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: 23 May 2019
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. Problem 1

(30 points in total)

1) Explain the purposes of establishing forward and backward traceability in software projects and how to establish forward and backward traceability in software projects and artifacts. (5 points)
   Purpose
   - Forward traceability (2 points)
     - Ensure all requirements are implemented
     - Change impact analysis
   - Backward traceability (2 points)
     - Avoid "gold plating"
     - Change impact analysis
     - Defect impact analysis
     - Root cause analysis of defects

   How to (1 point)

   - Traceability matrix
   - Trace tagging

2) Explain what static backward slicing is and how to create backward slicing using data flow information. (2 points)

   The purpose of code slicing is to choose only a subset of code that is relevant to a particular variable at a certain point to minimize the code for more cost-efficient testing, debugging, or testing. (0.5 points)
   Static backward slicing: A slice with respect to a variable *v* at a certain point *p* in the program is the set of statements that **contributes** to the value of the variable v at p (0.5 points)
   To establish static backward slicing, all Defs and All P-uses of a variable v before a certain point p are chosen (1 point).

3) Explain decomposition-based, call-graph based, and path-based integration testing strategies and compare their pros and cons. (6 points)
   Decomposition-based (2 points): integration testing can be a top-down, bottom-up, sandwich, or big bang. The basic idea is to use "stub" and "driver" to simulate the called function or call function.
   - Pros: Intuitive and easy fault isolation
   - Cons: Need "stub" or "driver"

   Call-graph based (2 points): integration testing is based on the call graph of neighbors or interface matrix
   - Pros: Do not need "stub" or "driver"
   - Cons: Difficult to isolate and locate the fault

Path-based (2 points): integration testing is like an enlarged path-based unit test. Here the node is not a statement but a component.

- Pros: Closely coupled with actual system behavior
- Cons: extra effort is needed to identify message path

4) Explain the general regression test selection process. (3 points)
   Regression test selection process usually includes three main steps:
   - Run the tests and establish the link between the test and the code executed (1 point)
   - After the code is changed, identify the changed code (1 point)
   - Based on the link between the code and the test, choose only those tests that are related to the changed code and re-run them (1 point)

5) Explain the essential ideas of the two types of standards to verify safety-critical systems. (2 points)
   - "How-standards" focus on how we should work – e.g., how we should work to achieve safety or security. At present, most standards are "How"-standards (1 point)
   - "What-standards" (also called Goal-based standards) focus on what shall be achieved, not on how to achieve it. It is up to the developers to convince the assessor that what they do will meet the standard's requirements (1 point)

6) Explain what adaptive random test is and its benefits. (3 points)

The purpose of the adaptive test is to generate test cases that are evenly spread over the entire input domain as possible. You start with one test case that is generated randomly. Then for the next test case, you pick the one that is the farthest away from all executed test cases. The benefit is that it is good for F-measure, i.e., it usually requires fewer test cases to detect the first failure comparing to random test.

7) Explain why code inspection and testing are complementary. (4 points)

The benefits of testing are that (1 point)

- Can, at least partly, be automated
- Can consistently repeat several actions
- Fast and high volume

However, testing has also limitations (1 point)

- Is only a spot check
- May need several stubs and drivers

The benefits of code inspection is that (1 point)

- Can see the complete picture, not only a spot check
- Can use all information in the team
- Can be innovative and inductive

However, code inspection could be: (1 point)

- Unreliable (people can get tired)
- Slow and low volume

8) Explain MySQL dual licensing. (2 points)

   - MySQL free use (1 point)
     - Within a web site
     - ISP may make MySQL available to its customers for free
     - All projects that themselves run under the GPL
   - MySQL commercial license, e.g., (1 point)
     - Develop and sell a commercial product that is geared toward MySQL as the database

9) Based on the content of the guest lecture "coordination in large-scale agile development," describe some coordination challenges and possible solutions in large scale agile projects. (3 points)

   - Challenges: Scrum-of-Scrum meetings involving representatives from all teams were severely challenged: the audience was too wide to keep everybody interested. Inter-group coordination becomes a major challenge when groups enjoy high levels of autonomy (2 points).
   - Possible solutions (List of anyone below can get 1 point)
     - Have enough formal arenas to enable informed informal coordination
     - Be prepared to add or change the content of the formal arenas over time as needs change
     - Specialized roles in the team contribute to cross-team coordination
     - Mini demos enable faster feedback and closed-loop communication with customers
     - Trust in line with clear decision-making arenas enables autonomous handling of problems with feedback loops to the project

2. Problem 2: Requirement Engineering

(8 points)

Company A is going to pay Company B for developing Autonomous Truck Platooning. Truck platooning (as shown in the above picture) is the linking of two or more trucks in convoy, using connectivity technology and automated driving support systems. These vehicles automatically maintain a set, close distance between each other when they are connected for certain parts of a journey, for instance on motorways.

The requirements Company B got from Company A are as follows.

We are going to develop Autonomous Truck Platooning. We want some numbers of the trucks can drive autonomously, and other trucks are driven by humans. The autonomous trucks should drive at a default speed. We hope the truck platooning can drive along the route we set in GPS before the journey and follow the human drivers' command to change the route. We hope the autonomous trucks should have three chairs for the drivers. We want the truck to drive safely. So the trucks should have sensors to avoid collisions with obstacles because there are always other vehicles or obstacles in the motorway. When there are obstacles on the road, the trucks should stop. We also want the trucks to drive efficiently, which means that the trucks should minimize the distance between them and maximize the speed of the whole Platooning. The trucks should use sensors to detect the distances between trucks and the speed of the trucks.

Task 1: Identify quality issues in these requirements according to the requirements quality metrics: *ambiguity*, *inconsistency*, *forward referencing*, and *opacity*. If one requirement has several quality issues, list all of them. Then, try to fix the requirements quality issues of each requirement and write down the improved requirements. (4 points)

Each category of quality issues can have several cases. If you list any of the cases in the four categories and fix them, you will get one point of each category.

- Ambiguity (1 point): *So the trucks should have sensors to avoid collisions with obstacles because there are always other vehicles or obstacles in the motorway (Not explain obstacles).*
- Inconsistency (1 point): E.g., *We hope the truck platooning can drive along the route we set in GPS before the journey and follow the human drivers' command to change the route (Conflict on who can make the control decision).*
- Forward referencing (1 point): E.g., *The autonomous trucks should drive at a default speed (Does not define default speed).*
- Opacity (1 point): We hope the autonomous trucks should have three chairs for the drivers (not relevant to the autonomous truck).

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

- The list of meaningful boilerplate is 1 point.
- Re-expressing the requirement using boilerplates is 1 point.

Task 3: Based on your improved requirements, use the informal temporal patterns below to describe goal decomposition and agent responsibility model of the Autonomous Truck Platooning. 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 ...). (2 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: ...

This is just one possible solution. Other solutions which can show that the student understands the goal-oriented approach can also get full points. The goal hierarchy and proper use of the temporal pattern will count 1 point, and the proper link between the goal (AND or OR relation is specified) and agent listed and linked to the goals will count 1 point.

1. Achieve[Safe and Effective driving]

  1.1 Avoid[Collisions]

    1.1.1 Maintain [More than the minimum safe distance between trucks] – Agents: Camera and Lidar

    AND

    1.1.2 Cease [Unavoidable obstacle is identied] – Agent: Camera, Lidar

    AND

    1.1.3 Avoid [Overspeed] – Agent: Speed meter

  1.2 Achieve[Effective driving]

      1.2.1 Maintain [Speed close to the speed limit] – Agent: Speed meter

      OR

    1.2.2 Avoid [Uncessary slow down]- Agent: Speed meter, Camera, Lidar

3. Problem 3: Testing

1. Combinatorial test (4 points)

An online car renting web application calculates the car rental prices based on four variables (Party size, Car specification, Mileage/Kilometres, and Damage insurance).

The possible values of the variables are as follows.

- Party size: Small (S), Medium (M), and Large (L)
- Car specification: Air Conditioning (AC), Automatic Transmission (Auto), and Manual Gearbox (Man)
- Mileage/Kilometres: Limited (Lim), Unlimited (Unlim)
- Damage insurance included in the price: With Insurance (WithIN), Without insurance (NoIN)

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

This is one reference solution (4 points).

Party size	**Car specification**	**Mileage/Kilometres**	**Damage insurance**
S	AC	Lim	WithIN
S	Auto	Unlim	NoIN
S	Man	Lim	WithIN
M	Ac	Unlim	NoIN
M	Auto	Lim	WithIN
M	Man	Unlim	NoIn
L	AC	Lim	**NoIN**
L	Auto	Unlim	**WithIN**
L	Man	*Lim*	*WithIn*

2. System test (3 points)

You are assigned a task to run scalability testing and stress testing of a popular online air ticket booking application. Explain the purpose of scalability testing and stress testing and how to perform scalability testing and stress testing to test the application. (3 points)

The purpose of scalability testing is to test how well the system can be scaled up to process the increased workload. To perform scalability testing, the testers can gradually increase the workload to the system, and also add more servers and workload managers, in order to measure how well the system can process the increased workload through adding more servers (1 point).

The purpose of stress testing is to test how well the system can perform acceptably under worse-case condition. The system is deliberately stressed by pushing it to and beyond its specific limits for a while in order to identify some bugs related to memory, e.g., memory leak (1 point).

For the online air ticket booking application, to run scalability testing, the testers will simulate the increased number of parallel transactions and add more servers and configure the workload manager to check how well the added servers deal the increased workload. To run the stress testing, the testers will simulate the maximum number of parallel transactions for a while to see if the performance of the system is still acceptable (1 point).

4. Problem 4: Code refactoring case study

Identify bad code smells in the following code and propose how to refactor them. The proposal of code refactoring could be refactored python code or pseudo-code to explain how to refactor. (5 points)

Note: The code is a working code. Your task is not to identify the security vulnerabilities of the code. Your task is to identify bad code smell and to refactor the code.

```
"""
Package: utils.config
"""

CONFIG_NAME = {
  "ENABLE_LOGGING": "enable_logging",
  "LOGGING_LEVEL": "logging_level",
}

def get_logging_level():
  pass

class ConfigHelper:
  def get(self, config_name, default=None):
      pass

def set(self, config_name, value):
      pass

def _get_settings_helper(self):
      pass

def get_logging_level():
      pass

def is_logging_enabled():
      pass

class LOGGING_LEVEL:
  VERBOSE = "verbose"
  STANDARD = "standard"
```

The main code smells are:

- Duplicated code: def get_logging_level() appears twice (1 point)
- Naming convention: function names or variable names are not inline with Python naming convention (2 points)

- Too small classes or constants : the classes and constants can be moved to a separate file, be removed, or be merged to optimize the readability of the code (2 points)