A.Y. 2019-2020 Software Engineering 2
Mandatory Project: goal, schedule, and rules

READ THIS VERY CAREFULLY
NO EXCUSE FOR IGNORING WHAT WE WRITE HERE

Table of contents

1 Goal and approach

The objective of this project is to apply in practice what you learn during lectures with the purpose of becoming familiar with software engineering practices and able to address new software engineering issues in a rigorous way. The project includes two assignments:

1. The preparation of a Requirement Analysis and Specification Document (RASD) for a problem we provide you.
2. The definition of the Design Document (DD) for the system considered in point 1 above.

The two assignments will be reviewed during the final discussions that will take place during the winter exam sessions according to a schedule that will be proposed in the forthcoming months. The evaluation will assess the quality of the artifacts you prepare (accurateness, completeness, soundness) and the quality of your presentation (if you are able to explain your point in an appropriate way and if your presentation fits in the allowed time). Please check the introduction to the course for more information on the evaluation criteria for the mandatory project. The two assignments are described in the rest of this document.

2 Project schedule

- Group registration deadline 13/10/2019
- RASD submission deadline 10/11/2019
- DD submission deadline 9/12/2019
- Final presentation (to be scheduled)

All deadlines are assumed to expire at **23:59** of the days listed above.

3 Rules

- The project MUST be developed in groups of two or three persons. Groups composed of a single student are allowed even if strongly discouraged. The assignments and the corresponding expectations of the professor will be calibrated based on the size of the group.
- Each group MUST register to the project following the steps indicated in Section 4. “Mixed” groups involving students of the two sections are allowed. When registering, each such group will need to indicate a single “reference” professor. This will be the one holding the discussion at the end of the course and deciding the grade. The choice is up to the students, but if we will realize that there is an unbalance between the groups under the responsibility of each of us, we may change your reference professor. In this case, we will inform you a few days after the group registration deadline.
- Each group MUST provide the requested artifacts within the stated deadlines. A delay of a few days, if notified in advance to the reference professor, will be tolerated but it will also result in a penalty in the final score. It is mandatory to provide these artifacts and to present them to the reference professor in a final meeting that will be scheduled later.
- Each group MUST release artifacts by committing them into a specific folder of the github repository created for the project (see the following section).
- Each group MUST use the repository not only to upload the final versions of deliverables but also to commit intermediate versions. We want to see commits performed by all group members.
- The material included in your artifacts is not fixed in stone. You can (and are encouraged to) provide updates at any point before the end of the course, if you think these are needed.
- During the development of the project each group will keep track of the number of hours each group member works toward the fulfillment of each deadline.
- For any question related to the project that could be interesting also for the other groups, please use the forum available on the Beep website. We will answer as promptly as possible.
- This year, some groups (not all) will be assigned a tutor from industry. Tutors will provide advice concerning the quality of the produced documents and how these can be improved. The selection of the groups which will be assigned a tutor, and the assignment itself, will be done randomly, and the groups will be picked among those that will have indicated in their registration form an interest for this tutoring activity. Please notice that there will not be enough tutors for all groups. Thus, while, in the long term, we plan to have all groups supported by a tutor, this year we will start with a pilot experiment that will involve 10-15% of the total number of groups. The assignment of groups to tutors will be communicated by the end of October. Students who will be assigned a tutor will invite him/her to the project repository and will schedule with him/her two meetings, one right after the RASD submission and the other right after the DD submission. They will then update their documents based on the received feedback. In the assessment of groups with tutors, the assessment criteria will be the same as for non-tutored groups, but we will take into account the status of the initial version of each document submitted to tutors, the received suggestions, and the improvements achieved after applying the changes.

4 Group registration and organization of your repository

You should form your group and register it by going through the following steps:

1. Create a private repository for your project on Github (https://github.com). Please note that, as students, you have the possibility to create a private Github repository for free. Your repository should be named by combining the names of all group members. For instance, BianchiRossiVerdi will be the name of the repository of the group composed of the students Tommaso Bianchi, Maria Rossi e Veronica Verdi. Make sure that all group members have a Github account and have access to the repository. Moreover, invite your reference professor (Github accounts dinitto and matteo-g-rossi) to access your repository (reading access is sufficient).
2. Register your group by filling in the following form https://forms.gle/NF1F8KcFckdvVgkb8. Do not forget to include in the form all relevant data!
3. Create a directory for each of the documents you will be working on.
4. Moreover, create a directory called DeliveryFolder where, by the due deadlines, you will commit the pdf version of your documents (name it RASD1.pdf or DD1.pdf, depending on the document you are releasing) plus any additional file you may want to include (e.g., the Alloy model and/or any UML model).
5. After the deadline for submission, should you need to update your document, you can commit in the same folder another pdf file with an increased version number, e.g., RASD2.pdf. The new file should include a section that describes the performed changes.

5 The problem: SafeStreets

SafeStreets is a crowd-sourced application that intends to provide users with the possibility to notify authorities when traffic violations occur, and in particular parking violations. The application allows users to send pictures of violations, including their date, time, and position, to authorities. Examples of violations are vehicles parked in the middle of bike lanes or in places reserved for people with disabilities, double parking, and so on.

Basic service: SafeStreets stores the information provided by users, completing it with suitable meta-data. In particular, when it receives a picture, it runs an algorithm to read the license plate (one can also think of mechanisms with which the user can help with the recognition), and stores the retrieved information with the violation, including also the type of the violation (input by the user) and the name of the street where the violation occurred (which can be retrieved from the geographical position of the violation). In addition, the application allows both end users and authorities to mine the information that has been received, for example by highlighting the streets (or the areas) with the highest frequency of violations, or the vehicles that commit the most violations. Of course, different levels of visibility could be offered to different roles.

Advanced function 1: If the municipality offers a service that allows users to retrieve the information about the accidents that occur on the territory of the municipality, SafeStrees can cross this information with its own data to identify potentially unsafe areas, and suggest possible interventions (e.g., add a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking).

Advanced functions 2: In addition, the municipality (and, in particular, the local police) could offer a service that takes the information about the violations coming from SafeStreets, and generates traffic tickets from it. In this case, mechanisms should be put in place to ensure that the chain of custody of the information coming from the users is never broken, and the information is never altered (e.g., if a manipulation occurs at any point of the image showing the violation, for example to alter the license

plate, the application should discard the information). In addition, the information about issued tickets can be used by SafeStreets to build statistics, for example about the most egregious offenders, or the effectiveness of the SafeStreets initiative (e.g., by looking for trends in the issuing of tickets).

You are required to explore the above problem and define the corresponding RASD and DD. In particular:

- **Single-person groups** are required to focus only on the basic service.
- **Two-people groups** are required to focus on the basic service and on one of the two additional functions (either the one concerning the use of information about reported accidents, or the one about the issuing of tickets).
- **Three-people groups** are required to consider all three functions described above.

Notice that the additional functions are realized with the help and support of the municipality. In particular, while the implementation of the external services is not part of SafeStreets, their interfaces are defined by the SafeStreets designers in collaboration with the people from the municipality, so they fall under the purview of the design of SafeStreets.

6 The documents to be created

Each document you produce will include the following elements:

- **A FRONT PAGE** that includes the project title, the version of the document, your names and the release date.
- **The TABLE OF CONTENTS** that includes the headers of all the first three levels headings in your document with the corresponding page number. At the beginning of this document you find a table of contents that you can use as an example. Since in this document there are no level three heading (e.g., 3.1.1), they are not part of the table of contents as well.

The specific characteristics each document should have are described in the next subsections.

6.1 Assignment 1 - RASD

The *Requirements analysis and specification document (RASD)* contains the description of the scenarios, the use cases that describe them, and the models describing requirements and specification for the problem under consideration. You are to use a suitable mix of natural language, UML, and Alloy. Any Alloy model should be validated through the tool, by reporting the models obtained by using it and/or by showing the results of assertion checks. Of course, the initial written problem statement we provide suffers from the typical drawbacks of natural language descriptions: it is informal, incomplete, uses different terms for the same concepts, and the like. You may choose to solve the incompleteness and ambiguity as you wish, but be careful to clearly document the choices you make and the corresponding rationale. You will also include in the document information on the number of hours each group member has worked towards the fulfillment of this deadline. As a reference structure for your document, you should refer to the one reported below that is derived from the one suggested by IEEE. Please include in the document information about the effort spent by each group member for completing this document.

1. **INTRODUCTION**

A. *Purpose*: here we include the goals of the project
B. *Scope*: here we include an analysis of the world and of the shared phenomena
C. *Definitions, Acronyms, Abbreviations*
D. *Revision history*
E. *Reference Documents*
F. *Document Structure*

2. **OVERALL DESCRIPTION**
   A. *Product perspective*: here we include further details on the shared phenomena and a domain model (class diagrams and statecharts)
   B. *Product functions*: here we include the most important requirements
   C. *User characteristics*: here we include anything that is relevant to clarify their needs
   D. *Assumptions, dependencies and constraints*: here we include domain assumptions
3. **SPECIFIC REQUIREMENTS:** Here we include more details on all aspects in Section 2 if they can be useful for the development team.
   A. *External Interface Requirements*
      A.1 *User Interfaces*
      A.2 *Hardware Interfaces*
      A.3 *Software Interfaces*
      A.4 *Communication Interfaces*
   B. *Functional Requirements*: Definition of use case diagrams, use cases and associated sequence/activity diagrams, and mapping on requirements
   C. *Performance Requirements*
   D. *Design Constraints*
      D.1 *Standards compliance*
      D.2 *Hardware limitations*
      D.3 *Any other constraint*
   E. *Software System Attributes*
      E.1 *Reliability*
      E.2 *Availability*
      E.3 *Security*
      E.4 *Maintainability*
      E.5 *Portability*
4. **FORMAL ANALYSIS USING ALLOY:** This section should include a brief presentation of the main objectives driving the formal modeling activity, as well as a description of the model itself, what can be proved with it, and why what is proved is important given the problem at hand. To show the soundness and correctness of the model, this section can show some worlds obtained by running it, and/or the results of the checks performed on meaningful assertions.
5. **EFFORT SPENT:** In this section you will include information about the number of hours each group member has worked for this document.
6. **REFERENCES**

6.2 Assignment 2 - DD

The *Design document (DD)* must contain a functional description of the system, and any other view you find useful to provide. You should use all the UML diagrams you need to provide a full description of the system. Alloy may also be useful, but not mandatory. You will also include information on the number of hours each group member has worked towards the fulfillment of this deadline. As a reference structure for your document please refer to the following one:

1. **INTRODUCTION**
   - A. *Purpose*
   - B. *Scope*
   - C. *Definitions, Acronyms, Abbreviations*
   - D. *Revision history*
   - E. *Reference Documents*
   - F. *Document Structure*
2. **ARCHITECTURAL DESIGN**
   - A. *Overview*: High-level components and their interaction
   - C. *Component view*
   - D. *Deployment view*
   - E. *Runtime view*: You can use sequence diagrams to describe the way components interact to accomplish specific tasks typically related to your use cases
   - F. *Component interfaces*
   - G. *Selected architectural styles and patterns*: Please explain which styles/patterns you used, why, and how
   - H. *Other design deci*sions
3. **USER INTERFACE DESIGN**: Provide an overview on how the user interface(s) of your system will look like; if you have included this part in the RASD, you can simply refer to what you have already done, possibly, providing here some extensions if applicable.
4. **REQUIREMENTS TRACEABILITY**: Explain how the requirements you have defined in the RASD map to the design elements that you have defined in this document.
5. **IMPLEMENTATION, INTEGRATION AND TEST PLAN**: Identify here the order in which you plan to implement the subcomponents of your system and the order in which you plan to integrate such subcomponents and test the integration.
6. **EFFORT SPENT:** In this section you will include information about the number of hours each group member has worked for this document.
7. **REFERENCES**