Software Dependability

Fabio Palomba

Midterm Exam

Apr 29, 2019 @ 9:00 - 11:00

Student name and number: _____

Instructions, Information, and Rules

Please, read these instructions carefully. Failure to comply with *any* of the following instructions means *invalidation of your exam*:

1. You have two hours to complete the midterm exam.
2. The exam consists of 18 multiple-choice and 6 open questions. For each multiple-choice question, *only one* answer is correct. The questions are shuffled among the attendees.
3. A correct answer to a multiple-choice question will add one point to your grade. An incorrect answer will *decrease* your grade of one point. Not answering a question is *not* considered an incorrect answer.
4. A correct and detailed answer to an open-question will add two points to your grade; A correct but partial answer will add one point to your grade.
5. In principle, given (i) M correct answers and I errors to the multiple-choice questions, and (ii) O correct answers and P partial answers to the open questions, the grade G of the midterm exam will be determined by the formula:

   $$G = (((1 * M) - (1 * I)) + ((2 * O) - (1 * P))).$$

   The grade G ranges between 0 and 30. It will be increased by up to the three points of the hands-on assignment *if and only if* (1) G will be higher than 18 and (2) the assignment has been delivered by the fixed deadline (Apr 26 at 12:00 - I will check the exact time of the submission using the `git log` command). Moreover, in case you are interested in further increasing your grade, you can do the *oral* part of the exam this afternoon. The outcome of the midterm will be made available on the GITHUB repository of the course[1] within three hours since the end of the exam. To book a slot for the oral exam, you must send me an e-mail by this afternoon at 15:00. The oral exam will take place in the SESA LAB at 16:00 and comprise three questions spanning across all topics treated in the course.
6. Fill in your answers on the provided answer sheet and hand it in at the end of the exam.
7. The use of books, papers, computers, phones, smart-watches or other material is *not* permitted.
8. You are *not allowed* to talk to your colleagues; this is forbidden for *both* questions related to the exam and any other form of communication (requests for pens, papers, etc.). All you need for passing the exam is this document and a pen: if you do not have a pen, ask *before* the beginning of the exam. Note that there exist *different forms of communications*: trying to spy the exam of one of your colleagues is considered a forbidden communication.

Good luck!

[1] Link here: https://github.com/fpalomba/SWDependability-Unisa2019.

Page left empty on purpose.

Part I

Dependability Requirements

1. Select the description that best explains the *dependability* of a software system:

   A. The degree to which a system operates without interruption, delivers the services that are expected by the stakeholders, does not have negative effects on the system's environment and does not damage its data or the system itself.

   B. The degree to which a system can deliver the services that end-users expect, regardless of the surrounding effects on data preservation and environmental constraints.

   C. The degree to which a system is able to provide services that can defensibly be trusted within a certain specified time-window or time-period.

   D. The degree to which a system complies to the organization standards in terms of quality of product and processes, which only includes the management of people, failures, communications with the stakeholders, and prompt reaction to external events such as new incoming change requests.

2. Fill in the blank: The _____ of a software system comprises methods and techniques to enable a safe management of data of its users as well as the safety of its users.

   A. maintainability

   B. reliability

   C. availability

   D. None of the others

3. Fill in the blank: The availability of a software system can be stimulated through _____.

   A. the hiring of database experts.

   B. a mere definition of a strategy for exception handling.

   C. refactoring of code smell instances.

   D. data redundancy and fault recovery approaches.

4. Define the dependability requirements and provide an example showing whether and how they are connected to each other. NB: Do not exceed the available space.

Part II

Code Smells and Refactoring

5. Fill in the blank: One of the principles driving refactoring is cohesion; this indicates _____.
   A. the degree to which a certain class or module relies on other classes or modules.
   B. how focused the responsibilities of a certain class or module of a system are.
   C. the extent to which a class or module is feasibly testable.
   D. none of the others.

6. Select the description that best explains the term *refactoring*:
   A. It consists in a set of practices that allow stakeholders to control the evolution of source code and its associated requirements.
   B. It is the process of improving the external behavior of the source code without altering its internal structure.
   C. It is the process of improving the internal structure of the source code without altering its external behavior.
   D. None of the above.

7. Consider the case in which you have a method that has more dependencies with another class with respect to the one it is actually in. Fill in the blank: This method is affected by a _____ code smell.
   A. Feature Envy
   B. Inappropriate Intimacy
   C. Long Method
   D. Middle Man

8. Describe the main characteristics making a class affected by the *Blob* code smell. Describe also what is the corresponding refactoring operation and what are the steps required to perform it. NB: Do not exceed the available space.

Part III

Software Vulnerabilities

9. Select the description that best explains the term *vulnerability*:
   A. It is a fault made by a programmer that leads to a failure.
   B. It is a maintainability problem that may make the source code more prone to be defective.
   C. It is a weakness in the source code that be exploited by a threat actor.
   D. None of the above.

10. Fill in the blank: One of the approaches to identify vulnerabilities is fuzz testing; this consists of _____.
   A. applying stochastic models to find vulnerable areas of source code
   B. analyzing the source code without compiling it
   C. testing how the system accepts input data and how it manages corner cases
   D. running machine learning models trained on previous available vulnerability data

11. Select the description that best explains the *symbolic execution* strategy for statically identifying vulnerabilities:
   A. It represents program's inputs through symbol values rather than actual data, and produces algebraic expressions that simulates the implementation process.
   B. It formally describes the source code and checks for semantic violations of the requirements.
   C. It includes source code patch and binary code patch comparison with the aim of finding loopholes.
   D. None of the above.

```
void readChar() {
  // code here…
  char s[256];
  scanf("%s", s)
}
```

Figure 1: Source code snippet presenting a vulnerability.

12. Describe the type of vulnerability shown in Figure 1 and how an external attacker may exploit it. NB: Do not exceed the available space.

Part IV

Machine Learning for Software Dependability and Defect Prediction

13. Which of the following sentences about supervised learning techniques is *false*?
    A. The machine learner requires a set of known observations to be effectively trained.
    B. The ground-truth definition is key to correctly train a supervised machine learner.
    C. This type of learning is primarily used when the data needs to be organized to find patterns.
    D. Most of the supervised algorithms require a configuration of their hyper-parameters.

14. Fill in the blank: Data imputation techniques _____.
    A. are used to discard highly correlated independent variables
    B. are used in case the training data is incomplete
    C. are solely based on statistic analysis of the training data
    D. none of the above

15. Fill in the blank: The problem of multi-collinearity refers to _____.
    A. a potential bias in the interpretation of the performance of machine learning models caused by the little to null ability of a classifier to learn what distinguishes the dependent variable classes.
    B. the reduction of the input space of the training set, namely a filtering mechanism of the instances that assigns a higher weight on some of them.
    C. a potential bias in the interpretation of the performance of machine learning models caused by the little to null ability of a classifier to assign an explanatory meaning to some of the independent variables.
    D. a potential bias in the interpretation of the performance of machine learning models caused by the missing configuration of the hyper-parameters of the selected classifier.

16. Define what is defect prediction. Describe also a typical work-flow, from feature extraction to results interpretation, leading to the definition and validation of a defect prediction model. NB: Do not exceed the available space.

Part V

Software Metrics and Traceability

17. Traceability can support many software engineering activities, such as:
    A. Code inspection, software validation, and software testing.
    B. Software architecture design, software quality evaluation, and effort estimation.
    C. Control of requirement evolution, management of risks associated with missing requirements, and management of maintenance costs.
    D. Release Planning, requirements specification, and UML diagrams specification.

18. Fill in the blank: Traceability is the discipline that shows _____.
    A. the relationships between software developers' communications.
    B. forward and backward relationships linking requirements with design, implementation, test, and maintenance.
    C. the backlog between clients and software developers.
    D. the backlog between software developers and software management.

19. Fill in the blank: Given a software artifact, you have a *forward traceability link* when you look for software artifacts that have been developed _____.
    A. in a later stage of software development.
    B. in an earlier stage of software development.
    C. in the context of requirement specification.
    D. in the context of UML modeling.

20. Define the goal of cohesion, coupling, and complexity metrics. Provide also an example metric for each category; finally, focus on the McCabe's cyclomatic complexity, reporting the steps required to compute it. NB: Do not exceed the available space.

Part VI

DevOps and Infrastructure Code

21. Select the description that best explains the concept of *omniscient DevOps analytics*:

    A. It represents the set of practices for which all aspects of software development are measured and controlled over time.

    B. It represents the set of practices for which the technical aspects of software development are measured and controlled over time.

    C. It implements the idea of measuring developers' communications and coordination to best organize software development teams.

    D. None of the above.

22. Fill in the blank: Continuous Architecting is a DevOps tactic that _____.

    A. aims at continuously building an infrastructure that allows to merge all developer work-copies to a shared mainline.

    B. aims at continuously building an infrastructure that allows to check and validate the source code changes applied on a shared mainline.

    C. aims at continuously building an infrastructure that allows to test, build and deploy, but also take quality attributes into account.

    D. aims at continuously building an infrastructure that allows to verify both the status of the developer committing the change and the change itself.

23. Which of the following sentences about containerization is *false*?

    A. Containerization automates the deployment of applications inside software containers.

    B. Containerization provides an additional layer of abstraction and automation of operating systemlevel virtualization.

    C. Containerization allows the execution of a software project within a specific environment.

    D. Containerization allows the automatic creation of images using a build script.

24. Define what is DevOps. Describe also the main processes involved in a DevOps pipeline. NB: Do not exceed the available space.