# Project The Game

Initiation part

Aleksander Kuśmierczyk
Kornel Żaba
Mikołaj Słoń
Mateusz Szperna
Alexey Zhavrid

# Introduction

As the method of developing of this project we have chosen the **EXTREME Programming** approach. The reason behind this choice was that this way minimizes the documentation and design work in order to focus on the programming part. Additionally we have already worked in this group and we managed to test out programming together.

The possible advantages (apart from the ones already mentioned) of this development model are the: comfort of working, high reliability and quality of developed software. On the other hand we are aware of the possible drawbacks which we will have to tackle: the messiness of the project and the feeling of being observed.

## Technology

The technology we chose for the project was **C# .NET 4.6.1**. Our choice was based on the familiarity of the whole group.

## Specification acceptance

We agree with the specification having in mind that all of the issues mentioned on the gitlab (among others our issue was: https://se2.mini.pw.edu.pl/17-results/17-results/issues/46 we also agree with the rest of the students) will be resolved by the time we will start working on the particular parts of the project.

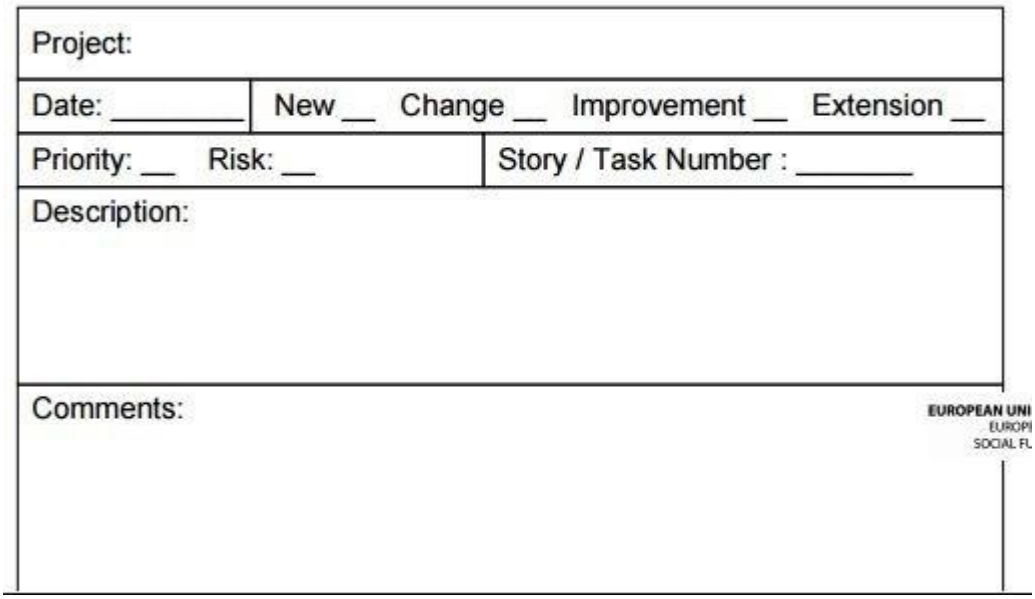
## Description of our approach

In the following section we will list all of the basic rules of the EXTREME Programming and their description with the possible changes to adjust it to our group.

All of the points with the * at the end are references to the lecture given by dr inż. Krzysztof Kaczmarski during the previous semester as the preparation for this project.

1) Planning*

- *Client creates a list of wishes, ordered on importance, similar to CRC cards.*
- *Developer estimates costs of a single part and how many parts can be done in certain time.*
- *Client selects desired functions to be developed first and eliminates less important wishes.*




We are considering dr inż. Michał Okulewicz as the client and we will refer to him as such during the rest of this document. Additionally, as “list of wishes” we will be considering the specification given to us by the client. Order of importance as well as the “desired functions to be developed first” we will be considering the requirements for the deadlines and their specific content given at the first lecture.

2) Small releases*

- *We always start from a small subset of functionalities of the system to get first working program as soon as possible.*
- *In each iteration a small part of a new system is added.*

3) System's metaphor*

- *A project should have a metaphor and easy names assuring understanding of the system and simplifying communication.*

We consider the game concept given by the specification as the "System's metaphor" which goes as follows:

"The idea of The Project Game is to simulate competitive project development by two teams of players. The game simulates the following properties of projects development: (1) tasks are connected with risks (usually negative), (2) goals of the project are unclear (and need to be discovered), and (3) communication between members helps to speed up the process of the development."

4) Simple design*

- *Always the easiest and simplest way to a goal is chosen, but assuring that the system will do what is intended.*
- *Requirements are to be changed sooner or later so let's do what is needed today.*
- *YAGNI – You Are not Gonna Need It!*

We will be uploading the parts of design directly to our Gitlab repository.

5) Continuous testing*

- *Before a new function is added a programmer writes a test procedure for it.*
- *Tests are divided into two classes: module tests and acceptance tests.*

6) Refactoring*

- *Any duplicated code must be refactored.*
- *Refactorings are accepted only in 100% assurance that nothing would be broken.*

7) Work in pairs*

- *Code is written in pairs.*
- *Whole code is checked during writing.*
- *Pairs are changed everyday.*
- *Knowledge about certain solutions is propagated to the whole team.*

As we are not professional programmers and we are not working in the same workspace we are unable to meet every day, therefore we agreed on meeting 3 days a week for 2-3 hour sessions. Additionally we will change the pairs less frequently than once a day.

8) Collective ownership*

- *Each programmer should be able to work with any part of code anytime.*

9) Continuous integration*

- *Each new part of code is integrated with the whole system everyday.*
- *All tests must work properly before and after the integration.*

As we have explained earlier we won't stick to the everyday work plan but we will adjust this rule to fit to the aforementioned schedule.

10) 40-hour week*

- *All programmers go home on time.*
- *In the worst possible scenario one week of overloading is acceptable.*
- *If it happens more often then the project is in trouble.*

As our schedule differs from the one explained during the lecture we will stick to the fewer number of hours.

11) Continuous client interaction*

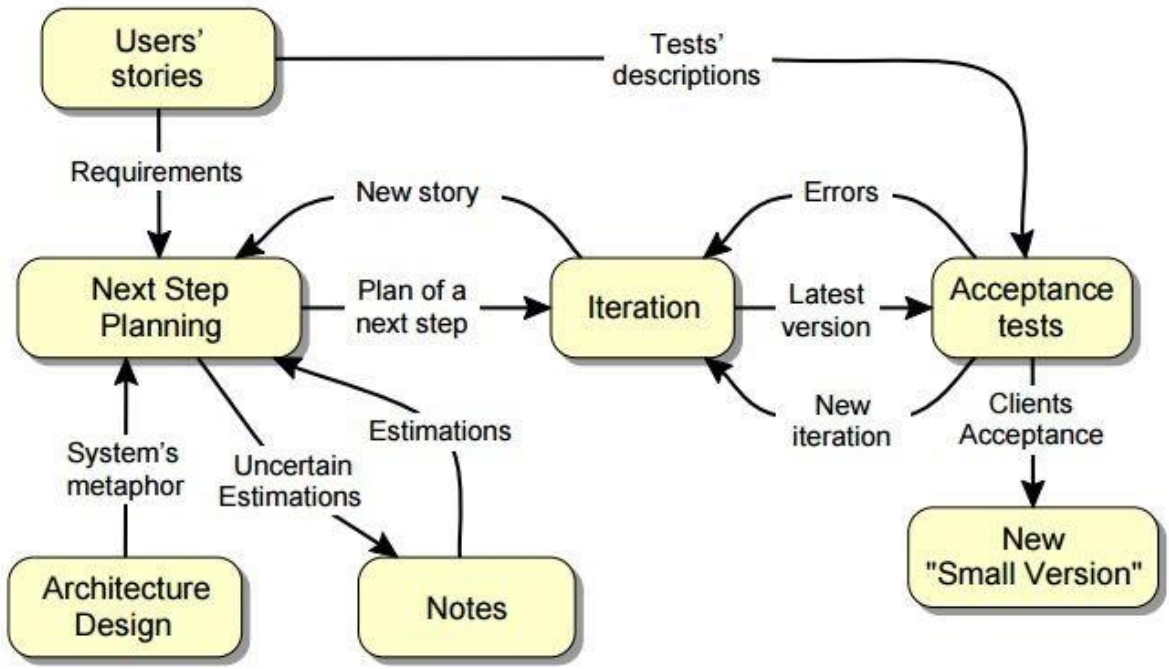
- *Developer is in continuous contact with the client.*
- *Client is responsible for acceptance testing, preparing new or modifying old story cards.*

We will consider continuous contact with the client as the form of conversation over the e-mails sent between our group leader and the client. Additionally we will have meeting with the client each Friday during the labs.

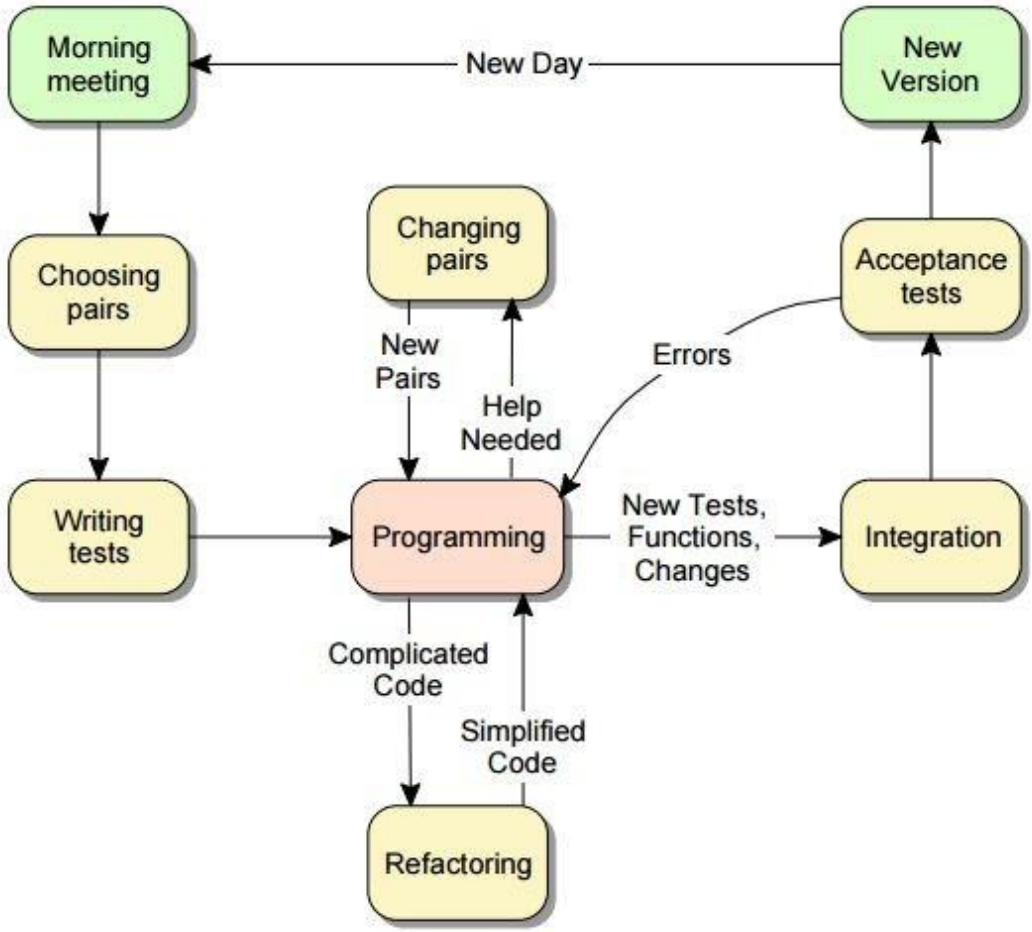
12) Coding standards*

- *All programmers use the same coding standard.*
- *Looking at the code there is no way to recognize who wrote it.*
- *These standards should be as simple as possible and accepted by a whole team.*

General idea of EXTREME Programming



A chart representing the meeting session between programmers



# Preview of the wish list for the next release (deadline).

| No | Priority | Description | Estimated Cost (in the % of time spent during development) |
|---|---|---|---|
| 1) | high | A working prototype is delivered on a testing week | 30 |
| 2) | high | Messages are passed between distributed nodes and server | 20 |
| 3) | high | The system is not interrupted by failure of a node | 25 |
| 4) | high | The system has proper and complete integration and unit tests | 15 |
| 5) | medium | Mock player and game master provide a flow of mock board data and random moves and actions | 5 |
| 6) | medium | Server prints the state of the system | 2.5 |
| 7) | medium | Components can be run in verbose mode printing out messages and/or types of messages | 2.5 |

We will post each wish list on our Gitlab repository at the beginning of each release period.