Nominal Workbench – Methodology

No-Work Dev. team

10 décembre 2013

1 Overview

In this document, we will describe the methodology that we apply for the development of the project Nominal-Workbench. This description will follow three main axis which are :

- Agile methodology
- The tools used
- The interactions with the client

2 Agile methodology

In this project, we chose to apply an agile methodology for an efficient development. We had two options in mind, which were Scrum or Extreme Programming.

We finally decided to choose the Scrum methodology for many reasons : first of all, the development in Scrum can be done by each developer independantly. In Extreme Programming, dev team has to be in a single room and sometimes has to work by pair of developers. More and over, the interaction with the client is simpler with Scrum. The client has to talk mainly with the customer relations manager to follow the evolution of the project while, in Extreme Programming, he has to interact with every developer to stay aware.

Meetings take place weekly in order to answer those three questions :

- What have I done ?
- What is my next work ?
- What difficulties have been encountered ?

We had to adapt the Scrum methodology to match with our schedule and our disponibility. Therefore, we defined the Sprints duration to be of one week instead of a month.

3 Tools

All along this project, we will use some tools to improve our efficiency.

3.1 Rallydev

Rallydev is a Scrum platform on which we can add, describe and edit user stories. We can also assign a developer with a task and see the progression of the development. A “story” can be either defined, in progress, completed or accepted.

3.2 Git

Git is a distributed revision control and source code management system. We decided to use the web-based hosting service : GitHub. This choice was motivated by the fact that both the client and the development team were already familiar with this kind of repository system.

GitHub also provides the possibility to create wikis and to easily report issues thus enhancing the communication between the team members.

The project repository is forked from the client's one and will be asked to be pulled upstream after each "sprint" iteration.

3.3 Building system

There were several building systems we could use. We decided that classical Makefile would not suffice as the scalability was difficult to maintain as the project would grow. After consideration, we finally decided to use *ocp-build* which allows the project to be well-organized, to spend less time resolving dependencies issues and to have an easy way for everyone to include the new tests and libraries without a deep knowledge about the building system.

4 Client interactions

Once every two weeks, or on demand, the client will be able to discuss with some of the team members. During these meetings, a full report of the advancement will be presented to the client. Afterwards, the client reacts if any concern is raised about the work achieved.

If necessary, there will be also a discussion about development issues. It may be about technological choices or implementation decisions.

We will conclude the meeting with an overview of the future work that will be done until the next one. The client will then be able to request any new features he'd like to add into the project.

If there is an urging matter between two meeting, such as a bug related issue or an important feature to quickly add then the project manager will adapt the current sprint and include the new stories with a suitable priority. Such an event may be signaled either by a mail or directly on GitHub depending on the client preferences.