Software Engineering II- Lab Statement for the term 2021/2022

Statement: Pilgrimfy: Management of Pilgrim Support during Camino de Santiago

Some former students of Software Engineering 2 have walked the Camino de Santiago this summer. They liked it so much that they have decided to leave their professional careers and dedicate themselves to create a business that allows them to support pilgrims by creating the **Pilgrimfy service**. With Pilgrimfy, pilgrims can have access to various resources needed during the Camino, such as the identification and booking of hostels and albergues, places to eat, places to stamp the pilgrim passport, places to buy souvenirs, stage information, weather forecast, nursing care points to heal wounds ...
This service requires a client / server tool, which should have different categories of client: one for pilgrim users, and another for the owners of different businesses that can offer specific resources to pilgrims, such as hostel owners, restaurant owners, ...

In order to provide better support, it is necessary to understand how Pilgrimfy owners believe pilgrims would use the tool. For example, some prefer to book all hostels well in advance, while others prefer to adapt on a day-to-day basis and stay in the hostels they find available upon arrival at the end of each stage.

As part of the commercial strategy, our friends, the owners of Pilgrimfy, would like to have ready by January 2023 an application with a series of the following minimum services to show them to potential service owners and to a group of potential experienced pilgrims who can validate the tool. There are some aspects that are fundamental, such as using geopositioning services to offer the services closest to the pilgrims' current position, and, above all, being able to exploit social networking capabilities: pilgrims are expected to be able to rate businesses, propose stops with high tourist or religious interest, make recommendations of places where to find specific resources, ...

The requirements they would like to include would be the following for the client application of business owners:

- Add, edit, delete, display information about their business, including the most important details.
- Read and respond to pilgrims' comments.
- Make offers to the most active pilgrims on the network (if they have agreed to privacy requirements that allow them to share specific data so they can be located.

The requirements that they would like to include for the client application for pilgrims are:

- Plan the stages according to the availability of resources and time available to the pilgrim.
- Display information on the different paths
- Search for a specific type of resource close to their current position, or at a specific point of the destination.
- Display information on the current stage and on the remaining stages.
- Create a blog of the route, allowing the sharing of photos, points of interest, ...
- Add comments to the businesses included in the network

For a better management of their business, the owners of Pilgrimfy themselves would also like to have certain information, such as:

- Number of businesses (being able to refine by type and localization) that have been added to Pilgrimfy.
- Number of pilgrims who have used and are using the application
- Number of pilgrims who are happy (at least 4 out 5 stars in their comments) with the different services.
- Analysis of the most important reviews from pilgrims about the businesses visited, in order to be able to promote those with the best results....

As an example, it is suggested to visit some pages of real examples existing in the market such as:

- https://www.pilgrim.es/
- https://santiagoways.com/es/
- ….

Results to be achieved.

The main goal of the lab sessions is **the simulation of the execution of the previously described project within the context.** Consequently, your team is required to face up with decisions which will impact in the final cost of the project, having in mind the limitations you considered when created the company, and using the control, collaboration, and communication tools you chose. However, if for some reason you see that the human and material resources you have established are not sufficient, consider incorporating new resources.

The **Unified Development Process (PUD)** explained in Lesson 1.2 will be used as the development methodology. Make all the assumptions you consider necessary and appropriate but explain and reason them. The development technologies will depend on your choice; in any case, the preferred option is development in **Java 8**.

In the following, it is described what is expected from the working groups. Please have in mind that the results shall be viewed globally, and there will be not partial deliverables within the block (iterative and incremental work).

Block 1

Lesson 1 – Planning and execution of the project by using UDP

When planning the project with the PUD, the parameters that should be obtained are the final cost of the project, the project schedule, and the completion date (which should include a *release* schedule).

For the sake of the simplicity, we propose you to make the following initial assumptions:

- one functional requirement will be mapped to exactly one-use case.
- one use case will be realized into one and only one iteration (1:1);
- For simplicity we will assume that we will have an *Iteration 0* in the inception phase, several iterations corresponding to the realization of the different use cases, and a last *Iteration N*, which would be executed in the transition phase and where tasks related to the closing of the project would be performed, such as integration, integration testing, deployment, user manual, etc. Let's suppose that the cost of the iteration 0 is 1000 € (it would be very interesting to estimate the cost of this iteration really, but for simplicity we will assume that cost), and the cost of the iteration N is 2000 €. Consider that these costs are the ones corresponding to human resources. Discuss if there are any more costs to be included.
- One working week will have 40 workable hours.

In addition, the following aspects should be observed:

- All the planning, as well as all the corresponding documents, minutes of meetings, description of use cases, analysis documents, Visual Paradigm files, time allocation files, ...) **must be linked from the wiki of the GitHub repository[1], so that the project wiki is the guide for reading the documentation**, thus facilitating the correction[2].
- **It is mandatory that all meetings held by the working groups as part of the collaborative work over the time are documented**. All decisions/commitments obtained will be mapped to different *issues* on Github, and progress will be properly monitored by controlling a simplified Kanban board on a project that *will be created ad hoc for practice.* This will be used to mark the traceability of the project, and **there must be coherence between the planning done, the commitments made at the**

[1] Every group's leader must create a repo with the name ISO2-2022-*GroupName.* So, as example, the leader of the A1 group will create the public repo ISO2-2022-A1

[2] Later in this document, we will give you some examples.

meeting, the issues created, and the sequence of project progress (marked by the review of the commitments in the projects). That is, the steps that should be taken are the following as an example:

- In the planning made in the Inception phase a specific milestone is marked (e.g., the class Persona.java v1.3.0 will be created on 28/10[3]
- In the work meeting held on 15/10 is taken and accepted the decision that *Pepito Lopez will develop the class Persona.java v1.3.0*
- The project administrator creates the task in the project "Development of Module X" in which is the task *Creation of the class Persona.java v1.3.0* and will create an issue "*Creation of the class Persona.java v1.3.0*" and it will be assigned to Pepito López to make the corresponding commit around 28/10.
- Pepito Lopez will work from 15/10 (after the coordination meeting) and will make the corresponding *commit* around 28/10 and if the circumstance occurs the corresponding *pull request.*
- In the *Insights* section of Github you can check the progress of the Project
- You must incorporate my user account Ismael.Caballero@uclm.es (IsmaelCaballero) as a contributor to the project so that I can monitor the progress of the state of the practices.

- To better guide the development, it is highly recommended to use the project management functionalities available in Visual Paradigm (see Figure 3). Remember that ***in each iteration you will work exclusively in the case or cases of specified uses[4].*** In addition, and to save costs, the automatic code generation functionality provided by Visual Paradigm.
- Regarding implementation, students must "implement" the necessary modules as to meet all the functional requirements specified, considering the planning done and the working method (iterative, incremental, collaborative).
- Taking into account the 1:1 approach from which we start, it will be considered that each iteration will initially generate one component, whose development must be established in the planning by adequately marking the versions that are expected to be achieved - the versions will be named using Semantic Versioning. At the end of the last iteration, all the components will be integrated. Each of these components will be treated as if it were a module of a Maven project**.** Therefore, the development should be managed as if it were a multimodule project (see chapter 6 of the book "Maven by examples") (http://books.sonatype.com/mvnex-book/reference/index.html). This management can be done from the command line or using the appropriate plugins of the IDE you use (Eclipse recommended, with the plugin m2Eclipse).
- The resulting code will be subject to testing and maintenance (subject to the practices corresponding to topics 4 and 5)
- The wiki must include a section of "Self-evaluation and Experience" where all the members of the group incorporate a self-evaluation and self-criticism of their experience in the project[5].

[3] The plans should be met as far as possible, however, as it has been said in practice classes, it is normal that these dates may vary for various reasons: lack of experience, technological complications, ... all these circumstances are assumed in the assessment, but the groups of students must make a sincere analysis of the reasons why the planning has not been met, and provide means to mitigate the deviations from the agenda, explaining how the problem will be solved to finish the project on time.

[4] In other words, **it is not correct to do all the use cases, or all the class diagrams at once, but when it corresponds to the planning done.**

[5] Self-criticism is not necessarily a negative thing: it can be things like "*I learned a lot*" or "*I consider that the practices have not helped me much*", or "*I think I should have worked more*", or "*now that we have finished the practice, I begin to understand many more things*".

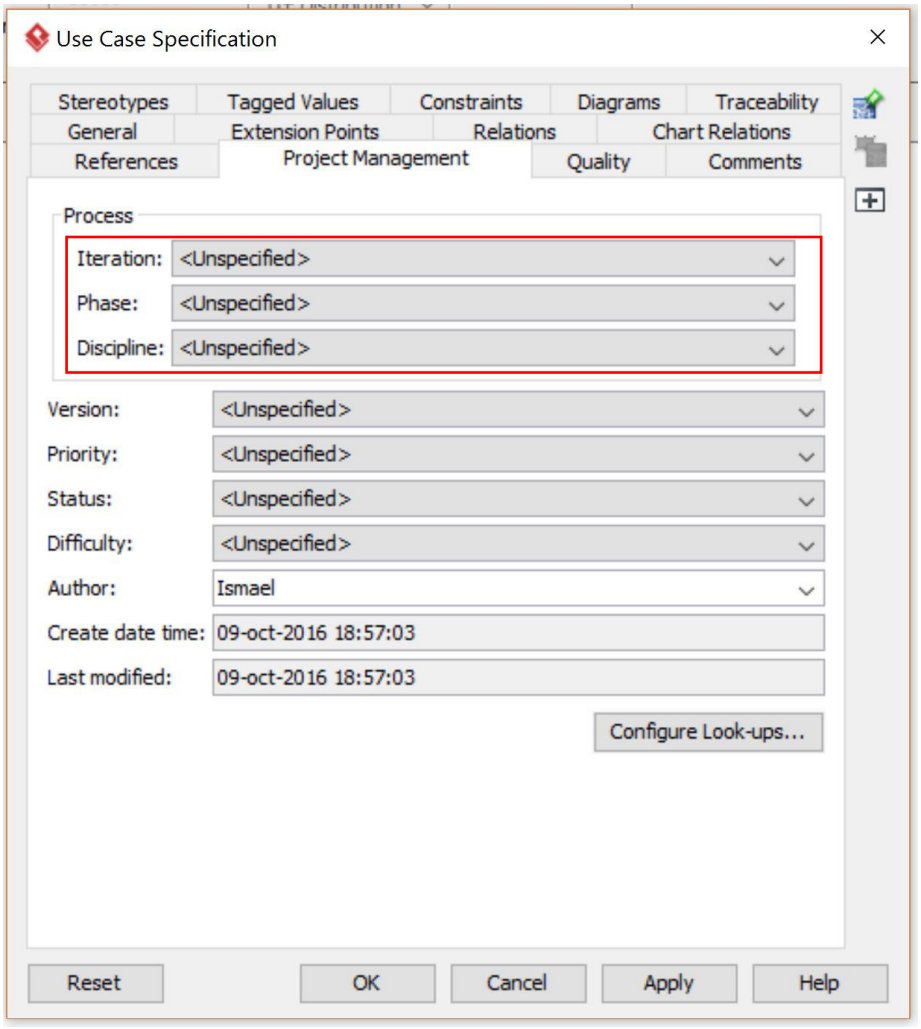

Figure 3: Project management mechanisms in Visual Paradigm

Some important concerns raised from learnt lessons

The experience of the realization of practices of previous terms has allowed us to learn a series of lessons that we transmit to our students:

- **Implementation of the unified development process**: It would be very interesting to read once an again this statement once it has been a while. One of the main problems we find is that students do not understand the mapping of use cases to iterations well. For simplicity, we recommend you do 1RF: 1CDU and 1CDU: 1Iteracion. But this is a recommendation-which somehow reduces the application of the PUD to a cascading cycle-until you really understand that the development of a software should be split into several blocks (which we call *modules*) which must comply with the rule of maximum consistency and minimum coupling (SOLID Principles).
- **Interfaces-oriented programming, not to classes**: As a result of the previous statement, one of the main problems we encounter every year is that students do not have enough programming knowledge to understand the difference between interface-oriented programming and programming Class-oriented. The iterative and incremental nature of the UDP, and the fact of being use case driven, necessarily leads to an approach in which the software must be grouped into several blocks (which we will call **modules**), which must be developed – independently Preferably- in each of the iterations of the PUD application as determined in the planning. This independence

obliges to a minimum coupling between modules that are solved by means of the declaration and use of Interfaces and the application of the multilayer model (MVC pattern – Model-view-controller). Each of the modules requires to make software packages and use the classes properly so that the communication is produced through the corresponding interfaces. So, we encourage you to review the creation of packages and use them adequately in development. To make a more professional programming you though to use some Spring development framework that will simplify the work a lot once you have exceeded the minimum learning curve

- **Use of branches on Github and Maven**. These are the characteristics that most problems usually cause when doing the Lab Session exercise. The problem again is that the students approach the development of the software as a whole, without taking into account the architecture of the system. However, if the students managed to understand that development using iterative and incremental methodologies (regardless of whether they are structured or agile) implies the breakdown of the application in what we have been calling module. It is very important that the modules **don't duplicate the code, they use it as needed.**
- **Addressing small projects to better understand the concepts of Git and Maven**. The main difficulty of the practice is to assume the change of paradigm (Iterative and Incremental and Collaborative). If you add to this, the complexity of tools like Git and Maven, you may find that students feel confused and overwhelmed. For that reason, we encourage students to approach learning about Git, Maven, and other tools outside of the context of the project by applying them to small projects. Once you have learned and matured these concepts, you will be able to apply them to the project.

Lesson 2. Configuration Management

The main goal of this lesson is to implement the Configuration Management Plan (CMP), which will be produced as theoretic exercise. The implementation of the CMP will be gathered in issues like: who approves the requirement, who is responsible for every requirement, versioning plan of the components, and versioning plan of the building.

You shall use the ***Semantic Versioning*** notation (http://semver.org/) and be reflected according to the recommendations indicated in Git Flow, with the creation of the different branches.

Lesson 3. Quality Management.

The objective of this block of practices is the derivation of diverse functional and non-functional requirements that could appear when considering the quality aspects of software products (quality by design).

Block 2

Lesson 4. Testing Management

The main goal for the practices of this topic is to develop and execute the testing plan for the project. In this regard, the following aspects should be considered:

- The testing plan must include testing cases to achieve at least a **coverage level of 75%.**
- The corresponding test report will be generated by integrating the plugins for Surefire and Jacoco in Maven, and results should be also linked in the project wiki. More information will be provided when the test management topic is reached

Lesson 5. Maintenance Management

Taking as a starting point the test report obtained in the previous report, the aim is to correct the errors in the code and or to fix some aspects of clean code. For this, a maintenance plan will be made focused on the system maintenance, which will start with the creation of an automated report generated with Maven by including the different plugins: PMD/Findbugs. More information will be provided when the practices of Topic 5 begins.

Required or Suggested Developing Tools

1) Eclipse with Maven and Git plugings.
2) A MySql server (or virtual machines with the RDBMS deployed and the corresponding firewall ports open).
3) MySQL Workbench.
4) Visual Paradigm

There is a wiki open in the project section so you can link to the github repository of your project.