement and write down the improved requirements. (6 points)

Task 3: Make a requirement reading guideline for testers, using template for perspective-based reading.

The **Introduction** section of the guideline is "Requirements must be testable". Your task is to fill in the **Instruction** and **Questions** sections of the guideline. In the **Questions** section, you need to fill in at least 3 questions. (4 points)

Task 4: Based on your improved requirements, use the informal temporal patterns below to describe goal decomposition and agent responsibility model of the robot to deliver objects. In the agent responsibility model, a goal should be placed under the responsibility of one or several agents. If the agent is not specified in the customer requirements, you can define an agent yourself.

(Note: If it is difficult to draw the goal decomposition and agent responsibility model in the digital exam context, you can use text to describe them. For example, top level goal **1** is: ... The sub-goal 1.1 is ..., the sub-goal 1.2 is ..., 1.1. and 1.2 have AND relationship, the agent of 1.1 is ...) (4 points)

Achieve [TargetCondition]
Cease[TargetCondition]
Maintain[GoodCondition]
Avoid[BadCondition]

Improve[TargetCondition]
Increase[TargetQuantity]
Reduce[TargetQuantity]
Maximise[ObjectiveFunction]
Minimise[ObjectiveFunction]

To answer the questions in a structured manner, it is better to answer like follows:

Task 1: ...

Task 2: ...

Task 3: ...

Task 4: ...

Answers:

Task 1: (1/2 points) You need to show meaningful boilerplates. Examples are:

The <system> shall <function>

The <user> shall able to <capability>

(1/2 points) You need to show that the requirements are reformulated using your boilerplates.

Task 2: There are more than one case in each requirement error category. These are just examples. If you give other valid cases, you can also get full points.

(1/6 points) ambiguity: "we would like the robot to have sensors to avoid collisions with obstacles", no definition of obstacles.

(1/6 points) inconsistency: "The robot should move autonomously without our intervention. However, we sometimes want to interrupt the robot if necessary." Inconsistent control.

(1/6 points) forward referencing: "We want the robot to move with a default speed." No definition of default speed.

(1/6 points) opacity: When the robot is close to people, the robot should go to the charge site to charge its battery. Requirement rational or dependencies are invisible, or are not meaningful.

(2/6 points) Meaningful fix of the above requirements errors. Each fix counts 0.5 point.

Task 3: (1/4 points) Instruction: Read the requirement to answer the following questions.

(3/4 points) Questions (These are many valid questions. These are just some examples. If you provide minimum three valid questions, you will get full points):

- Are there indefinite pronouns in the requirements?
- Is there passive voice in the requirements?
- Are there vague words in the requirements?
- Is it costly to test the requirement?

Task 4: Several answers could be correct here. The goal-model should contain AND&OR relationship (1 point), agents are linked to goals (1 point), meaningful goal model overall (2 points).

3. DevOps terms

(3 points)

Explain the following three DevOps terms: lead time, cycle time, and percentage complete and accurate.

Answers:

(1/3 points) Lead time is the time from the moment when the request was made by a client and placed on a board (or a ticket is created) to when all the work on this item is completed and the request was delivered to the client. It is the total time the client is waiting for an item to be delivered.

(1/3 points) Cycle time is the amount of time, that the team spent actually working on this item (without the time that the task spent waiting on the board or the ticket). The cycle time should start being measured, when the item task enters the “working” column, not earlier.

(1/3 points) Percentage complete and accurate: percentage of artifacts that could be consumed by the practitioners receiving them without need to modification or rework.

4. Firewall approach

(2 points)

Explain what the firewall approach of regression test selection is, and explain how to determine the firewall of object-oriented systems.

Answers:

(1/2 points) A firewall in regression testing separates the classes that depend on the class that is changed from the rest of the classes. Firewall approach is based on the first-level dependencies of modified components.

(1/2 points) Given two successive versions of an OO-system, find the difference of the two versions and identify those classes that have changed. If any of the changed classes are in an inheritance hierarchy, also consider descendants of the changed class as changed. For each changed class, identify all the classes that send messages to the changed class or receive messages from the changed class and include them in the "firewall".

5. Domain testing

(5 points)

An online PC shopping web application has five variables (Availability, Payment method, Delivery method, PC type, and Damage insurance).

The possible values of the variables are as follows.

- Availability: Available (AVA), Not In Stock (NIS), DIScontinued (DIS)
- Payment method: Credit Card (CC), Gift Voucher (GV)
- Delivery method: Mail (MA), UPS (UPS), Fedex (FE)
- PC type: DESKtop (DESK) and LAPtop (LAP)
- Damage insurance included in price: With Insurance (WithIN), Without insurance (NoIN)

You task is to write all all-pair combinatorial test cases (Abbreviations of each variable value can be used)

Answers:

(5/5 points) Several combinations can be correct. This is just one possible solution. All pairs mean that "every value of a variable is combined with every value of every other variable at least once". If you miss any pairs, or if you do not show that you understand how to make the all-pair test cases, you will lose all points.

Availability	**Delivery method**	**Payment method**	**PC type**	**Insurance**
AVA	MA	CC	DESK	WithIN
AVA	UPS	GV	LAP	NoIN
AVA	FE	CC	DESK	WithIN
NIS	MA	GV	LAP	NoIN
NIS	UPS	CC	**LAP**	WithIN
NIS	FE	GV	**DESK**	NoIN

DIS	MA	CC	DESK	**NoIN**
DIS	UPS	GV	LAP	**WithIN**
DIS	FE	CC	DESK	WithIN

6. MBT

(2 points)

Explain what MBT (Model-based testing) is, and how MBT works.

Answers:

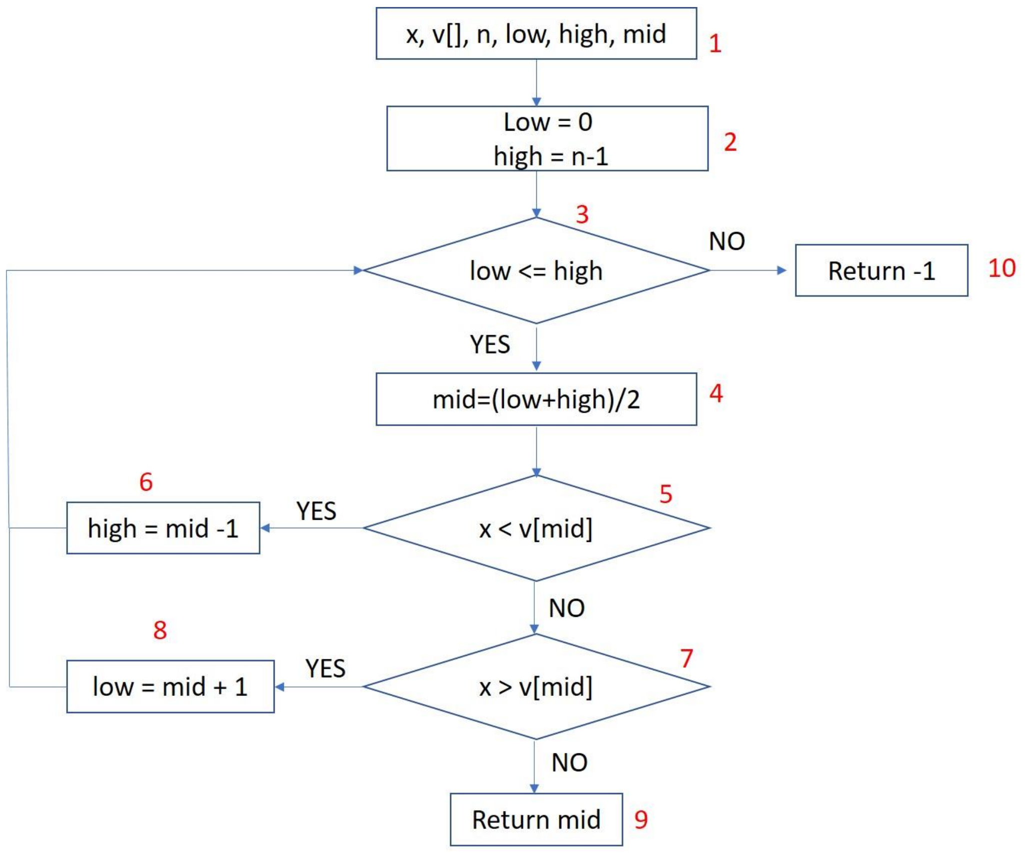
(1/2 points) What MBT is: MBT is a testing approach where test cases are automatically generated from models. The models are the expected behavior of the system under test and can be used to represent the test strategy.

(1/2 points) How MBT works: MBT usually includes the following steps: 1. Design a test model. 2: Select some test generation criteria. For example, the test generation criteria could be full state coverage, if the model is a finite state machine. 3. Generate the tests. 4. Execute the tests.

7. Data flow testing

(4 points)
The following CFG (Control Flow Graph) is from the binary search code. Based on the CFG, your tasks are:

- Identify paths to be covered for the variables "high" and "mid", if All Uses (AU) is the test strategy to be applied



```
int binarysearch(int x, int v[], int n){
        int low, high, mid;
        low = 0;
        high = n-1;
        while (low <= high){
           mid = (low + high) /2;
           if (x < v[mid])
                 high = mid -1;
           else if (x > v[mid])
                 low = mid + 1;
                 else
                   return mid;
        }
        return -1;
}
```

Binary search is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array; if they are unequal, the half in which the target cannot lie is eliminated and the search continues on the remaining half until it is successful. If the search ends with the remaining half being empty, the target is not in the array.

Answers:

(2/4 points) Paths to be covered for the variables "high".

<2,3><2,4><6,3><6,4>

(2/4 points) Paths to be covered for the variables "mid".

<4, 5><4, 6><4, 7><4, 8><4, 9>

8. Software Ecosystem

(5 points)

Explain characteristics, success factors, and challenges of an application-centric software Ecosystem, and list two examples and explain the examples briefly.

Answers:

For the characteristics and success factors, and challenges, you need list at least two out of the possible items to get full points.

(1/5 points) Characteristics:

- Starts from a successful application
- Provide APIs for developing customer-specific functionality
- Application developers extend the basic application
- Deep integration for data, workflow and user experience

(1/5 points) Success factors:

- Number of customers
- Simplified contribution of 3rd party extensions
- Mechanisms for extending data models and workflows

(1/5 points) Challenges:

- Product versus platform strategy
- Viable business model for 3rd party developers

These examples are just reference examples. If you provide other valid examples, you can also get full points.

(1/5 points) Example 1: Facebook which provides APIs for 3rd party developers to develop customer-specific functionalities.

(1/5 points) Example 2: Amazon also provides APIs for 3rd party developers to develop customer-specific functionalities.

9. Service oriented architecture

(5 points)

List benefits of SOAP (Simple Object Access Protocol) web service and benefits of REST (REpresentational State Transfer) web service?

Answers:

(3/5 points) You need to list minimum 3 items below to get full points

SOAP is good for

- Enterprise services
- High reliability and security
- Asynchronous processing
- Contract-first development
- Stateful operation
- Standards support
- Tool support

(2/5 points) You need to list minimum 2 items below to get full points

REST is good for

- Limited bandwidth (smaller message size)
- Limited resources (no XML parsing required)
- Exposing data over internet
- Combing content from many different sources in a web browser

10. Agile distributed development

(6 points)

Explain communication, trust, and control challenges of agile distributed development and list/propose strategies to address those challenges.

Answers:

(1/6 points) Communication challenge: Agile requires more informal communication. However, distributed development makes it difficult to communicate frequently.

(1/6 points) How to address communication challenge (You need to list at least one item):

- Documenting requirements at different levels of formality
- Informal communication but through formal channels
- Constant communication

(1/6 points) Trust challenge: In distributed development, people do not know each other very well. Thus, they do not trust each other very much. However agile development requires high team cohesion and trust.

(1/6 points) How to address trust challenge (You need to list at least one item):

- Focus on well-understood functionalities in early iterations
- Trust but verify

(1/6 points) Control challenge: In distributed development, it is difficult to control process and quality across distributed teams. Thus, frequent quality requirements changes are not welcome. However, in agile development, requirement changes are more acceptable.

(1/6 points) How to address control challenge: Short cycle but not-boxed development