Title

Author

Affiliation

email

December 1, 2014

Abstract

NEREID is a bundle of projects shared between TUM and INSA Lyon.

Our project is a bit peculiar. We are meant to improve Evince, the default PDF reader on Linux (that also works on Windows).

With the initial project, we intended to improve annotations, mainly addind embedded highlighting. This was already done by someone at *Google Summer of code*, therefore we decided to work on hypertext link navigation. We are strongly commited to having the features be accepted by the community.

Being an open-source project, it will be a great experience, which will get a better understanding of the open-source ecosystem. The project also follows up our engineering courses, at TUM as well as at INSA Lyon. Moreover, the code itself is very advanced and will enhance our programming skills.

1 Story Line

Every week we hold a meeting on "Skype"; in this meeting we distribute new tasks, we process the quality of what we've done during the prevoius week and share our new findings about each part of the project with other members.

We use "Trello" as a collaboration tool and task distributer that organizes our project into cards. It provides us an overview on what's being worked on, who's working on what, and where something is in a process.

The first step for all of us was compiling the project on our systems. This part was a bit challenging as it requires many libraries and dependencies. Then we tried to get familiar with the code. The next step was one of the most important parts, to choose useful and inovative features in order to implement. All team members searched and studied about it and finally we asked the community to confirm the most relevant and we started to distrubte the tasks for that specific feature.

2 Project Planning and Management

Complex projects usually involve highly qualified specialist project managers. We are trying following tools for our project management.

1. Work Breakdown Structure: We planned start by breaking down the overall objective into smaller packages, until each parcel of work is self-contained and can be attributed to one or more than one team members.

2. Dependencies: We list the tasks and which depend on the completion of which before they can begin. At this point we can also estimate how long each will take. This kind of analysis allows projects to be planned to make

the most effective use of time and resources, and to deliver key outcomes and milestones to a specific timetable.

3 Quality Process

Every software development needs to be focused on the delivery of quality. One of the most important customer service skills is the ability to understand and effectively respond to the customers needs and concerns. Quality process must be a part of the entire software development life cycle (SDLC) from inception through implementation.

We see quality process fundamentally as a way of solving problems. Once we sense a problem, good problem solving technique involves alternating between the levels of thought and experience. For instance, after we sense a problem, we should collect some data to get insight regarding the area of the problem, choose the specific relevant improvement activity we will undertake, collect some more data, analyze the data to find the causes of the problem, plan a solution and try it, collect some more data to evaluate the effects of the new solution, standardize on the solution if it works, and conclude by reflecting on what we did.

We use the following steps to ensure that we measure the right quality-control factors in the right way.

- Determine what to measure.
- Determine our measurement process by selecting the best process for our needs.
- Define exactly how we'll use the selected measurement process.
- Perform the measurements and compare to customer specifications.
- Confirm the quality of our data with compare-and-review checks and the help of a computer.
- Make sense of our data with coding and different data charts.

4 Risk Management

4.1 Risk Factors

Technical issues The specificity of the field of study requires strong specialization of the team members.

Degree of integration and project size The project is ambitious. Its scale suggests a modular architecture resulting in strong dependencies and interactions.

It will be necessary to be rigorous and consistent to face them, so we decided to use Agile development methods (such as Scrum).

Team Instability Mutual emulation is necessary to prevent any down-spirit in the team.

4.1.1 Organizational Configuration

Groupwares Establishment of standards, styleguide, graphic charter and collaborative tools. We define classic workflow of the production cycle and the validation of a deliverable.

Maturity It is necessary to set up a regular monitoring planning and a wide frame. The project managers and quality assessers should regularly monitor compliance with deadlines and anticipate the next ones.

4.2 Project Risks

Financial risks In this case, this is an analogy for the amount of work. It is the responsibility of the project leaders to carry out a strict planning structured by a regular monitoring of the work on each task. It will be necessary to provide meaningful indicators of these aspects on project monitoring dashboards.

Human risks Possible cases of incompetence will be notified by the allocation of technology watch slots, and the use of systematic mutual assistance.

Technological risks To prevent the loss of documents, a Version Manager will be used consistently (Google Drive and Git).

5 Task weight

Technical specification	12h
Implement software tests	8h
Compile up-to-date Evince code	6h
Call another app from the code (Linux)	4h
Find out a link is a link	4h
Integrate the POCs into Evince last up-to-date code	4h
Make a link clickable	4h
Cross-platform investigation	4h
Clear Git	3h
Description of every task	2h
Project management documentation	2h
Stakes of international work	2h
Check out the validation process	2h
Preparing Slides	2h
Check out the test process	2h
Introduction and Storyline	1h
Check dependencies between tasks	1h

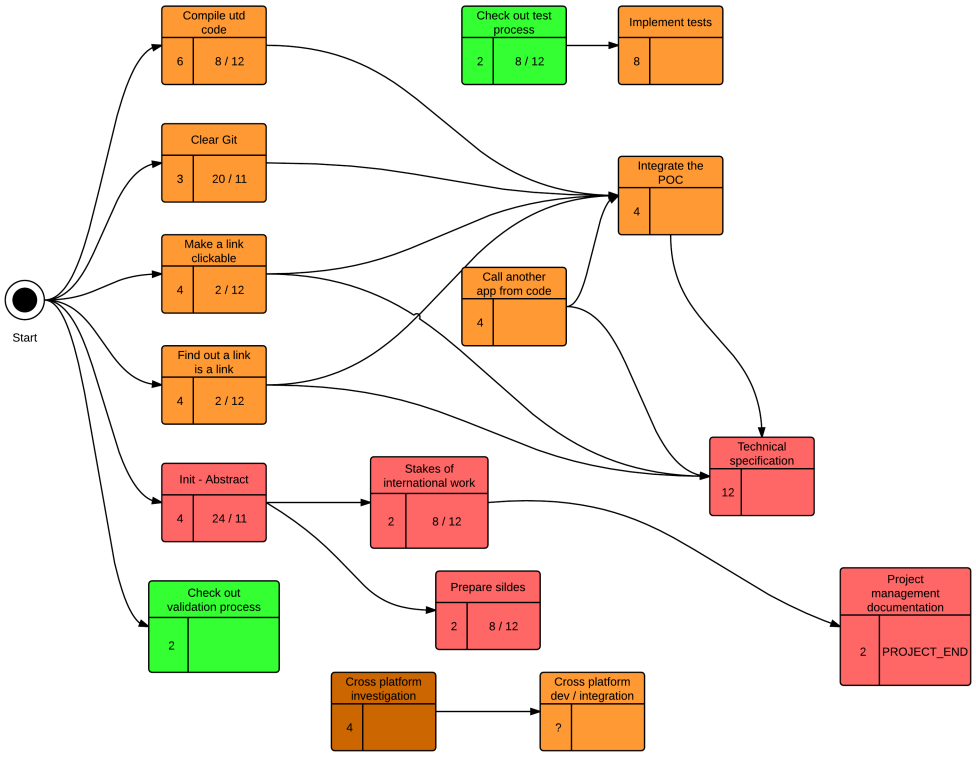

Figure 1: Previsional organization

6 Task Distribution

Table 1: Implementation tasks

Tasks	Adrian Delmarre	Shabnam Najafian	Quentin Coursodon	Mohammad Mahadi Hasan
Call another app from the code (Linux)		√	√	
Integrate the POCs into Evince last up-to-date code	√		√	
Implement software tests	√			√
Find out a link is a link			√	√
Make a link clickable	√	√		
Compile up-to-date Evince code	√	√		

Table 2: Documentation tasks

Tasks	Adrian Delmarre	Shabnam Najafian	Quentin Coursodon	Mohammad Mahadi Hasan
Discribtion of every task	√	√	√	√
Introduction and Story line	√	√		
Check dependencies between tasks				√
Technical specification	√	√	√	√
Project management documentation	√	√		
Stakes of international work	√	√	√	√
Preparing Slides	√	√	√	√

Table 3: Architecture tasks

Tasks	Adrian Delmarre	Shabnam Najafian	Quentin Coursodon	Mohammad Mahadi Hasan
Technical specification	d	d	d	d
Cross-platform investigation	d			
Clear Git			d	

Table 4: Validation and Community tasks

Tasks	Adrian Delmarre	Shabnam Najafian	Quentin Coursodon	Mohammad Mahadi Hasan
Check out the validation process	d	d		d
Check out the test process		d		d