SOFTWARE ENGINEERING 2

INITIATION PHASE

06.03.2017

Ksawery Jasieński, Filip Matracki, Fardeen Mohammed & Tomasz Marcińczyk

Documentation stages:

1. Software development methodology.
2. Software technology and version.
3. Presentation of a project meeting.
4. Personal work schedule and project schedule.
5. Initial code repository with proper permissions.
6. Specification acceptance or list of its errors and ambiguities.

1. Software development methodology.

The methodology we will be using is going to be Agile because it's not just a methodology or a specific way of developing software or a framework/process but it is also a set of values and principles for better software development by giving a foundation for our team to make efficient decisions.

Following the Agile manifesto we will be valuing:

- Individuals and interactions more than processes and tools
- Working software over excessive documentation
- Responding to changes over a following a strict limiting plan

We think the the Agile methodology will be effective in this software development process because:

- Welcoming changes even in late in-development Agile processes will make it easier for us to accept changes with regards to other teams which is important because not all teams will be following the same development strategy.
- Delivering working software stages frequently will help us track our progress and compare our project with other teams at every level so that we can ensure no major conflicts in the end stages.
- Motivating regular united work and face-to-face conversations among our team will be the most efficient way of conveying information about different parts of the projects assigned to each member and solving each other's problems even other teams if they lack in some stages.
- Primary measure of the progress will be the working software. We are planning for a sustainable development process and a constant pace which will help us compare each stage with other teams and respond to any changes as soon as possible .
- It promotes simplicity and minimizing any unnecessary work so the end stage of our software will be efficient and giving us more window to see our mistakes and fix them in time.
- It promotes a better working environment among the teams thanks to which self organizing teams get the best architecture done.

2. Software technology and version.

We have decided to use the Python programming language for our project. Why is that the case? Python emphasizes code readability and brevity. Many modern languages such as C# and Java have more heavyweight syntax which might distract from the actual functionality of the code. When some implementations of the same procedure in other languages are quite lengthy, the same thing could be done in Python in fewer lines of code. Python developers often strive to make it enjoyable

and “fun” to use, the name of the language itself is a reference to Monty Python. Client and server TCP libraries are one of the fastest and well-optimized ones on the market. Due to the fact that client-server communication is a critical aspect of our project, it is necessary to make sure that we limit potential overhead. It can also be seen as a learning experience for our group, as not everyone in the group is entirely proficient in the language. In our future a language as common as Python is unavoidable in the work environment.

For example, Python uses newline to delimit lines of code instead of semicolons, and whitespace indentation in order to delimit blocks of code. Therefore the language itself enforces neat and clean-looking code. It is readable in a similar way to reading a sentence in the English language, making it obvious what is happening behind the scenes of the code. Python also has few syntactic exceptions and special cases compared to traditional C-style languages. The developer community is vast with endless documentation and tutorials existing in the internet. Python has also been ranked as within the top 10 most used programming languages according to the TIOBE Programming Community Index.

In this specific project Python will be quite valuable for us for multiple reasons. Python can be interpreted on the fly line by line and executed in the command line environment, making it faster and easier for us to test functions and procedures in our program. One can easily add command line arguments in order to customize and configure the server and game master. By executing selected fragments of the code, it will make debugging a lot easier as well.

For the IDE we have decided to use *PyCharm Professional* by JetBrains. JetBrains is a leader in the IDE world, developing hit products such as *IntelliJ* for Java. JetBrains offers their professional products for free for students, normally one would have to pay hundreds of dollars for access. Apart from the financial benefits, PyCharm offers many features including code-completion, integrated Python debugger, integrated unit testing (very important for this specific project), Google App Engine, and built in GitHub support. Pulling and pushing code changes to the remote repository will be simplified and could be done straight from the GUI of PyCharm.

3. Presentation of a project meeting.

One of the most important aspects of the Agile methodology is the iterative and flexible solving of the arising problems and execution of the project goals. For this, we need to ensure proper communication between team members in order to keep everyone in the know of the current status of the project: this includes information on what work has already been done, on what there’s still left to do and on any possible problems that might interfere with the execution. These points will be summarised and discussed during every project meeting, in order to best coordinate the work done by the team. These are also an opportunity to address the division of work between team members, to ensure that it’s fairly split.

To start off a meeting, each team member should briefly summarise what part of the project they managed to complete since last meeting. This is also an opportunity to review, ask questions and discuss the code provided by other team members, to ensure that there are no ambiguities with regard to the functioning of the system, and that all the written code is compatible and of high quality. In general, the most important part of any team meeting will be to discuss and gather feedback.

Further on, the meeting should concern the upcoming tasks which will have to be completed by the next meeting. Tasks should be assigned priorities to provide transparency on what is the most important short-term goal to accomplish. Team members should select tasks for themselves basing on their own preferences and abilities.

Each project meeting will be properly documented. This will allow for the progress done on the project to be monitored. Proper documentation will also make it easier for team members to keep track of who's working on what part of the project.

Project meeting documentation will include:

- Listing what parts of the project have been finished since the last team meeting - this includes information on what code has been written, by whom, with necessary comments: such as issues that arose during work on the respective parts
- Information about the parts of the system the team was working on during the project meeting, precisely documenting who's working on what part of the code
- List of tasks which have been assigned to team members to be finished before the next project meeting
- At the end of the meeting, some brief information about the current state of the project should be provided - what parts still remain to be finished etc.

4. Personal work schedule and general project schedule.

Project schedule:

With accordance to the Agile methodology, the project will be produced in roughly two-week *sprints*, whose scheduling will overall be in line with the deadlines set by the Product Owner (in this case, the coordinator of the course). Members of the team will work on their respective parts of the project by themselves, and also during planned project meetings. These will be held during the lab sessions scheduled in the course, with possibly more meetings being organized in between them if necessary. At all times communication will also be maintained through communicators.

Considering the current lack of details about the structure of the project, the general project schedule will be, as mentioned, consistent with the deadlines set up by the course coordinator.

Week 2	Communication phase: implementation
Weeks 3, 4	Communication phase: testing and bugfixes
Week 5	Game phase: implementation
Weeks 6, 7	Game phase: testing and bugfixes
Weeks 8,9	Cooperation phase: implementation
Weeks 10,11	Cooperation phase: testing and bugfixes
Week 12	Optimalization

This proposed project schedule should ensure that the appropriate modules of the system will be working correctly by the deadlines. In order to achieve this, more emphasis was put on phases dedicated to testing and bugfixes of the appropriate modules.

To provide more clarification on the specific phases of the project:

- Communication phase - building and testing of the communication protocol which will allow for the messages to be passed between the Players, the Communication Server, and the Game Master
- Game phase - implementing the logic of the game. The features of the game to be provided will be obtained through the passing and validating of the appropriate messages between the agents in the system.
- Cooperation phase - ensuring the correct behaviour of the Players with respect to one another through the use of communication, designing and implementing a strategy which will best take advantage of the system's specification in order to win.
- The last week of the project will be devoted to some final tweaks and optimization of the program, but in principle the system should already be working properly.

Personal work schedule:

The Agile methodology assumes flexible splitting of work, with respect to team members' preferences and their personal skills. This becomes especially relevant in the context of this project, where the exact details of what's to implement are not entirely known beforehand. We will thus strive to be able to adapt to new challenges on the fly through flexibility and cooperation.

5. Initial code repository with proper permissions.

Repository will consist of main 5 folders:

- **src,** which will be main development code repository.
- **test**, which will cover all test files and results
- **doc** - project documentation files
- **dist** - compiled and working project files

- **lib** - will include all external libraries and plugins.

6. Issues with specification

The following is a list of all the issues with the specification raised by our team through the GitLab server:

- https://se2.mini.pw.edu.pl/17-results/17-results/issues/50
- https://se2.mini.pw.edu.pl/17-results/17-results/issues/49
- https://se2.mini.pw.edu.pl/17-results/17-results/issues/48
- https://se2.mini.pw.edu.pl/17-results/17-results/issues/45
- https://se2.mini.pw.edu.pl/17-results/17-results/issues/44
- https://se2.mini.pw.edu.pl/17-results/17-results/issues/43
- https://se2.mini.pw.edu.pl/17-results/17-results/issues/42
- https://se2.mini.pw.edu.pl/17-results/17-results/issues/41
- https://se2.mini.pw.edu.pl/17-results/17-results/issues/51