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)

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

3) Explain decomposition-based, call-graph based, and path-based integration testing strategies and compare their pros and cons. (6 points)

4) Explain the general regression test selection process. (3 points)

5) Explain the essential ideas of the two types of standards to verify safety-critical systems. (2 points)

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

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

8) Explain MySQL dual licensing. (2 points)

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)

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)

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

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: ...

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)

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)

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"
```