TDT4705 Systemutvikling, fordypning - Høsten 2005

Open Source - Artistic Software

Thibault Collet

Maximo Ramirez

NTNU

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE,
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING.

Abstract

Nowadays there is two new movements that are taking place in the society, concretely into the computer science community: Artistic Software and Open Source Software. Net art is still young but it is transferring its ideals to the Software. One example is the quick but intense interest for GUI that almost all the software get.

OSS is rising, even in companies. Its ways to develop software have faced up to commercial software, thus his processes are reused in the industry (eXtrem programming). OSS philosophy takes good thing between the individual work and the group work and it includes the final users in the development.

Net Art is changing the concept of contemporary Art. The individual artist is becoming more than one. Now we can find Art communities. Philosophy of Net Art is quite similar to OSS philosophy. Freedom, sharing of ideas, community, etc. are the milestones of OSS.

The question now is why not mixing Artistic Software and Open Source Software?

Acknowledgements

This work was written as a part of our TDT4735 course at the Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU), during the autumn of 2006.

We would like to thank Øyvind Brandtsegg, at the Department of Music, for letting us work with the extraordinary project ImproSculpt and making it an OSS project and providing us with advice and personal expertise. We would also like to thank to Letizia Jaccheri (our supervisor), at the Department of Computer and Information Science, for be really patient with us and for the OSS and Artistic courses which have been a very good background for our project.

I – Introduction

I.1 - Motivation

Is it possible to produce art with computers? How developers can be introduced in the artistic field? What is the kind of relationship between artists and computer scientists? How to manage it? What are the benefits for each one to work with the other? So many questions we asked ourselves before starting this project. Both of us are attracted by art, and this project was a good opportunity to have an overview of artistic issues in computer science.

Apart from that, for few years Open Source raised to access today a level that futures engineers cannot skip. It is playing an active part into industry and when a project manager has to make a choice between Open Source or Commercial Out-of-the-Shell tools for development he should be aware of the advantages and inconvenient of each one. In the case he chooses to deal with Open Source, the licenses can cause several troubles; he also has to know how to deal with the community which is the "support" of the tool.

Open Source offers a completely different way of programming compared to company development models. "Freedom" may be the word which represent it the best (no dead lines, the programmer choose the part of the software he would like to work on…). As art is free expression, can Open Source be the artistic evolution of computer programming? This intersection between both fields looked interesting and deserved from us to focus one semester on it.

I.2 - Research Context

"Candidates will have to participate to one open source artistic software project in order to learn about open source and software art, improve technical skills, and contribute to the discussion around open source and software art. The candidate can influence the research focus, as well as the project to work with. The research questions will have to be grounded in the literature."

The project stakeholders are our supervisor, Letizia Jaccheri, our client and author of ImproSculpt, Øyvind Brandtsegg, and the researchers and authors of this thesis, Thibault Collet and Maximo Ramirez.

As we are exchange students in Norway for one year, a brief description of our previous studies and background in our own countries is necessary for a better understanding of this report.

I.3 - Previous Studies

"*EFREI (French private engineering school of information technologies and management) is located near Paris. I'm actually in the 5th and last year of my studies. During the two first years, the program was focused on scientifically background like mathematical, applied physiques as also basics of programming and electronics. The third and forth years, the specialisation in Information Technologies offered me an opportunity to improve my technical skills as also study deeply project management. A manager should be aware of Open Source opportunities and dangers. This is not taught in my school. As we said before, I have always been attracted and interested by Art. This project suited perfectly with the lacks of knowledge I needed to fill in as my own interest."*

Thibault Collet

"UPM (Polytechnic University of Madrid) is one of the Madrid Universities which offer computer science study program. Now I'm in my 5th year of my studies. The first two years, planning includes courses related with mathematics, physiques, electronics and programming. In the third year I have got knowledge about computer architectures, I got knowledge in the whole hardware area. And finally, the last year was the first year that I had a Software Engineer course. This year I got modelling skills also because I attended to Artificial Intelligence, Data Base and Operative System Design courses. For me, this project gives me management knowledge that I have not got before and another vision of Software Engineer and its processes in the OSS framework."

Maximo Ramirez Robles

I.4 - Problem Definition

Research Question

What is artistic software? The exact meaning is not clear. Where is the situation of the "art"? Two definitions could fit to the "artistic software" appellation.

On one hand it can be software realized by artists where aesthetic material is code and whose expressive form is software programming.

On the other hand this name is used for software designed for artists. In this situation, the software is only a tool used by the artist like a pencil and paint used by a painter.

But the intersection of both can sometimes be blurry. It is the case of ImproSculpt, software realized by an artist for an artist. It is a piece of art in itself because it can produce music thanks to algorithmic composition, but is also used as a tool in stage by Øyvind. ImproSculpt match both definitions.

Open source philosophy takes a place more and more important in software development. This new approach of producing source code has a strong influence on artistic software thanks to freedom impossible to find in commercial solutions.

In this research study we would like to investigate the position of open source in each case of artistic software. Thus this study follows the question:

"What happens when open source meets artistic software?"

Answering this question will help to better understand what OS can bring to artistic software and more especially to ImproSculpt. On another part, these investigations will give good advice on "How to apply the OS philosophy to artistic software?"

Practical Goal

ImproSculpt has become important Software which only one developer has difficulties to improve. Thus, to recruit people to help, to improve, to give ideas from others points of view, to find bugs using ImproSculpt, etc. is nowadays a considerable task to carry out. Especially for Øyvind who needs time to focus on the composition work. Naturally, open the source was a solution to most part of these problems.

Although ImproSculpt was declared Open Source Software by Øyvind few years ago (when the source code has been accessible on internet), a lot of management tasks are required before call it Open Source "*Project*".

Launching the open source project "ImproSculpt" (as our practical goal) will give us the knowledge necessary for answering the latest questions.

II – Prestudy

II.1 - Open Source: Presentation

"Open source describes practices in production and development that promote access to the end product's source materials—typically, their source code. Some consider it as a philosophy, and others consider it as a pragmatic methodology."

From the Wikipedia[1]

The history of Open Source begins with the Free Software Foundation (FSF)[2], which was established in 1985. Free refers originally to freedom and no to "no-cost". The pioneer of this organization, Richard Stallman, started his endeavours after commercial software spread. His intention was idealistic, to bring the freedom to modify and copy back to software. In 1998, the term "Open Source" was introduced. This coincided with the source code release of the web browser Netscape Navigator. Since then, it is widely accepted that both terms refer to the same concept, but still have different values. Open Source/free software has come a long way since the inception of the idea. From being just a way of making software available to users, it has developed into a development method that complements, or even competes with commercial development. More and more hybrid solutions are proposed by companies, and Open Source became today a face of software development we cannot skip.

The basic Open Source software philosophy is extremely simple: when programmers are allowed to work freely on the source code of a programme, this will inevitably be improved because collaboration helps to correct errors and enables adaptation to different needs and hardware platforms.

This has in fact happened and Open Source software is well known today for its high degree of reliability and portability.

The competition posed by Open Source to commercial software has impacted the industry as a whole. Consumers don't need to buy expensive solutions for operating a computer, word processing, Internet browsing, etc.

II.2 - State-of-the-art: ImproSculpt

Birth of ImproSculpt

Øyvind Brandtsegg[4], originally from a jazz improvisation background, has always had a strong desire to experiment and discover new ways of making music. This extended to experimenting with all kinds of synthesizers and music hardware available to him.

Always looking for new ways to improvise in musical contexts, Brandtsegg picked up a piece of Macintosh software in 1996 called “Max”. This provided a graphical programming interface, which could output MIDI signals to an acoustic piano for playback. In the years that followed, Brandtsegg wanted to accomplish something similar with audio instead of MIDI, but the available computer hardware just wasn’t powerful enough for this task at the time.

In 2000, Brandtsegg started working on the original ImproSculpt. It was developed in part as a compositional tool for a commissioned work from a choir, which put him in a position to realize some of the ideas he carried with him. Computers were now becoming powerful enough to handle a lot of the tasks Brandtsegg had visualized in software, and 1-2 years later, the choir work was performed with the first version of ImproSculpt. Originally created as a tool for himself, Brandtsegg quickly discovered that his ideas sparked a lot of interest from other composers and music technology enthusiasts, and it was published as an open-source distribution. At the time of writing, ImproSculpt has been used across the globe as both a compositional tool and a live performance instrument. As the author puts it; composing music with ImproSculpt is a kind of live performance in itself.

Students start to help Øyvind

Øyvind is, before all, an artist (even if he showed impressive programming skills with the development of ImproSculpt). As the original ImproSculpt project matured, more features were continuously added, and the source code eventually grew large and unruly. Brandtsegg discovered that the task of adding new features became increasingly strenuous, as virtually the entire system was affected by even minor adjustments to the source code.

Eventually frustrated with the amounts of "collateral" work associated with further development of the system, Brandtsegg decided to give up developing the old source code any further, and rather start from scratch with a new and more structured approach.

Thanks to education and experience within the field of software engineering their intent was to help Brandtsegg avoid the same pitfalls in the new version of ImproSculpt.

Flyndre

Flyndre [6] is an application of ImproSculpt. This work of sound has been built over an existing sculpture by the sculptor Nils Aas, which carries the same title, and which is located at Straumen in Inderøy, Norway. Based on environmental parameters such as temperature, percentage of luminosity, level of water, etc, ImproSculpt generate a music reflecting the nature around the sculpture. The sensor data and the sound signals are streamed via the Internet between Inderøy and Trondheim where the computer which calculates the sound is located. The so generate music is sent back to the sculpture. The installation makes use of a loudspeaker technique in which the sound is transferred to the metal in the sculpture. And the purpose of the project is that the installation is supposed to run continuously during 10 years (from 2006 until 2016)

III - Problem Elaboration

ImproSculpt was created for personal use of its creator: Øyvind Brandtsegg. He quickly discovered that ImproSculpt was a useful and interesting tool for other composers and music technology enthusiasts on the globe. For this reason, Brandtsegg decided to declare ImproSculpt as an open-source distribution. Thus, he uploaded the ImproSculpt's source code on the net. But, as Karl Fogel wrote [7], to consider Software as an Open Source Project, ImproSculpt would need more than only publishing the source code on internet. ImproSculpt would need a work environment with technologies "*that support the selective capture and integration of information"* [7]. With this kind of technologies, ImproSculpt would become a successful project which persuades others to work in it.

As Fred Brooks observed [8], the complexity of a project increases as the square of the number of participants of the project. The main weapon to fight with the work-in-group problems is to make easy the communication and coordination in the work group.

Another issue added to ImproSculpt development is that it is an Artistic Software. I.e. we must bear the communication between artist and computer scientist in mind. Thus, these communication technologies should be easy to use.

As we said before, our project lies in launching an open source project from the ImproSculpt bundle (Flyndra, ImproSculpt Classic and Unstable ImproSculpt). That is, we have needed to carry out some launch required task in order to provide ImproSculpt with work architecture. The next section provides you a launch required task overview.

III.1 - OSS Launch Required Tasks

Launching an Open Source Project based on the existing source code of ImproSculpt requires, as Karl Fogel researched [7], the following technologies:

- *Web Site*: One centralized way to show the audience information about the project.
- *Control Version System*: This allows developers to manage the code versions and to work in the same files.
- *Mailing List*: It is the most active way of communication.
- *Bug Tracking*: Enables developers to follow the evolution of they are working, coordinating themselves and planning releases. Enables the ability of record bugs and follow its life. Bug tracking is not a mechanism just to manage bugs, but also tasks, releases, new features, etc.
- *Real-time chat*: For the real communication, when somebody needs real-time help or advice.

Nowadays we can find on internet web sites exclusively aimed to OSS (Open Source Software) projects. These web sites offer to project administrators all the technologies needed to set up a work environment as Karl Fogel described in his book. In order to understand the features offered by these web sites we are going to split the work environment in two layers.

Calling *the physical layer* and *the logical layer* to the two layers needed to establish a successful work environment on which developers are able to work efficiently.

As we have said above, an OSS project needs some technologies in order to allow communication and work coordination among developers. For example, one of these technologies is a web site, and all the websites need a real place where they can be located. This is the web server and this technology is offered by the physical layer. Thus, the web site will be offered by the logical layer and it will need the corresponding technology in the physical layer.

Web sites, as SourceForge.net [9], offer to project administrator all of these physical technologies and some of them, even some of the logical technologies.

Being based on the offered features by SourceForge.net, we have needed to set up and configure some tools in the logical layer using technologies supported by SourceForge.net in the physical layer. Thus, the first task that we had to performance was to research how the SorceForge.net work environment is. When we understood how SorceForge.net works and which features offers we were able to clarify and write our work.

The following is an informal list of the task that we had to performance in order to set up a new work environment where new developers will be able to work with ImproSculpt:

- First of all, we need to start a new project named ImproSculpt on SourceForge.net website
- As everybody will have access to the source code, this one should be protected under a license [10]. The source code should include basics terms of the license.
- In order to manage the different versions of the source code, a CVS environment [11] will be used. It has to be set up.
- All the developers should have access to basics knowledge related with ImproSculpt, and be able to improve it. This will be done thanks to a Wiki [12].

Basically, we can divide the work in three kinds of tasks: CVS, Wiki and Documentation. CVS tasks will provide us coordination support for developers who work on the code versions of ImproSculpt. The centralized way to show the audience information about ImproSculpt, in our case, is the Wiki. Finally, we can also divide the documentation part in two sets of tasks, SourceForge.net and Wiki. SourceForge.net offers us features to show knowledge about the project like news, mailing list, bug tracking, etc. And the Wiki part will show the rest of information that SourceForge.net will not able to keep.

As we can see, almost all the Fogel needs to start an OSS project are filled. Real-time chat could be supported by the Wiki for future work. We have not carried out this task for our project.

Finally, in the next section we are explaining more deeply each requirement. Perhaps the reader could feel some of the requirement little abstract and fuzzy. The reason of this is that we have worked with management goals and, for example, project documentation tasks are difficult to define and delimit. But, in order to clarify and delimit these requirements we have asked us the following questions: *Will Brandtsegg (the future leader) be able to manage and lead ImproSculpt OSS project? Will the end-users be able to install the ImproSculpt bundle?* For more information about these two questions, please refer to the evaluation chapter.

III.2 - ImproSculpt OSS Launch Required Tasks

Start the SourceForge.net Project

Description

SourceForge.net is a web site which repertory most part of the existing Open Source projects. If we want attract some developers or just give a reputation to ImproSculpt the project has to be visible on SourceForge.net

Expectations & goals

SourceForge.net is the base of an Open Source project. We expect two different kinds of advantages using it:

- It offers a lot of good features to manage open source development (for example: CVS server, forum, website server, etc.). Everything related to the project will be centralized on this website.
- On the other hand, this will result as a good Advertisement for ImproSculpt as it will be visible and accessible to everyone.

Functional Requirements

Creating a new project in this web site needs a lot of information about the architecture of SourceForge.net. We need to find this information, but also describe properly the purpose of ImproSculpt because it will be examined by SourceForge.net administrators before being available.

OSS Software License

Description

Software licenses are designed to take away your freedom to share and change it. However, OSS licenses are thought to guarantee the condition of source code available for everybody who willing to read it. Thus, they enable the characteristic that software id free for all its users. The most popular is GPL (General Public License) [13].

Expectations & goals

The main purpose of releasing the code under OSS license is to protect it against companies which may want to earn money with. If they patent the code, Øyvind will not be able to work on it, use it (in stage or in studio) any longer. It is also a requirement to publish the code on the SourceForge.net web site.

Functional Requirements

In order to choose the license which fits perfectly with ImproSculpt and Øyvind goals, we need to understand the differences between each license. We have to decide the best one with the Øyvind's opinion and add it to all of the files of ImproSculpt bundle.

Concurrent Version Server

Description

CVS implements a version control system. It means that this software is designed in the purpose of keeping track of all work and all changes in a set of files, typically the implementation of a software project, and allows several (potentially widely separated) developers to collaborate.

Expectations & goals

Thanks to CVS server, everybody will have access to the source code. Many developers will be able to work at the same time on the same file without any conflicts. In the case of a strong bug or a bad implementation, it is always possible to access to recover an older version.

Functional Requirements

The first task is to understand and configure the CVS server related to ImproSculpt project on SourceForge.net. After that, a generation of a public/private SSH key is necessary to access to the server. Then it is important to upload properly the last version of the source code on the server. We choose the CVS client TortoiseCVS [14] in collaboration with Putty [15] for the SSH key generation. This client is the easiest and the most widespread one to use with Windows. When this will work, we will add to the Wiki a small quick tutorial on how to set up TortoiseCVS.

Wiki

Description

As we said before, all the information related with ImproSculpt project will reside on the Wiki. This is very important because the attraction of the project will depends of the information that will be distributed and how it will be organized. To sump up, this task lies to set up a powerful Wiki which allows us to do this.

Expectations & Goals

The main expectation of this point is to get a sure and powerful Wiki which centralize all the information. After that, another important goal that we follow is to get a filled Wiki with ImproSculpt's information as clear as possible. If these two goals become well done, the expectation of doing an attractive project to developers will be achieved.

Functional Requirements

SourceForge.net offers to the launchers some space to set up a web page [16]. This space is located on a server which has installed a PHP [17] hypertext processor and MySQL stuff [18]. We have to look for a Wiki implemented on PHP 4.3.8 [19]. In addition, we have two possibilities for save the Wiki's information on SourceForge.net. One of them is using a MySQL database and the other one is using files. We need to look for a Wiki that uses files to save the data because we do not have knowledge about how to create and manage a MySQL database. And to make up the security lack that saving in files produces in SourceForge.net [20], we will establish a backup policy [21] in the writeable space of SourceForge.net.

After this, we need to configure the Wiki in order to make it more familiar with ImproSculpt. Thus, people will be able to find the information easily and quickly. We need to classify the information, to set up the Wiki's appearance, to include the SourceForge.net logo in all the pages, to establish an ImproSculpt logo, to configure the different kind of users (Administrator, editor, reader, etc.) and finally to fill in the Wiki with ImproSculpt's information.

Documentation

Description

One of the most important characteristics of OSS philosophy is the ability to recruit new developers for work in this kind of projects. Registering ImproSculpt OSS project on SourceForge.net is an important advance for ImproSculpt publicity. Once SourceForge.net includes a project called ImproSculpt, it is necessary to document it in order to get attention of the OSS community.

Expectations & Goals

As of today, all the information related with ImproSculpt is scattered. Creating an OSS with ImproSculpt bundle in SourceForge.net will allow us to centralize all the information related with. And secondly, we will get an organized template where will be easy to find any information.

In OSS community, usually future developers will be first end-users. I.e. they will install and use ImproSculpt before they will become ImproSculpt developers. For this reason, it should be easy to install it. Thus, the second expectation of this requirement is that anybody (with computer knowledge or without) will be able to install ImproSculpt with the documentation that we will provide.

Functional Requirements

We will divide the ImproSculpt's information between two supports: SourceForge.net documentation tools and ImproSculpt Wiki. The following list shows the task that we will performance related with the documentation requirement.

- *SourceForge.net*
    - Obtaining and filling ImproSculpt project details
    - Deciding and creating the necessary packages to classify the initial releases of ImproSculpt bundle (Flyndre, ImproSculpt Classic and Unstable ImproSculpt).
    - Releasing ImproSculpt bundle classifying its components in the created packages.
    - Documenting each release of ImproSculpt bundle.
    - Writing some news in order to announce OSS community that the first release of ImproSculpt is ready.

- *ImproSculpt Wiki*
    - Designing the Wiki information structure for classifying the documentation.
    - Writing an overview about ImproSculpt.
    - Documenting each component of ImproSculpt bundle (Flyndre, ImproSculpt Classic and Unstable ImproSculpt).
    - Explaining how to use CVS.
    - Teaching to manage ImproSculpt Wiki.
    - Looking for a logo.

IV – Research

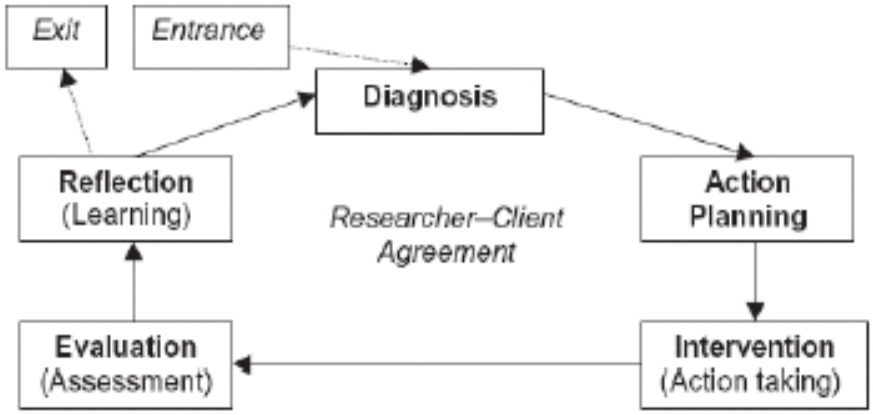
IV.1 - Research methods

Action research

Presentation

"A method that may be well suited for empirical research in the context of education is action research (AR). AR originates from social sciences, and is used for learning from experience by intervening into various systems. One orientation of AR is Canonical Action Research (CAR), proposed by Davison et al. [22]" [23].

CAR is defined as an iterative process on the product developed with a descriptive plan and performed cycles of activities. One of the main points of CAR is the importance of the diagnosis process. Because based in this diagnosis the student gets feedback to apply for future improvements.



Application to our project

"*Iteration* *in computing is the repetition of a process within a computer program***"**

"*Iterations* *in a project context may refer to the technique of developing and delivering incremental components of business functionality. This is most often associated with Agile software development, but could potentially be any material. A single* ***iteration*** *results in one or more bite-sized but complete packages of project work that can perform some tangible business function. Multiple* ***iterations*** *recurse to create a fully integrated product. This is often compared with the waterfall model approach.***"** [24].

Above we have two definitions of Iteration from Wikipedia. The first one is the definition of iteration in the programming context meanwhile the second one is in the software engineer context. This project includes many small tasks to perform in order to set up a high quality work environment (see Problem Elaboration). We could call these tasks: atomic task. I.e. they are indivisible. For that reason we cannot establish a iterative plan in each of these task.

Another question that could arise in this point is: *Why do not divide the whole work in iteration including set of task in each iteration?* We cannot do that. The main point of CAR is problem diagnosis. I.e. it necessary get feedback from one iteration to apply to the next one. If we, for example, attempt to get feedback from the wiki, this feedback could not be applied to the CVS.

In conclusion, the quantity of small task and the size of them make impossible to fit this method with our project.

Case Study

Overview

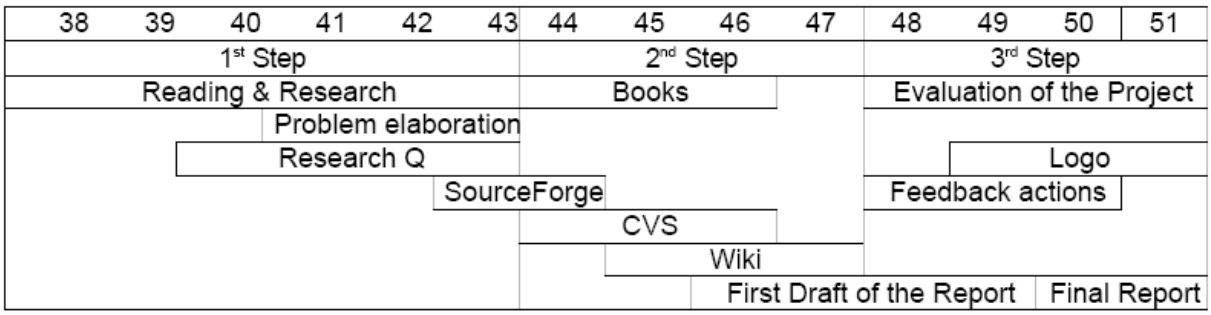
A **case study** is a research strategy, to be likened to an experiment, a history, or a simulation and not linked to any particular type of evidence or method of data collection (Yin 2002 [25]).

Rather than using large samples and following a rigid protocol to examine a limited number of variables, case study methods involve an in-depth, longitudinal examination of a single instance or event: a case. They provide a systematic way of looking at events, collecting data, analyzing information, and reporting the results. As a result the researcher may gain a sharpened understanding of why the instance happened as it did, and what might become important to look at more extensively in future research.

Application to our project

As we would like to investigate both faces of artistic software, we thought applying case study to two different project, and compare them. A deep research in the mailing lists, forums, website and an interview for earch project was planned. But as Letizia explained us, case study is based on the study of one particular project. And as we really did not miss information on one or the other case, we could not apply this longitudinal research method.

IV.2 - "Home-made" Research method



Although we have not followed any research method found in the literature like Case Study, Action Research, etc., the evolution of our project has tracked some predefined steps. But, what is a method? A method is a process to get something, in our case it is a process to get knowledge.

***"Scientific method** is a body of techniques for investigating phenomena and acquiring new knowledge, as well as for correcting and integrating previous knowledge. It is based on gathering observable, empirical, measurable evidence, subject to the principles of reasoning"* [26].

Thus, our project evolution has followed the method shown in the above picture. As we have said before, we have not been able to fit our project with Action Research or Case Study, but our method has one similarity with Action Research. We have divided the work in three steps depending of the work nature. And from the conclusion of each step we have extracted knowledge to confront the next step. This thinking is similar to the Action Research Process. As we said before, Action Research divides the work in iterations and takes out feedback from each iteration in order to improve the final product in the next iteration. However, the main point of Action Research is that all the work and the improvements based on feedback are focus on the final product. As our project works with the management topic in OSS, we do not get a real product at the end. We have had to perform small tasks in order to establish an OSS work environment. Thus, it would not be smart divide each task in iterations because these task does not have enough work charge.

As the picture shows, we have divided the work in three steps:

- *First Step:* Readings and researches; Definition of our project between practically as much as research.
- *Second Step:* Launching ImproSculpt OSS project.
- *Third Step:* Evaluation.

Our poor knowledge in Open Source Software and Artistic Software before starting the project forces us to make a deep research in these two topics. Indeed, it was really difficult to understand how ImproSculpt works. As we said many times before, ImproSculpt OSS project does not just include one software. It is a set of software which form the ImproSculpt environment and, accordingly ImproSculpt OSS project. We needed to soak in this environment in order to describe all its meanings in ImproSculpt Wiki, SourceForge.net, etc.

Thus basically, the first step included researches in OSS, Artistic Software, Research Methods and ImproSculpt bundle as well as SourceForge.net architecture.

With the knowledge acquired in the first step and kept in the Problem Elaboration and Research Question documents, we were ready to start the second step. This step basically includes the necessary tasks to launch ImproSculpt OSS project. We noticed that all this small task (see Problem Elaboration Chapter) could be performed concurrently. SourceForge.net, Licenses, CVS, Wiki and Documentation tasks kept on the Problem Elaboration document were not dependent. Just SourceForge.net created dependency with CVS because CVS server used by ImproSculpt OSS project is located in the SourceForge.net organization. Thus, we decided to perform first the SourceForge.net tasks. After that, we were working in parallel in the rest of the task.

As you can see on the picture, Books slot belongs to the second step. This is because the readings step went on the second one. The book which we read was *"The Cathedral and the Bazaar"[27]*.

"*Eric S. Raymond on software engineering methods, based on his observations of the Linux kernel development process and his experiences managing an open source project, fetchmali [28]. It was first presented by the author at the Linux Kongress on May 27, 1997 and was published as part of a book of the same name in 1999. It is commonly regarded as a manifesto of the open source movement ".*

This gave us the perfect feelings to write the Documentation part of the project (see Problem Elaboration).

Finally, at the end of the second step we began to decide the structure of our final report. Thus, when the second step was finished we got the first draft of our report and all the practical tasks performed.

In the third step, the valuation one, we started to raise the number of emails with Øyvind Brandtsegg, our client, in order to get the needed feedback for our performed work. With this feedback we made the final improvements.

In this step, we were able to establish a communication in a OSS forum (www.linuxforum.org) in order to find some OS artist who would create a logo.

The final version of our report was the output of this step.

IV.3 - Communication

How does all of this started?

Everything started during the introduction week for Exchange students where both authors of the present document met. We noticed we had the same project and few days after, we learned this project can be done in teams. In our home universities, the team work is more accentuate than in Norway. Here most part of the projects is done alone. This paragraph does not have any purpose of criticising the Norwegian educational system! It is just a comparison. A single student will work differently if he is alone on a project. He will do all the tasks and have a much better overview of the whole project. He will be responsible of every problem, but also every quality. But the point is we were used to work in team, we decided to do this project together.

After meetings with many teachers offering us some possible project, Letizia told us about her branch and her investigations into Open Source / Artistic Software. As nobody before exanimate the close relation-ship there could be between both, we thought this could be very interesting.

But we needed to apply this on a practical goal. After some research on the different existing projects on the web, we did not find the two faces of Artistic Software (make by artist, or for artists) reunite in one project. As we wanted to investigate both ways but could not work on two projects at the same time, we started to think about orienting differently our project. But finally, after explaining this problem to Letizia, she told us about Øyvind Brandstegg who used to work on the ImproSculpt project and who needed to open his sources.

Thus, we decided that launching an Open Source Project based on his source code could be a very interesting practical goal as ImproSculpt is right in the border-line between the two categories of Artistic Software.

Letizia Jaccheri

Letizia [29] was our project supervisor. Her implication in this project was in the research/method part. She gave us plenty of advice relative to research methods (Case Study and Action Research) but as explained above we wanted to follow our "own research method"(which is presented in the preceding part and discussed in the Evaluation).

Apart from that, she gave us a lot of good references for both Open Source Software and Artistic Software. We were not used to read so much. This is also a difference between France / Spain and Norway. We had a very interesting discussion with Letizia during a TDT69 course. We argued that Internet is a very good way for researching as the information is more spread, and with feedback very fast. For example the Wikipedia [1] which is based on all the information coming from all over the world gives more objectives information than a book written by only one author. More than that, the cost of providing for one person with some information is not a greater expense than providing 10 mission people with the same information, no matter where in the world they are [30].

She was trying to convince us that the information provided by a book are more specific, and better written that an article on an internet blog. Anyway, this was a very interesting discussion and we decided to read at least one book (which was the Cathedral and the Bazaar [27]). This added with all the texts of the syllabus from the two courses and the articles we found on the internet was much more than enough to establish a good research data-base.

Letizia was also here to ask regularly for deliveries in order we never forget to focus on the project. To finish, we would like to thank her for all the information she provide us, whatever it is relative to internet texts, books, or even to bring us to the Matchmaking festival [31]!

Øyvind Brandstegg

The relationship with Øyvind was completely different. We were working for him, but also with him in the sense he was our client and our project-mate. As in many Open Source Software the face-to-face is not necessary, it was the same for us. We meet one time at the beginning to set up the basis of the project, and then all the discussions were done by casual tools used during Open Source development (mail, instant messenger, forums …)

During the first step, the communication flow was quite low. As we were reading and looking for information about AS/OSS, we could do that alone.

Then, when we started to investigate the source code to run it, to start the project on SourceForge.net, the number of mail per day started to pass the ratio of more than 1.

The second step became lower again. As we were learning about CVS, Wiki and set up everything, we worked autonomously.

Then the third step arrives and we needed to introduce Øyvind to the project development environment we designed for him. The number of mail per days increased to reach the top of 10.

He gave us plenty of advices, information about ImproSculpt, musical software and artistic software in general.

Rune Flobakk

In this communication part, we must talk about Rune. He was also working on ImproSculpt, and following the TDT69 course. We exchange quite a lot of information relative to ImproSculpt (how to install it). He was very kind to help us with a lot of casual problems. All the information relative to the University, administrative problems, language, how to go to Øyvind's office, is just a non-exhaustive list of what he did for us.

IV.4 - Discussion around AS and OS

Deeply into the research question

The first time we looked for artistic software on the web, we found tools for artist (Photoshop, 3D studio Max, blender). There is also another field of artistic software: when the software is artistic. But can software be a piece of art?

Art is defined by the following:

"The use of skill and imagination in the creation of aesthetic objects, environments, or experiences that can be shared with others"

Britanica Online (from The wikipedia definition) [32]

What is aesthetic?

"Aesthetic is a branch of value theory which studies sensory or sensori-emotional values, sometimes called judgements of sentiment or taste"

The wikipedia [33]

So based on this, if we think about a video game, an informatics program can easily be considered as a piece of art.

But what about ImproSculpt? It can produce music by itself as we saw with Flyndre, so it is clearly a piece of art. But Øyvind use it for playing in live. It is also a tool.

What we would like to explore is the influence of open source in each of these case.

What happens when artistic software meets open source?

Thus will help us to understand the position of ImproSculpt and what can open source bring to it.

Software designed in an artistic way

What open source can bring to a project like a video-game (which can be considered as a piece of art nowadays)?

In one word: freedom.

An artist needs to work as he wishes to transmit his entire artistic skills to his piece of art. This is something impossible in a huge company (the Cathedral) where you can feel the clearly delimitation. In an open source project, the programmer can work on the part he wishes (or on the part where he is good). The choice of the project part you will work is decided only based on your skills and wishes.

For example, if you are working on the design of the main character of the next big production of a huge video games company, and you have a very great idea on how the sound should be for a particular move, there is very few chances someone listened to you.

In an open source project, you realize this sound and you propose it to the community. It will be accepted or not, but if you have the skills as explained above, it has great chances of success.

When we read about open source, we notice that very often open source philosophy is argued to be found on gift relationships as each piece of software is a gift to the community. One paper from Magnus Bergquist & Jan Ljungberg explains this with a lot of details and relevant examples [30].Open Source philosophy is a meritocratic system. The one who works the best (the much) get the respect he deserves and gain access to higher position in the development. This is due to the power of gifts. Mauss [34] argues that gifts express, but also create, power relations between people. One of the norms of gift giving is the rule that gifts be accepted. Therefore, the receiver becomes subordinate to the giver. In the open source community, this can be the fact. But the opposite is also important. Refusing to accept a gift can, in some situations, be a way of showing superiority.

In brief, an artist with some skills will get better and faster a powerfully position for influencing the project.

But one question comes again in many papers: Why peoples produce thousand of code line for free?

From A. Bonaccorsi & C.Rossi [35], the main motivation for working on an open source project for free can be altruism. But also the production of open source software is a form of intellectual gratification with an intrinsic utility similar to that of a scientific discovery: sharing knowledge in order to access to a common goal. One part of this goal is of course the prestige (gift economy), but not only.

Apart from this intellectual work, hackers also regard programming as an art form. Several authors describe Open Source programming as artistic satisfaction associated to solving complex computer problems.

Finally there is the pleasure of creativity which is progressively lost in the commercial world full of deadlines.

These three reasons are very close to the reason that a musician will write a song.

But the programmer should be able to give his gifts. And for that's, a development environment is required. This is the reason why we set up a CVS server.

Software designed for an artistic purpose

What happens in the other case?

The advantage of open source is the close relationship between the developers and final users. An open source software is constantly in evolution, integrating new features or fixing bug based on the feedback of the users.

To come back to the main motivation of developers, if someone starts a new open source project it is because the tool he needs does not exist. In that case the final user is the programmer. Open source offers to artists the possibility of creating, tuning, and fixing their own tools. They have two solutions for that. Or they program their own tool, or they try to influence the community with good feedbacks, bug detection, ideas of improvement …

This explained us the utility and the importance of a good communication system in an open source project. Based on this, we noticed that just a CVS server would never be enough. An architecture for the communication issue was required. This was done with the Wiki, responsible of keeping the community aware of the progress of the project.

Software designed artistically for artistic purpose

ImproSculpt is a piece of art and a tool at the same time. It should join both categories, but this is not so clear. Øyvind designed it. Before being a programmer, he is a musician. The code is not artistic in itself, but more designed for an artistic purpose.

This mixing is quite blurry, because we can also ask the question: when someone else plays in stage with ImproSculpt? Who is the artist? Accorded to the presentation of Anna Notario during the Match Making festival[31], the author tends to disappear to the profit of cooperation art.

This is precisely the point of Open Source in Artistic Software.

V – Own Contribution

V.1 - Licenses

We had to face to the choice of a license. As there is actually more than 50 licences approved by OSI [36] and this is law text quite hard to read, we had to investigate the different discussion and articles available on the internet to learn more about that.

First of all :

What is a License ?

To **license** or grant **licence** is to give permission. A **licence** (British English) or **license** (American English) is the document demonstrating that permission. License may be granted by a party ("licensor") to another party ("licensee") as an element of an agreement between those parties. (from Wikipedia[1]

What is Copyleft?

As " Is the glass half-full or half-empty" we can see the things differently. A licence can be a document which ***forbids*** the access to everybody ***except*** few peoples. Even if this is not due to the licence, but to the copyright, the license can work in the other sense to changing completely the meaning of the former. But here is the first lines of the GNU-GPL licence [13] (the most used) which explain better this than us :

"The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things."

"To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library."

This way of using the copyright in the opposite sense of it is qualified of ***Copyleft***

Which warranty do I have?

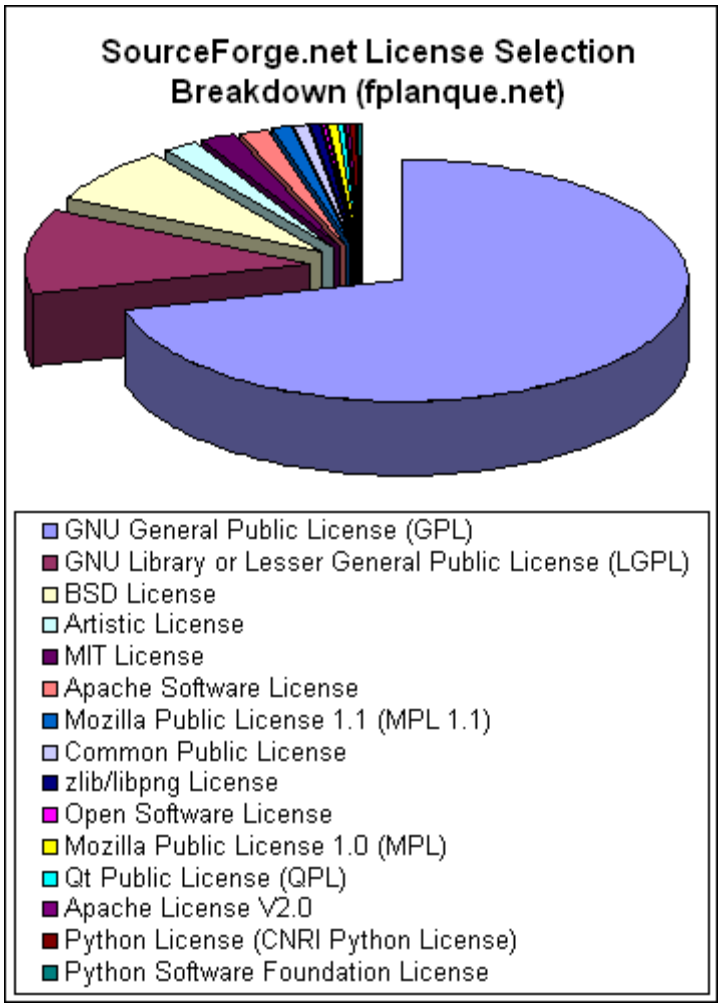
To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Why there is so many licences?

Everybody can create a licence. You just need a piece of paper and a pencil. After if you want your licence “open source” you have to make it approved by Open Source Initiative. If you distribute your software under one of these licenses, you are permitted to say that your software is "OSI Certified Open Source Software." [37]. That is why everybody who want can create a new licence with the specifically clause needed.

But when we look at the repartition of the licences (found on a very good explanation of Open Source Licence but in French [38]):



We notice that GPL rest the most widespread one. Even if for example in this interview from Eric Raymond [39], a lot of good arguments are given to explain why GPL licence is too old (20 years old) and is not as useful as it was.

We can classify them in three categories:

- Licences allowing the user to change for closed source whenever you want (the only condition is to mention the author and his copyrights) :
  - BSD License,
  - MIT License,
  - X.Net License,
- Licences forcing to keep the code in closed source when modified but allowing combination with closed source code :
  - MPL (Mozilla Public License),
  - LGPL (GNU Lesser Generak Public License).

- Licences allowing no concession relative to the open source state or combination with closed source code.
  - GPL (GNU General Public License).

"What I want from the license is to give us protection against theft of the code into commercial closed source projects."

Oeyvind Brandstegg

As Øyvind wanted the highest protection possible, the GPL was chosen

The method for implementing a licence

To protect a project with an OS licence, the only thing to do is to include (at the beginning of each file) a summary of the licence with Øyvind's copyright:

```
#   Copyright 2006 Øyvind Brandtsegg

#   All contributors can be found in credit.txt file

#   This file is part of ImproSculpt.

#   ImproSculpt is free software; you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#   the Free Software Foundation; either version 2 of the License, or
#   (at your option) any later version.

#   ImproSculpt is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.

#   You should have received a copy of the GNU General Public License
#   along with ImproSculpt in the GeneralPublicLicense.txt file;
#   if not, write to the Free Software Foundation,
#   Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
```

Of course, this is not enough; the full licence should be distributed with the code. This file was added on the CVS server as inside the released files.

The licence gives us some freedom, but there is rules to respect (for example, the source code should be available AS a released version of the code). We needed also to write a "credit" file with all the contributors to the project.

V.2 - Experiments on ImproSculpt Source Code

The first thing to do when working on software, whatever the tasks assigned, is to launch it!

Something which can be easy when working in a company where all the working station are devoted to. But when it is question of a software realized by a musician where many generations of students have already worked on it, modify organisation of the files, when the artist implement regularly some features, the task is becoming much harder. First we had to investigate the way for running it, and then accorded to our Open Source project we needed to add a tutorial on how to run it.

Flyndre version

The first installation was tried on Windows with the Flyndre version. It is the most stable version which implements the most part of the actual features even if the application is especially designed for a special purpose.

Here is the list of the software needed for launch it:

Csound: Flyndre works on the last version (Csound 5.0), but only the *float version*, not the *double version* [40]

Python: This Software is implemented using Python technology, concretely with *Python 2.4*.

Flyndre does not work with the latest version 2.5. The unknown reason behind this problem could be an interesting task to research and to discuss on ImproSculpt's forum. [41]

wxPython.wx: The Flyndre GUI is implemented using the features of this library. [42]

Partikkel.dll: Copy this special version of partikkel.dll (available on the wiki) into the CSound/plugins directory.

After that, the station is ready to launch ImproSculpt, but there is still little modification to do in the source code.

We had to comment a line responsible of the program asking for the information relative to the sensors on the sculpture. As only the computer responsible of the website is allowed to receive these information, the connexion is refused. But the server could suffer of so much connexion attempt.

The main modification is the line **84** into the */control/eventcaller.py* file from Flyndra folder. This line looks like:

```
# set the parameters for the csound command line
csoundCommandline = "csound -b800 -B1600 -d -odac3 -m0 -+rtaudio=pa
--expression-opt temp.orc temp.sco"
```

The number after *-odac* represent the sound output used by Flyndre. If you try with a high number like 100, this will result by an error and it will offer you the list of your output sound devices and the number to use for each one.

In this line, *-b* refers to the Software buffer and *-B* refers to the Hardware buffer. The number after *-B* should be a multiple of the one after *-b* and at least two times bigger. The number after *-b* should also be a multiple of the *ksmps* value in *flyndra.orc* file in the Flyndre folder.

ImproSculpt Classic

It was the first stable version of ImproSculpt, before Python. ImproSculpt is based on Csound code and provide a GUI not so easy to manipulate. Thus is much easier to install. It only require Csound and the special version of partikkel.dll

The same line for setting up the parameters for Csound need to be arranged depending of your hardware sound configuration.

This code can be found line 26 into the .csd file:

-odac12 -iadc13 -b100 -B400 -M8 -m0 -+rtaudio=pa --expression-opt -d -+raw_controller_mode=1

To run, there is of course, no sensors problems, but still an “input” one. ImproSculpt classic is running based on midi-input

The -M switch in the options sets the midi input device. Normal usage is -M n where n is the number of your midi device.

If there is no midi device on the system, the -M part of the line should be removed.

Alternatively, MidiYoke which is a virtual midi device driver, can give a midi device in software

ImproSculpt Unstable

Since the Flyndre version, Øyvind continued to work on ImproSculpt. This is the last version which is actually in bug fixing state. We did not succeed to launch it on our machines and in parallel of the redaction of this report we continue to try different things. The most up-to-date information relative to ImproSculpt are available on the Wiki.

V.3 - SourceForge.net

When launching an Open Source Project, a good opportunity to consider is sourceforge.net. SourceForge.net is the world's largest Open Source Software development website, hosting more than 100,000 projects.

Here is a non-exhaustive list of SourceForge.net features:

File Release System:

Most users go to a project looking for files to download and new software to try. The SourceForge.net File Release System allows projects to make downloadable files available to users.
When files are posted through the File Release System, they are automatically placed on the group of download servers, called mirrors, located around the world. Users are able to pick a download server nearby their geographic location, as to get the best possible download times.

CVS

SourceForge.net provides CVS service to projects, allowing them to centrally store the source code to the software they are developing. This central storage allows easy access both for project developers and end-users. Projects have the option (it is not mandatory) of using the SourceForge.net CVS service to maintain their source code; each project is provided their own CVS space, called a repository.

Site users may retrieve the very latest (not necessarily ready for release) source code from CVS. Project developers are able to post their changes back to the source code stored in CVS. CVS helps to prevent developers from overwriting changes made by other developers. CVS also keeps track of each change made to the source code, including who made the change.

A CVS client software program will be needed in order to access a CVS repository. Read access to a CVS repository is available to all users. Write access to a CVS repository is provided to project developers with permissions provided by the project administrator.

Communication

Communication is the key to the long-term success of any project. In Open Source project the communication is even more important due to the non-physically proximity of the developers. SourceForge.net provides several tools to aid in the communication process, both within the project team, and between software end-users and the project team.

Trackers are web-based resources that allow the easy management of Bug Reports, Support Requests, Feature Requests, and user-submitted source code Patches. Each Tracker may be searched, so users can locate information on prior resolutions of the problem they are trying to solve.

Forums are web-based resources used to discuss development, usage issues, and other project details. Posts are threaded and may be viewed in several different ways. Each forum may be searched, so users can see if a question has been answered previously.

Mailing Lists are mail-based resources for users and developers to communicate with list subscribers. All mailing list subscriptions are provided on an opt-in basis. Both public and private-to-project-member mailing lists are available. All mailing list posts are archived for later web-based viewing and searching.

Starting a new project

This was our starting point. The creation of the new project named ImproSculpt on SourceForge.net with the information provided by Øyvind and found on internet took us quite a lot of time. In fact, a huge quantity of documentation relative to project management on SourceForge.net and to ImproSculpt on Internet had to be read before launching the project. After that, we divided the work between the Wiki to create and the CVS server to manage.

V.4 - CVS

How CVS works?

CVS uses client-server architecture: a server stores the current version(s) of the project and its history, and clients connect to the server in order to check-out a complete copy of the project, work on this copy and then later check-in their changes. Typically, client and server connect over a LAN or over the Internet, but client and server may both run on the same machine if CVS has the task of keeping track of the version history of a project with only local developers. The server software normally runs on Unix (although at least CVSNT server supports various flavours of Windows and Unix), while CVS clients may run on any major operating-system platform.

Several developers may work on the same project concurrently, each one editing files within their own *working copy* of the project, and sending (or *checking in*) their modifications to the server. To avoid the possibility of people stepping on each other's toes, the server will only accept changes made to the most recent version of a file. Developers are therefore expected to keep their working copy up-to-date by incorporating other people's changes on a regular basis. This task is mostly handled automatically by the CVS client, requiring manual intervention only when a *conflict* arises between a checked-in modification and the yet-unchecked local version of a file.

If the check-in operation succeeds, then the version numbers of all files involved automatically increment, and the CVS server writes a user-supplied description line, the date and the author's name to its log files. CVS can also run external, user-specified log processing scripts following each commit. These scripts are installed by an entry in CVS's `loginfo` file, which can trigger email notification, or convert the log data into a web-based format.

Clients can also compare different versions of files, request a complete history of changes, or check-out a historical snapshot of the project as of a given date or as of a revision number. Many open-source projects allow "anonymous read access", a feature that was pioneered by OpenBSD [43]. This means that clients may check-out and compare versions with either a blank or simple published password (e.g "anoncvs"); only the check-in of changes requires a personal account and password in these scenarios.

Clients can also use the "update" command in order to bring their local copies up-to-date with the newest version on the server. This eliminates the need for repeated downloading of the whole project.

CVS can also maintain different "branches" of a project. For instance, a released version of the software project may form one branch, used for bug fixes, while a version under current development, with major changes and new features, forms a separate branch.

CVS uses delta compression for efficient storage of different versions of the same file. The implementation favours files with many lines (usually text files) - in extreme cases individual copies of each version are stored rather than a delta.

CVS Terminology

- *CVS client:* Software run by a CVS user to access the CVS server.
- *CVS repository:* The CVS server stores a copy of the software and data that the project has uploaded. The server retains both the most recent version and every historical version (past changes). This copy of the software and data uploaded by the project is a CVS repository. Each project hosted on SourceForge.net has its own CVS repository.
- *module:* The toplevel directories in a CVS repository are called modules. A CVS repository can contain multiple modules. Each module lives independently within the CVS repository. A module is created using the CVS import command.
- *working copy:* Though the CVS repository stores every version of every file that has been uploaded to the repository, when you retrieve data from the CVS repository using your CVS client, only one version of each file is saved to your hard drive. The copy of the data you get from the CVS server is called a "working copy", obtained using the "checkout" command. When you checkout your working copy, you will specify which module you wish to retrieve; only the files and directories located in that module will be placed in your working copy.

Even if SourceForge.net provides a free CVS server already set up, they clearly advise to train on a local server first. We follow this piece of advice and on a Linux computer we succeed to create a CVS repository and could train with the basic commands lines(commit/update/checkout/add file/remove file).

CVS is a complex tool. We encourage all new developers to practice CVS commands locally before using SourceForge.net CVS servers for the first time. Instructions on configuring a local CVS repository for testing are provided in CVS client configuration instructions.

The second step was to try to create a CVS server on the web space offered by NTNU. As we did not have all the required access and because it would have took us too much time to ask for it, we decided to skip this part and to start directly with the SourceForge CVS server.

There is no required manipulation to set up the CVS service provided by SourceForge.net. Most part of the work describe here was to set up the environment (CVS client + SSH based authentication) for a developer. As some users may be artist without any deep background in computer science, a tutorial for a rapid and easy access should be written.

First, what is SSH ?

SSH is a protocol designed to allow secure communication between hosts on the Internet. All SourceForge.net project services for developers (including shell service, CVS write access, and Compile Farm service) are provided using SSH. SSH is used for improved security over alternatives, such as older protocols like TELNET and FTP.

SSH tool suites typically include an SSH client that can be used to access remote hosts interactively, one or more SSH-based file transfer tools, and utilities for managing the SSH keys used for authentication to remote hosts.

Each SSH key pair has a public key component and a private key component. Only public key data should ever be uploaded to SourceForge.net. SourceForge.net developer services (shell, developer CVS, Compile Farm) should only be accessed from secure, trusted machines.

As explained above the developer-access requires a SSH-key generator which will produce a couple of public/private key, and an SSH-key manager.

The two most famous tools used for that OpenSSH for Linux [44], and PuTTY [15] for windows. We suppose that the developers working with Linux are enough experimented to know this tool (or to learn fast how to use it). We focused our research on PuTTY which provides a graphic interface for windows users. PuTTY is a collection of tools. We will explain here the mechanism of two of them:

- *PUTTYgen.EXE*: tool for generating SSH keys. Similar purpose to the 'ssh-keygen' tool from OpenSSH. PUTTYgen will create a private key based on the movement of your mouse during few seconds in order to get the best random number possible. Based on that, the public key will be generated. The private key is able to produce public key as the opposite is of course not possible.

- *PAGEANT.EXE*: tool for storing SSH key credentials for authentication. Similar purpose to 'ssh-add' and 'ssh-agent' tools from OpenSSH. When you run Pageant, it will put an icon of a computer wearing a hat into the System tray. It will then sit and do nothing, until you load a private key into it. If you click the Pageant icon with the right mouse button, you will see a menu. To add a key to Pageant, press the ‘Add Key’ button. Pageant will bring up a file dialog, labelled ‘Select Private Key File’. Find your private key file in this dialog, and press ‘Open’. Pageant will now load the private key. If the key is protected by a passphrase, Pageant will ask you to type the passphrase. When the key has been loaded, it will appear in the list in the Pageant window.

Now if you start PuTTY and open an SSH session to a site that accepts your key, PuTTY will notice that Pageant is running, retrieve the key automatically from Pageant, and use it to authenticate. You can now open as many PuTTY sessions as you like without having to type your passphrase again.

TortoiseCVS

TortoiseCVS [14] is based on the same protocol than a PuTTY client. It means you will be able to work directly on your CVS repository without having to type any password (expect the time you load your key), thus with a high secure connexion.

Now the SSH-key are produced and the working station is ready, the CVS client needs to be installed.

TortoiseCVS is freely available under the GPL.

Even if there are some CVS things that you can't do with TortoiseCVS, it is more than enough for a developer on SourceForge.net and much easier to use than a command line CVS client. With TortoiseCVS you can directly check out modules, update, commit and see differences by right clicking on files and folders within Explorer. You can see the state of a file with overlays on top of the normal icons within Explorer. It even works from within the file open dialog.

Here is a list of the main operations available in TortoiseCVS:

Common read operations (available to everyone):

• Checkout: To download the last version of the source code. Create a new directory on your hard drive, then right-click and CVS Checkout. Use the settings presented above. The whole project will be downloading into the directory.

• Update: To incorporate changes done by others in your local working copy use the "update" option. Updating may be done on single files, a set of selected files, or recursively on entire folder hierarchies. To update, highlight the files and/or folders you want, right-click and select CVS Update.

Common write operations (available to developers via SSH authentication):

• Add: To add a new file simply place it into the project folder. The question mark indicates that these files are not stored in CVS. To make them part of CVS, select them, right-click, and then choose "CVS Add". The red icons mean that the file has been modified compared to what is stored in the CVS repository. The icons would also change to red if you had modified files that you have updated.

• Commit: To store the changes made in your files to CVS, select them, right-click, and then use CVS Commit. You'll be prompted to enter a comment. Enter some information about the changes that you just made.

Resolving Conflicts:

It is possible that when you go to upload your changes, someone else has made changes to the file and uploaded them in the interim. When this happens, a conflict occurs. As CVS knows nothing of your project, it leaves resolving the conflicts to the developers. Whenever a conflict is reported, you should open the file in question, and search for lines starting with the string <<<<<<<.

The conflicting area is marked like this:

<<<<<<< filename

```
  your changes
```

=======

```
  code merged from repository
```

>>>>>>> revision

You must decide what the code should look like, do the necessary changes, remove the CVS markup, and commit your modifications to the repository.

Reverting to an old file version

Developers occasionally need to undo changes that have already been checked in. In the case of a bug which appears after an updating, or to have a few of the previous versions of the software. The FAQ about the file release system explain better the reason that we would do:

Should I remove older file releases to save you space?

In short, no. One of the major benefits of Open Source development comes in retaining historical file releases for your project. Old releases can be an important debugging tool. Old releases can help show the development efforts of your team. Other people may have reasons to run older releases, depending on file format changes or resource requirements. Code included in old releases, while having been discarded for newer releases, may prove useful for other projects. SourceForge.net has the space to store a large amount of release data in the long-term. Keep your old releases around: for your benefit and for the benefit of your end-users.

Going back to a specific file revision in TortoiseCVS is easy:

• Right-click on the file and select CVS --> History to bring up the History Dialog

• Right-click on the revision you need and choose the "Save this revision as..." option.

• When the Save As dialog appears, do not click the "Save" button; instead, double-click on the file's name in the directory listing.

• Answer "Yes" when TortoiseCVS prompts you to overwrite the file.

Permissions

The project permissions are directly managed through the public key of each member of the project from the admin page of SourceForge.net.

Project permissions impact host access in the following manner:

- The project permissions set by the project administrator do not restrict users from logging-in to the project shell service.

- Project permissions restrict the data that can be modified after logging-in.

V.5 - File Release

"Release early ,release often, delegate as much work as you can, and be open until the promiscuity point" (Linus Torvalds [45]).

Following this sentence the Linux community achieved such productive. Nowadays this philosophy is followed by almost all the OSS projects. These affirmations show the importance of a high quality release process.

We need to distinguish between CVS and release. It is true that in the project CVS repository you can find all the code version and even you can download it. But it is necessary to set up a release mechanism. This is because no everybody is allow to access to the CVS repository.

SourceForge.net offers a release system. *"One of the cornerstones of Open Source software development is the distribution of that software. The SourceForge.net file release system (FRS) provides a mechanism for SourceForge.net-hosted projects to post files to our network of high-capacity download servers. The file release system is used by project administrators (and file release technicians) to post content to the download servers. Once content has been posted using the file release system, it becomes available for download by the general public and is stored on our high-capacity download servers"* [46]. This FRS is organized by packages where the releases are kept. Within each file release system package will be one or more file releases.

In our case, ImproSculpt OSS project includes at the beginning three different artefacts: Flyndre, ImproSculpt Classic and Unstable ImproSculpt. It has been necessary to decided a release structure in order to classify our first three releases.

Now we are making an overview of each artefact in order to understand our final decision about how organize it.

ImproSculpt Classic: It is the stable version of ImproSculpt. This is the original ImproSculpt version built in Csound only, with a FLTK Csound GUI. This version has been actively used by Oeyvind and others in performances earlier than 2006.

- Unstable ImproSculpt: *Python technology was connected to ImproSculpt Classic and was born Unstable ImproSculpt. This is the current development branch, considered unstable as of December 2006. The application has been split into GUI, composition logic, and Csound server applications. The applications communicate via PYRO.*

- Flyndre: *It is the music sculpture software of Inderøy. It use the basis of Unstable ImproSculpt, even Python technology, and in fact it is the most stable version of ImproSculpt 4.0 (called unstable).*

We have had to create a taxonomy of packages for keeping these three different branches. The final decision was to create two packages: ImproSculpt Alpha 4.0 and ImproSculpt Classic.

ImproSculpt Classic package includes ImproSculpt Classic. And ImproSculpt Alpha 4.0 includes Flyndre and Unstable ImproSculpt. We decided fit Flyndre in this package because, although Flyndre is in the middle between ImproSculpt Classic and Unstable ImproSculpt, it is nearer of Unstable ImproSculpt. The reason is that Flyndre use the Python technology and we could say that this is the jump from ImproSculpt Classic to ImproSculpt 4.0.

SourceForge.net offers to the users a news system with two important characteristics [47]:

- *Promotion and publicity:* By releasing news articles regularly, you increase the chances that users will return more frequently to see what changes have been made within the product. This increased interest in the project may increase the number of potential developers who wish to join your project, and the understanding of the goals and achievements of the project.
- *Front page exposure:* A portion of the news articles posted by projects will appear on the front page [48] of SourceForge.net, further increasing project exposure.

We have used this system to announce the first release of ImproSculpt OSS projects. Thus, we will have the possibility to appear in the front page of SourceForge.net and likely recruit some new developers.

In order to see the practical package taxonomy part open [49].

V.6 - ImproSculpt Wiki

As we explained in the *Problem Elaboration* chapter one of the Technologies or tools that a successful OSS project needs is the web site. Karl Fogel described the OSS project web site as one centralized way to show the audience information about the project.

SourceForge.net has installed some technologies in the physical layer which allow us to establish a web site with some actual features like PHP, data-base, etc.

As ImproSculpt OSS project is looking for new developers willing to work on it, we needed a system which allows us to show information and, in the other hand, it must allow developers include new information relative to ImproSculpt or its development. Thus, we were looking for an interactive tool in which people would be able to add some comments easily. Of course, this tool must be sure and it must contain reliable data. I.e. if somebody unconnected to the ImproSculpt project would add wrong or stupid information, this mechanism should be able to restore the last version of the corrupted page.

One technology that fitted perfectly with our petitions was *Wiki*. "*In the early 2000s, Wiki's were increasingly adopted in the enterprise as collaborative software*" [50]. Nowadays Wiki technology is the most powerful tool to solve the problem of communication in collaborative software.

"*Wiki is a type of Web site that allows the visitors themselves to easily add, remove, and otherwise edit and change some available content, sometimes without the need for registration. This ease of interaction and operation makes a wiki an effective tool for collaborative authoring*" [51].

There are many Wiki software available on the net as OSS projects and you can find them implemented using several programming languages like python, php, perl, etc. Usually, this kind of web sites are implemented using languages which need a server to compile the code of the web page that you are launching, e.g. php.

Requirements looked for wiki

When we looked for a particular wiki in internet we found a web page where you can compare many different Wikis. This web page is located on moinmoin, that is a particular implementation of Wiki[52], and it shows a table with two axis, one of them contains a set with the most popular Wiki's, and the other one includes many important features to bear in mind when you have to choose a Wiki. We requested for a Wiki with the following characteristics:

1. Wiki easy to install, configure and maintain.
2. PHP implemented Wiki because we have knowledge of this language. But we needed a special version of PHP because on the web server of SourceForge.net is installed PHP 4.3.8 [53].
3. Wiki which uses files support in order to save the information, instead of MySQL database. We did not choose the MySQL bundle because neither of us have notions about it.
4. Wiki which allows organization of the Wiki users in different categories, i.e. groups. Perhaps in the future could be useful divide the Wiki users between developers and end-users, for example.
5. To be able to recover the last version of one Wiki page, our Wiki had to have implemented “versioning features”. This allows, in attack case, to establish again the latter version of the Wiki.
6. The most important feature on the Wiki looked for was the permissions. We needed a protection mechanism in order to close (for read, for edit, etc.) some Wiki pages. It is very dangerous to open some important Wiki pages for everybody.
7. Of course, we needed a Wiki with some free license.

Description of the chosen Wiki

Thanks to moinmoin´s web site and its comparison table among most popular Wiki Software running today our decision was PmWiki [54].

"*PmWiki pages look and act like normal web pages, except they have and 'Edit' link that makes it easy to modify existing pages and add new pages into the website, using basic editing rules* [55]. *You do not need to know or use any HTML or CSS. Page editing can be left open to the public or restricted to small groups of authors"* [54].

Basic editing rules make this tool accessible for everybody. People without any knowledge of computer science should be able to add information. As ImproSculpt is considered as an artistic software[56], interested artist would be able to collaborate as the same level with computer scientist. And this tool allows this communication. Net art was established itself by 1997 [57]. The main characteristics of net artist who work on internet is that they do not work alone in the same piece of art. A work of net art owns usually more than one artist. Thus, artists who do not meet themselves face to face need a useful system of communication. In the beginnings of net art, this system was based on Email.

"Email, a system for sending and receiving messages electronically over a computer network, has changed the nature of work and communication in technologized cultures unequivocally since its widespread adoption during the 1990s" [57].

Exhibition formats are necessary when we are talking about art. *What is a piece of art without admirers?* Exhibition issues in net art are very different from the regular art. Meetings like "*Beauty and the East"* and on-line exhibitions are the answer about the exhibition issues. The question now is, *could we mix exhibition and communication in net art?*

Coming back to the Wiki´s topic and analysing its offered features, *couldn't we think that Wiki is the perfect support to mix these two issues?* We could use PmWiki for communication and exhibition (not live performances, but perhaps in the future).

The following list shows some of the key features which PmWiki. We can see that it cover all the desired features:

- *Access control*: PmWiki password protection can be applied to an entire site, to groups of pages, or to individual pages. Password protection controls who can read pages, edit pages, and upload attachments. PmWiki's access control system is completely self-contained, but it can also work in conjunction with existing password databases, such as *.htaccess*, LDAP servers, and MySQL databases [54].

- *Custom look-and-feel*: A site administrator can quickly change the appearance and functions of a PmWiki site by using different skins and HTML templates. If you can't find an appropriate skin already made, you can easily modify one or create your own This feature allows us to customize ImproSculpt Wiki making artistic the ImproSculpt environment.

- *Customization and plug-in architecture*: One principle of the PmWiki philosophy is to only include essential features in the core engine, but make it easy for administrators to customize and add new mark-up. Hundreds of features are already available by using extensions (called "recipes") that are available from the PmWiki Cookbook. Thank to the plug-in technology use by PmWiki the evolving future of ImproSculpt Wiki is not closed and perhaps, future works make of the ImproSculpt Wiki a nice place to do real-time ImproSculpt performances.

"PmWiki is written in PHP [58] *and distributed under the General Public License . It is designed to be simple to install, customize, and maintain for a variety of applications".* Complete description about PmWiki features is shown on *http://www.pmwiki.org/wiki/PmWiki/PmWikiFeatures.*

ImproSculpt Wiki description

All the Wiki requirements listed on the latest section are covered by Pmwiki. This choice was made using the moinmoin´s table. And the result is the ImproSculpt Wiki [62] located on the SourceForge.net web server. This Wiki is really easy to use and is organized like many Wiki's on internet. It has two different areas, the access bar on the left to enter to the main Wiki pages and the content area to show all the information. ImproSculpt Wiki has also an action bar on the top with the main actions that you can do on a page. These actions are View, Edit, History, Attach and Print. ImproSculpt Wiki starts by a home page which contains an overview about ImproSculpt.

In this section we are describing the carried out process to set up ImproSculpt Wiki. The following tasks were performed sequentially. For information about how to manage ImproSculpt Wiki see *Appendix 1*.

Wiki Information Repository

The wiki requirement number three tells about choose file support to save the information instead of MySQL database. With this decision we needed a place in the hard disk of SourceForge.net where store the files. SourceForge.net gave us some writable space in `/tmp/persistent/improsculpt/` [61]. We decided to store the information of the wiki into this folder. But this folder is writeable and everybody can modify it. In order to save the information persistently, we established a backup policy writing a backup script (see *Appendix 1)*. It has to be executed regularly. It depends of the security that you want to achieve. This file is into `/home/groups/i/im/improsculpt/htdocs/` and it saves all the backups into `/home/groups/i/im/improsculpt/htdocs/backsup/`.

Wiki Groups

The Wiki requirement number four establish the need to get a Wiki with groups management feature. ImproSculpt Wiki allows it and it sets up as the default two different groups. The Site Group and the Main Group. The first one includes people who works in administration task (Actually it just includes as). And the second one contains the rest of people.

ImproSculpt Wiki is managed by a set of passwords called default passwords. This set includes the following passwords:

- *Administration Password:* With this password you can do everything.
- *Edit Password:* As the default PmWiki has empty this password. This password close all the pages of ImproSculpt Wiki for editing. Actually ImproSculpt Wiki has also empty this password.
- *Upload Password:* ImproSculpt Wiki allows submit files. If this password is activated you need to know it in order to upload any file.
- *Attr Password:* The main function of this password is allow somebody to set up passwords in different wiki pages.

Basically, the main difference between a person from the Site Group and the Main Group is that the person from the Site Group knows the administration password and ImproSculpt Wiki permits him to access to the Site Administration pages.

For more technical information about how to manage the password system see *Appendix 1*.

History

The wiki requirement number five requests a wiki with control version. ImproSculpt Wiki includes this feature. Clicking on the History button on the Action Bar you can restore any latter version if the page does not have editing protection, or you have the edit password for this page.

ImproSculpt Wiki skin

ImproSculpt Wiki appearance is the result of pre-design skin called “A bit modern skin” [14]. We chose this skin because we thought it fits good with the ImproSculpt feelings. ImproSculpt appearance is easy to change by changing this skin. For more technical information about how to manage the skins in ImproSculpt Wiki see *Appendix 1*.

Editing ImproSculpt Wiki

ImproSculpt Wiki, like many Wiki's on the net, uses a set of mark-ups to edit wiki pages. All the support needed to edit complex wiki pages is on ImproSculpt Wiki [60].

ImproSculpt Wiki maintenance

Actually, ImproSculpt Wiki is set up and the hard work to launch it is already done. The Administrator role is now the ImproSculpt Wiki maintenance. This task lies in to keep reliable information on ImproSculpt Wiki. Using the ImproSculpt Wiki History and the Editing Passwords is easy to do this task.

Information Structure

In order to classify the documentation which explain ImproSculpt Bundle, we needed a high quality taxonomy or data structure. An information web site should be accessible, i.e. any information should be found quickly.

As ImproSculpt OSS project has three beginning releases (see File Release chapter), one for each artefact of the ImproSculpt Bundle, we decided to make a concrete section in the wiki for each release.

ImproSculpt Wiki is really easy to understand. As we said before, on the left, ImproSculpt Wiki has the navigation bar where the main pages of the wiki are found. And, on the right, ImproSculpt Wiki includes the container place where all the pages are shown.

Looking ImproSculpt Wiki (http://improsculpt.sourceforge.net) we can see on the navigation bar the following big sections: ImproSculpt Classic, Flyndre, Unstable ImproSculpt, CVS Support and Wiki Support. Information included into ImproSculpt Classic, Flyndre and Unstable ImproSculpt is organized following the same pattern. For each one, we have classified the data in four containers: Presentation, Download, How to install and in the Flyndre and Unstable case also Documentation part.

The reason because the Classic version does not include the Documentation part is that there is no tools for generating automatically documentation from Csound files. As Øyvind said:

"But there exists no automated tool for extracting documentation for ImproSculpt Classic, and I would say writing a full documentation for Classic is outside the scope of your project."

Flyndre and Unstable source code is documented following the Doxygen standard. Thus, using the Doxygen[65] tool you can create a complete tutorial on html format based on the comments inside the source code files. We have plugged the tutorial got from Doxygen in ImproSculpt Wiki.

For future work, a full documentation of ImproSculpt Classic could be done “by hand”. But this task can be a full project topic in itself. Furthermore, the structure of this version is completely different from the structure of the actual state of the project. There are other tasks that ImproSculpt project requires before this.

V.7 - Logo

In order to start early a publicly campaign on internet to recruit new developers for ImproSculpt OSS project is very useful to find an ImproSculpt Logo. ImproSculpt Logo will help to make more professional the project and, accordingly to make it more attractive for new people.

We wanted to get the logo from an OSS community. We wanted a OS Logo. Thus, we would include an artist in ImproSculpt OSS project. The chosen community was http://www.linuxforums.org/. We needed to post a message asking for interested artist. But we had to be careful in the way of writing it.

This kind of communities usually have some rules for socializing in newsgroups. For example, posting should be kept short and concentrated in order to create a current flow of questions and answers [30].

Following the advices given by Bergquist and Ljungberg in the scientific paper "The power of gifts: organizing social relationships in open source communities", we wrote a message short, concentrated, humble, respectful, trying not to upset anyone, etc. With these precautions, we were not likely punished, being flamed.

“Getting hundreds of angry mails from upset listeners” [30].

The mail that we wrote was the following:

Artist Wanted !
Hi everybody,

We are two computer science students really interested on Open Source Software related with Artistic Software. We are working launching an artistic software in the OSS world in SourceForge.net based on a source code evolving since more than 4 years. This software is called ImproSculpt (improsculpt.sourceforge.net). It is a tool designed to produce music automatically with algorithmic composition. It has been realized by Øvyind Brandtsegg. The main stable version is used by Flyndra (http://www.flyndresang.no/), "the musical sculpture". ImproSculpt has grown up and today it's the basis of a tool used by Øyvind for Live performances.

We wonder if any kind artist could be interested on drawing a logo for ImproSculpt. As ImproSculpt is an Open Source Software and come from Home-Software we cannot offer any amount of money. However, you would be participating on an OSS project and consecuently your name will appear on the project. And, we know that the motivations of OSS are not money

We know that ImproSculpt is not an important project and it just start to run, but it could be an opportunity for collaboration between artists and computer scientist with respect to the OSS philosophy

The only restriction Øyvind gave us is "no music notes". Appart from that, it's freedom for your imagination !

If you may be interested, know someone who might, or if you have the reference of a website, please let us know.

Thank you !!

We got two answers of two artists asking for more details. After offering the needed details to these artist we got quick answer with some logos. We just needed to ask Øyvind (our client) and he decided.

The last task was to plug the logo on ImproSculpt Wiki, in a good place, always visible.

For more information about the logo conversation see [67].

VI – Evaluation

VI.1 - Evaluation Method

Our project was to set up architecture and an environment for ImproSculpt development. It takes a long period to realize this project: the complete architecture was almost ready at the end of November! But it is even longer before we can receive the feedback of the users. Øyvind and the programmers who are interested in the project are busy, and timetables of artists are not always compatible with students. The only feedback (but the most important one) we could have, was from Øyvind's experience. As he is the first one interested in "ImproSculpt as an OSS", it was easy to convince him to install and configure the tools. According to his problems, reaction, suggestions and decisions, the second part of the project, "evaluation & improve phase" started.

The evaluation is based on three sources of data.

The first one is some questions we directly asked to Øyvind when we were close to the end. For the two main features of the project we asked him:

- *"What is good?"*
- *"What is bad?"*
- *"And what is missing?"*

The second source is the following: Øyvind will be introduced progressively into the administration tasks of an Open source project. Here is the list of the small "test" we want Øyvind pass before we left the project:

Wiki:

- Create a new section on the side-bar
- Include a new page and fill it.
- Protecting edit possibilities on one page thanks to edit, view, and upload passwords.
- Changing the administrator password
- Make a backup of the wiki

CVS:

- Set up the environment: Install TortoiseCVS, generate couple of SSH-keys with Putty, etc.
- Downloading a new "sandbox"
- Create a new module and upload a new version of ImproSculpt
- Modify a file and commit the result

Of course, our part of the project is to guide him. We wrote tutorials for almost all the tasks above. All this documentation is available on the Wiki and/or the report.

The last source of information will be us. Auto critic is essential to progress and going further.

Based on the results and feedback, we will know how hard it is to take the helm of the OSP. Adding to Øyvind answers, this will give us a better overview of the project quality.

But as we said before, this evaluation focused on the "technical" tasks, and as in all management projects, technique is only a small part. The most important impact of our project into ImproSculpt life would be visible in few months, even years.

VI.2 - Evaluation of Practical goal

First, we are evaluating the following questions: What is bad? What is good? And What is missing? About the two main points of our project. For doing this, we wrote an email to Øyvind just asking these three questions. We were willing to get all his opinions and feelings and these opened questions allowed him to do that. Now we are analysing his answer.

```
Thibault and Maxi,
here's some feedback:

I think the wiki is good. However, I don't have a lot of experience
with wikis and I have little to compare it to. From my point of view
the wiki gives some information about the project and provides links
to the code, installation instructions, and documentation
```

Although he showed his inexperience with this kind of tools, the aspects which he is talking about in this mail: information about the project, links to the code, etc. are almost all the support needed to use and even understand ImproSculpt OSS project. End users and new developers should be able to use it.

```
The one thing I think is "out of style" is the sentence
"And, isn't it the OSS philosophy?"
Yes, it is the OSS philosophy, and I don't see the reason for stating
it as this is an OSS project...
```

Here, in the above paragraph we can see one useful feedback. Almost all the times the critics are well received because shows the interest of the other part. Before receive this critic feedback on the home page of ImproSculpt Wiki appeared at the end of the description the following sentence: *"And, isn't it the OSS philosophy?"*. If we read this page (http://improsculpt.sourceforge.net) we can see that this statement is totally true. The result of this feedback was to skip this sentence.

```
The things that are missing are:
*How to install : for ImproSculpt Unstable
*Documentation: for ImproSculpt Unstable and Classic
I know I haven't provided you with details on this information, and as
such you might have considered it not to be part of the OSS
publication project.
The documentation for ImproSculpt Unstable could have been generated
by using doxygen on the sources. You could have done this.
But there exists no automated tool for extracting documentation for
```

```
ImproSculpt Classic, and I would say writing a full documentation for
Classic is outside the scope of your project.
```

Due to the latter advice from Øyvind about the necessity of upload Unstable ImproSculpt to SourceForge.net, we were not able to write a guideline for its installation. The result of that was to fill the required part.

We knew Unstable ImproSculpt includes Doxygen technology but we forgot link it to ImproSculpt Wiki. The result of this feedback was to generate this documentation, upload it to SourceForge.net and link it to ImproSculpt Wiki.

And about the documentation lack of ImproSculpt classic, we can see on the email that it does not exist any tools to extract the comments from the CSound source code. Thus, for future works on ImproSculpt one great project could be document by hand ImproSculpt Classic.

```
All in all your work have been of great help for me, as I wanted to
publish my code as an OSS project on sourceforge, and I also wanted to
start using CVS for my projects. The work you have done have enabled
me to do this. I would not have had time to learn these tools from
scratch, but with your help and guidance, I am now able to use the
tools (Wiki and CVS/Tortoise).
```

Finally with this paragraph we feel we have done a good work. During the third step of our project we have received more advice from Øyvind but we think these feedbacks are not relevant here. With this, all readers should have an idea about how we have dealt in the evaluation part.

The second part of the evaluation was to introduce Øyvind into our work. We thought to instruct him following the small tasks written above. The problem was the Øyvind´s free time to complete the whole set of tasks was not enough before the writing of this report. However, these tasks should be performed before Christmas.

To date, for ImproSculpt Wiki the following task are finishing:

- Create a new section on the side-bar
- Include a new page and fill it.

And we rest to do:

- Protecting edit possibilities on one page thanks to edit, view, and upload passwords.
- Changing the administrator password
- Make a backup of the wiki

Øyvind did not have any problem when he performed the above task.

However, for the CVS all the small tasks were performed and now Øyvind feels himself able to manage this tool. And all the mistakes that he made were not relevant, e.g. typing mistakes. Thus, we could say that this is a good feedback and CVS is set up well. What follows is an example of Øyvind´s mistakes in the CVS part.

```
When using tortoise,
I created an empty directory, right click in the new directory and select
"CVS checkout"
I filled out the form with the entries as suggested on the page:
http://imprOsculpt.sourceforge.net/pmwiki/pmwiki.php?n=Main.QuickStartGuide
I get this error:
***
In E:\Csound\csd\develop\modules\testSandbox: "C:\Program
Files\TortoiseCVS\cvs.exe" -q checkout -P flyndra
CVSROOT=:ext:anonumous@improsculpt.cvs.sourceforge.net:cvs/improsculpt

cvs checkout: CVSROOT
(":ext:anonumous@improsculpt.cvs.sourceforge.net:cvs/improsculpt")
cvs checkout: may only specify a positive, non-zero, integer port (not
"cvs").
cvs checkout: perhaps you entered a relative pathname?
cvs [checkout aborted]: Bad CVSROOT.

Error, CVS operation failed
***

I tried using my user name at sourceforge, and I also tried anonymous.

What could I be doing wrong ?
Is there a typo anywhere on my behalf ?

Oeyvind
```

The error was he wrote "anonumous" instead of anonymous and he confused the repository folder, he wrote */cvs/improsculpt*It's rather than */cvsroot/improsculpt.*

The logo task has been performed with our client and this mail could be its evaluation:

```
Yes, I like the second guy's logos the best.
I know this guy, Johan Bakken, he worked on the webpage for Flyndre too,
he's good.

I like the orange/yellow logo best,
thats the middle one on the top row.

The two color tones together do enhance the motion of the "bubble-waves".

Say hi to Johan from me.

best
Oeyvind
```

VI.3 - Auto-Critics

Another useful source of feedback is oneself. Sometimes it is important to stop and think about what is good and what is bad. But another issue even more important is to make this auto-critic at the end of the work in order to not repeat errors, and keep going on the good points.

Firstly, we did not feel very integrated into the project until the end of October. After our first and unique meeting with Øyvind, the project started to run. We have had many mistakes but the important issue is that we have learnt about it. The most important ones have been:

- *Planning:* It was difficult for us to establish a work planning without have a concrete project. But when we had it, we wrote a plan which we were not able to follow strictly.
- *Research Method:* The strange nature of our management project comparing with regulars project done for IDI has been a high barrier to work with a pre-establish research method like action research or case study. This is the first time that we are in contact with research issues and concretely with research methods and we have felt lost in this aspect from the beginning to the end. The good point here is that when you really do not make the things good (but you are aware of that), you learn even more than when you work well from the beginning. Thus, in the following research we will think in the way of work on a pre-establish research method.
- *Communication:* The communication between client and students in our project has been a little bit hard. Our client was very busy to organize meetings with and just the email communication is not always effective. But finally we have got an acceptable interchange of mails.
- *Research:* The best point of the project was the knowledge achieved. We have grounded very much in the OSS and Artistic Software literature and we think this report shows all the background that we have got. The high quantity and quality of references demonstrates it.
- *Practical Goal:* Finally the environment set up is a powerful set of tools waiting for new developers.

Conclusion

The process introduced by the Free Software Foundation more than 20 years ago is changing the face of Software development. But no just a process, the technologies, concretely internet is changing the way of working. Nowadays developers are distributed all around the globe. They use new communication tools, new development processes, sharing technologies… As the process is different, the result is also. Open Source software is very famous for their reliability, evolution and portability.

On the other hand software is becoming more and more artistic. The best representation is the iMac which bring more and more adepts thanks to an artistic and modern design/interface.

Could open source be a(the) link between computer scientists and artists?

References

[1] - http://en.wikipedia.org

[2] - www.fsfeurope.org/

[3] - Glass, R., 1999. Of Open Source, Linux and Hype. IEEE Software 16 (1), 126–128

[4] - Oeyvind website - http://oeyvind.teks.no/frames/Index.htm

[6] - http://www.flyndresang.no/

[7] - Karl Fogel. "Producing Open Source Software: How to Run a Sucessful Free Software Project". O'Reilly Media, 2005. Chapter 3.

[8] - Fred Brooks. "The Mythical Man Month". 1975

[9] - www.sourceforge.net

[10] - http://en.wikipedia.org/wiki/Software_licenses

[11] - http://en.wikipedia.org/wiki/Concurrent_Versions_System

[12] - http://en.wikipedia.org/wiki/Wiki

[13] - http://www.gnu.org/licenses/gpl.html

[14] - http://www.tortoisecvs.org/

[15] - http://www.putty.nl/

[16] - SourceForge webpage conditions :
https://sourceforge.net/docman/display_doc.php?docid=4297&group_id=1#acceptable_use

[17] - PHP information in SourceForge
https://sourceforge.net/docman/display_doc.php?docid=4297&group_id=1#php

[18] - MySQL information in SourceForge
https://sourceforge.net/docman/display_doc.php?docid=4297&group_id=1#mysql

[19] - PHP version used by SourceForge Server
https://sourceforge.net/docman/display_doc.php?docid=4297&group_id=1#overview

[20] - Backup guideline with Pmwiki :
http://www.pmwiki.org/wiki/Cookbook/SourceForgeServers

[21] - SourceForge Logo's Conditions :
https://sourceforge.net/docman/display_doc.php?docid=4297&group_id=1#logo

[22] - Robert M. Davison, Maris G. Martinsons, and
N. Kock. Principles of Canonical Action Research.
Information System Journal, 14(1):65–86, 2004.

[23] - Letizia Jaccheri and Håvard Mork, "Studying Open Source Software with Action Research"

[24] - http://en.wikipedia.org/wiki/Iteration

[25] - Yin, R. K. *Case Study Research, Design and Methods,* 3rd ed. Newbury Park, Sage Publications, 2002.

[26] - http://en.wikipedia.org/wiki/Scientific_method

[27] - http://en.wikipedia.org/wiki/Cathedral_and_the_bazaar

[28] - http://en.wikipedia.org/wiki/Fetchmail

[29] – http://www.idi.ntnu.no/~letizia/

[30] - Bergquist, M., and J. Ljungberg, "The power of gifts: organizing social relationships in open source communities", *Information Systems Journal*, Vol. 11(4), pp. 205-220, 2001. (16 pages)

[31] - Matchmaking festival http://test.teks.no/?lang_pref=en

[32] - http://en.wikipedia.org/wiki/Art

[33] - http://en.wikipedia.org/wiki/Aesthetics

[34] - Mauss,M. (1950/1999) *The Gift. The Form and Reason for Exchange in Archaic Societies.* Routledge, London.

[35] - Bonaccorsi, A., and C. Rossi, "Why Open Source can succeed", *Research Policy*, pp. 1243-1258, 2003. (16 pages)

[36] - Open Sources approved license http://www.opensource.org/licenses/

[37] - OSI http://www.opensource.org/

[38] - http://fplanque.net/Blog/itTrends/2004/09/29/choix_d_une_licence_open_source

[39] - http://www.onlamp.com/pub/a/onlamp/2005/06/30/esr_interview.html

[40] .- http://www.csounds.com/

[41] - http://www.python.org/download/

[42] - http://www.wxpython.org/download.php

[43] - http://en.wikipedia.org/wiki/OpenBSD

[44] - OpenSSH

[45] - Eric S. Raymond. The cathedral and the bazaar

[46] - https://sourceforge.net/docman/display_doc.php?docid=6445&group_id=1

[47] - http://sourceforge.net/docman/display_doc.php?docid=12834&group_id=1

[48] - http://sourceforge.net/docman/display_doc.php?docid=12834&group_id=1#front_page

[49] - http://sourceforge.net/project/showfiles.php?group_id=181045

[50] - http://en.wikipedia.org/wiki/Wiki#History

[51] - http://en.wikipedia.org/wiki/Wiki

[52] - http://moinmoin.wikiwikiweb.de/WikiEngineComparison

[53] - https://sourceforge.net/docman/display_doc.php?docid=4297&group_id=1#overview
[54] - http://www.pmwiki.org/
[55] - http://www.pmwiki.org/wiki/PmWiki/BasicEditing
[56] - http://en.wikipedia.org/wiki/Software_art
[57] - [Greene2004] Rachel Greene. Internet Art. Thames and Hudson, World of Art, June 2004. ISBN: 0500203768. Introduction pages 8 - 19, Chapter 2, Isolating the Elements, pages 73 - 118.
[58] - http://php.net/
[59] - http://www.gnu.org/copyleft/gpl.html
[60] - http://improsculpt.sourceforge.net
[61] - http://sourceforge.net/docman/display_doc.php?group_id=1&docid=4297#environment
[62] - http://improsculpt.sourceforge.net/pmwiki/pmwiki.php?n=PmWiki.Uploads
[63] - http://www.pmwiki.org/wiki/Cookbook/ABitModernSkin
[64] - http://improsculpt.sourceforge.net/pmwiki/pmwiki.php?n=PmWiki.DocumentationIndex
[65] - http://www.stack.nl/~dimitri/doxygen/
[68] - http://www.linuxforums.org/forum/classifieds/78170-artist-wanted.html#post407253

Appendix

Appendix 1

Back ups

ImproSculpt Wiki has designed a back up script in order to save the information of its web pages. The source code of this script is really easy and is the following:

```
#!/bin/bash
# Backs up wiki data.

# Perform the backup
cd backsup

mkdir backup-`date +%Y%m%d`
cd backup-`date +%Y%m%d`

tar zcvf backup-`date +%Y%m%d`.tar.gz \
 --exclude /tmp/persistent/improsculpt/pmwiki.d/sessions \
 /tmp/persistent/improsculpt/pmwiki.d/\
 /home/groups/i/im/improsculpt/htdocs/pmwiki/{wikilib.d,wikilib2.d}

# Move the especial directory sessions to the place of the backup
cd /tmp/persistent/improsculpt/pmwiki.d/sessions/

mv * /home/groups/i/im/improsculpt/htdocs/backsup/backup-`date +%Y%m%d`

# Move wiki pages out of the temp space.
mv -f /tmp/persistent/improsculpt/pmwiki.d/wiki.d/* \
 /home/groups/i/im/improsculpt/htdocs/pmwiki/wikilib2.d
```

It has to be executed regularly. It depends of the security that you want to achieve . This file is into `/home/groups/i/im/improsculpt/htdocs/` and it saves all the backups into `/home/groups/i/im/improsculpt/htdocs/backsup/`.

Password system

Nowadays ImproSculpt Wiki has protected by passwords. This passwords are the followings:

- *Administrator Password:* (**improSculpt06**) You can do everything with this password. (The Upload and Attr passwords are improSculpt06 too).
- *Editing Home Page Password:* (**oyvind**) In order to protect the information about ImproSculpt overview we have establish a password for edit this page.

All the passwords include inside the default passwords set are located in `/home/groups/i/im/improsculpt/htdocs/pmwiki/local/config.php` file. You can not read it because these passwords are encrypted and look very strange. For change some all these passwords you must go to the following internet address:

http://improsculpt.sourceforge.net/pmwiki/pmwiki.php?n=PmWiki.PasswordsAdmin?action=crypt

Type the chosen password. Then you will get your password encrypted and you must copy it on the latest file.

All the wiki pages on ImproSculpt Wiki can be protected under a password. To do this, you need to know the Attr Password or the Administration Password. If you want to protect one particular web page you must type in the address text box of your navigator the same address of the page followed by: `?action=attr`

Skin

ImproSculpt appearance is easy to change by changing this skin. Basically you must go to the following internet address in order to choose one skin:

http://www.pmwiki.org/wiki/Cookbook/Skins

Then, you must download the skin files of your choice and copy it on `/home/groups/i/im/improsculpt/htdocs/pmwiki/pub/skins/` inside of a folder with the name of your skin.

Finally, into `/home/groups/i/im/improsculpt/htdocs/pmwiki/local/config.php` file you have to change the `$skin` variable with the name of your skin.