Customer Driven Project - Netlight

Ivo Dlouhy, Martin Havig, Øystein Heimark, Oddvar Hungnes

September 17, 2012

Abstract

Imagine the old minimachine systems where the end user works in a domain specific application in a black-and-green terminal. She is super efficient, jumping between windows with short cuts and everything is "in her fingers".

This is then replaced with a web-frontend and a mouse. Everything is "wrong", the design makes the usage patterns locked and she gets "mouse sickness" from point-clicking every command.

The task is to modify a web-application and add a scripting-console where the end-user can enter commands into a DSL - similar to the older interface. When commands are run, the results are shown both in the console and the web-interface. The use can still use the mouse and navigate as usual.

The target is a simple proof-of-concept in order to research any changes to the API, the potential of the method as well as evaluate the usability of scripting languages/DSL and API.

NTNU – Trondheim
Norwegian University of
Science and Technology

Contents

List of Figures

Chapter 1

Introduction

1.1 NTNU

1.2 Netlight

Netlight, our customer, is a Swedish IT- and consulting-firm. Their field of expertise is within IT-management, IT governance, IT-strategy, IT-organisation and IT-research. They deliver independent solutions based on the customers specs. With the broad field of knowledge they can handle whatever tasks presented by their customers. They reach this goal by focusing on competence, creativity and business sense. [1]

1.3 General information about project

The project is the making of the course TDT4290 Customer Driven Project. This is a mandatory subject for all 4th year students at IDI and aims to give all its students experience in a customer guided IT-project and the feel of managing a project in a group. The customer assign the group a task which makes the project close to normal working life situation.

This is a proof of concept project. The underlying task is to research and develop a system where power users can benefit from a console. The concept aims to ease the workload of a power user who is working with object editing, and to see how the efficiency of a console might prove to improve the work. The power user is usually a user who often works with the system over a longer time, and is in depth familiar with the system. We will research already existing systems of this kind, and look at the possibilities and advantages of such a system in a chosen domain.

1.4 Contact information

Person	**Email**	**Role**
Ivo Dlouhy	idlouhy@gmail.com	Team member
Martin Havig	mcmhav@gmail.com	Team member
Oystein Heimark	oystein@heimark.no	Team member
Oddvar Hungnes	mogfen@yahoo.com	Team member
Peder Kongelf	peder.kongelf@gmail.com	The customer
Stig Lau	stig.lau@gmail.com	The customer
Meng Zhu	zhumeng@idi.ntnu.no	The advisor

1.5 Goals

1. Create a working prototype of a system where a scripting console is embedded into a modern web interface. The console should provide access to viewing and modifying the underlying data objects of the system's domain via a DSL.

2. Investigate the ramifications of the added functionality, in terms of usability and technical aspects.

3. Provide extensive documentation and a successful presentation of the end product.

[1] `http://www.netlight.com/en/`

1.6 Planned Effort

Role	Description	Assignee
Team leader	Is responsible for administrative tasks and makes the final decisions.	Ivo
Scrum Master	Shields the development team from external distractions and enforces the Scrum scheme.	Ivo
Customer Contact	Handles communication with the customer. The customer should contact this person regarding general requests, questions and reminders.	Ivo (backup Martin)
Advisor Contact	Handles communication with the advisor. The advisor should contact this person regarding general requests, questions and reminders.	Ivo (backup Martin)
System Architect	Is responsible for the system architecture including distinctions and relations between subsystems and general code design choices.	Martin
Code Master	Overall responsible for code management and structure. Managing branches in Git repository.	Oddvar
GUI Designer	Is responsible for the layout and design of graphical user interfaces.	Oddvar
Test Manager	Is responsible for testing including unit tests, integration tests and usability tests.	Øystein
Report Manager	Is responsible for delegating and overseeing work on the project report.	Martin
Customer Representative	Participates in regular meetings to discuss the progress, project status and future tasks. Represents the customer.	Peder Kongelf
Customer Technical Advisor	May be consulted about technical aspects of the project.	Stig Lau
Advisor	Serves as a one-man steering committee for the project.	Meng Zhu
Meeting Secretary	Is responsible for making sure notes get written and sent after each meeting with the advisor and customer.	Oddvar
Quality Assurance Manager		Øystein
Weekly Report Writer	Is responsible for finalizing the weekly report(s) for the advisor and customer, and getting these delivered for approval. Also responsible for meeting agendas and their delivery.	Øystein
Time Keeper	Responsible for making sure that everybody is logging their work, and logging team activities.	Oddvar

1.7 Goals

1.8 Planned effort

1.9 Schedule of results(Milestones, deliverables, sprint deadlines, etc)

Chapter 2

Project management

2.1 Project plan

2.1.1 Sponsor/customer

2.1.2 Background

2.1.3 Gantt diagram

2.2 Team structure

2.2.1 Roles

2.3 Risks

2.4 Architecture

2.5 Scrum

2.6 Quality Assurance

Chapter 3

Preliminary Study

3.1 Concept

3.2 Constraints

3.2.1 Time

3.2.2 x

3.3 Feasibility study

3.4 Version control

3.4.1 git

3.5 Development language and technologies

3.5.1 Google Drive

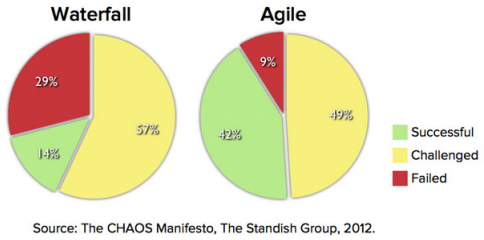
3.6 Development Methodology

3.6.1 Agile vs Waterfall

The waterfall method focus on planning the future in detail. It follows the principle of "Big Design Up Front". It relies on the fact that you are able to report exactly what features that are going to be implemented and tasks are planned for the entire length of the project. It forces you to specify all the requirements early in the development, when you actually know the least about the project and the problems that are to be solved. The rationale behind this is that time spent early on making sure requirements and design are correct saves you much time and effort later. A development team using the waterfall method will only consider to implement the most valuable changes, as changes in this process are time consuming and often requires that completed work is started over. The method places a lot of emphasis on documentation.

Agile methods, as opposed to the predictive methods, are designed to plan for changes in the requirements and features of a project. It emphasises on working code as primary measure of progress, instead of extensive documentation of for example the requirements. Agile methods consists of iterative and incremental steps in the development process, where requirements and solutions evolve through the course of the project. Requirements are bound to change, either because the customer didn't understand the problem in the beginning or because they would like to add new features. Agile methods facilitates the ability to accommodate these changes. Most agile methods includes delivering a working product in incremental stages, and gives the customer something to relate to during the developments process.

The CHAOS Manifesto is a survey published by the Standish Group each year and it measures the success of IT-projects. It divides the projects into 3 groups; Success, meaning it completed on time and budget, with all features and functions as specified. Challenged, meaning it completed, but was over cost, over time, and/or lacking all of the features and functions that were originally specified. Failed, meaning the project was abandoned or cancelled at some point and thus became a total loss.



Source: The CHAOS Manifesto, The Standish Group, 2012.

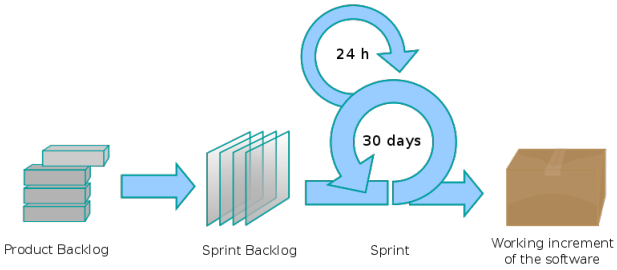
As the figure illustrates, agile methods although not perfect by any means, more often result in products that are successful(the method used for measuring the success of a project is to some degree debated, but the results serves a purpose nevertheless).

3.6.2 Agile Methods

There exists a lot agile methods for software development, and although all of them follow the basic principles of agile development, they differ in a lot of areas. Following is a detailed description of three different agile methods.

Scrum

Scrum is an iterative, incremental software development model with several short sprints - complete small sets of tasks each sprint.



The Roles:

- The Scrum master, who is responsible for leading the process and to enforce the Scrum rules onto the team. He has to make sure that the development team does not overestimate what they can handle during one sprint. He leads the scrum meetings and enlightens and handles obstacles that may appear.
- The product owner, represents the stakeholders and is the voice of the customer.
- The development team, is responsible for delivering potentially shippable product increments at the end of each Sprint. A development team is made up of 3–9 people with cross-functional skills.

The Sprints:

- Normally last from 7 to 30 days .
- Starts with a planning meeting, where tasks are identified and goals for the sprint is set.
- Product owner tells the team what tasks should be done in the sprint.
- The tasks comes from a prioritized list of requirements called the backlog.
- The team determines what is possible based on this and records this in a sprint backlog.
- The goals should not be changed during the sprint.

The Scrum process is well suited for projects where its difficult to plan too far ahead, where at least some of the aspects of the project are unknown. Its a versatile process which is gives you the ability to handle changes in the requirements and demands from the customer. It allows for the developers to work on different parts of the project at the same time. The design, requirements of the system are not set in stone from the start, and are allowed to evolve during the process. The process delivers unfinished versions at the end of each sprint, which gives the customer a chance to try the system and give continuous feedback to the developers.

The Scrum process is somewhat complex, and it will take time to properly learn and execute the method. You also have to decide on what type of Scrum you are going to use, as there exists multiple forms of Scrum. This can prove to be a time consuming process. And even though the team members know Scrum, they will have to learn the version of scrum decided upon, if it turns out to differ from the one they are used to. [1]

[1] `http://en.wikipedia.org/wiki/Scrum_(development)`

DSDM Atern

DSDM(Dynamic System Development Method) is an agile software development method, and it was originally meant to provide some discipline to the Rapid Application Development method. The most recent version was launched in 2007 in an effort to make DSDM tool and technique independent, and its called Atern.

DSDM is an iterative and incremental approach that embraces principles of agile development, including continuous user/customer involvement. It enforces you to deliver incremental versions of the product to the customer, where the main criteria of acceptance is that it meets the current business needs of the customer. It follows the principle that it is always better to deliver something "good enough" early than to deliver everything "perfect" in the end.

DSDM as a method fixes costs, quality and time at the beginning of the project. Through a prioritization method called the "MoSCoW Method", with musts, shoulds, coulds and won't haves, it adjusts the scope of the project to meet the given time frame. This allows the development team to focus on the critical functions of the system rather than delivering a perfect system. The method puts a strong focus on actively involving the customer in the development, and continually confirm the solution.

The principle-list of DSDM is quite long and complex. For a team that is not experienced in using the method, the process of learning the method will be time consuming. Also, always having to display the progress to the customer can be time consuming and hinder the development effort. Testing is central and shall be done through the whole development process. [2]

Extreme Programming

Extreme programming, hereby referred to as XP, is an agile method designed to reduce cost of changes in requirements by having multiple short development cycles. It includes elements such as pair- programming, extensive code review and unit testing of all the elements of the code. It emphasises frequent communication with the customer and between the developers.

The method embraces changes in the requirements of the project, and it doesn't attempt to define a stable set of requirements at the beginning of the project. In XP a representative for the customer is always available on site to answer any questions the developers might have. It also focuses on frequent releases of working code which serves as checkpoints where the customer can add new requirements.

XP puts a lot of focus on the code of the project, the advocates of XP argues that the code is the only truly important product of the system development process. XP as a process does not produce a lot of written documents during the development of the project. In XP programmers are expected to assume a unified client viewpoint and focus on coding rather than documentation of compromise objectives and constraints. [3]

3.6.3 Conclusion

If all of the requirements of this project were known in advance and provided by the customer, or the features of the finished product was known and unlikely to change, the waterfall method might be the way to go. But this is a prototype, proof of concept type of project where very little is known about the final product. We were certainly not presented with a finished set of requirements at the beginning of the project, and the requirements we settle on before we start the implementation are also more than likely to change during development. These kind of changes in a waterfall process will be time consuming. Thats why we think that an agile development method will be the best choice for this project.

All of the agile methods described above exhibits properties that will come in useful in this project. They are all based on iterative and incremental development steps, delivering prototypes for the customer to test after each step. This will give the customer a chance to try out unfinished versions of the product and give continuous feedback throughout the project. It can also help the customer to identify new features that they would like to add. They also allows the customer to decide which features to implement in each step, ensuring that the final product will contain the features the customer really need.

The agile methods embrace changes in the requirement and provides ways to handle those changes. They also encourage tight and continuous communication with the customer, which is important to be able to deliver a product that the customer is satisfied with

Of the three methods the group members are most familiar with the Scrum process. The DSDM method is complex and will take a considerable effort to learn and execute correctly. As none of the group members have used DSDM, and we have a limited amount of time in this project, DSDM is of the table.

XP puts a lot of focus on the code, and delivers a minimal set of documents. We are to write an extensive report about the project, and document every part of it, including compromises and assumptions made. The amount of code in this project will be limited and none of the team members have any experience in working with XP. As a result, XP seems like a bad fit for our project.

[2] `http://en.wikipedia.org/wiki/DSDM`

[3] `http://en.wikipedia.org/wiki/Extreme_Programming`

The Scrum process does not put as much focus on documentation as the waterfall process, but we think that the amount of documentation produced during the Scrum process will be satisfactory for the report. It will take time to learn how to execute Scrum properly, but since all of the group members are familiar with the basics of process, we think it won't be too time consuming and worth the effort. The final decision then is to use Scrum as a development method for this project.

3.7 Software Testing

3.7.1 Testing Methods

The purpose of software testing is to uncover software bugs in the system and to document that the system meet the requirements and functionality that was agreed upon for the system. Testing can be implemented at any stage in the development process, traditionally it is performed after the requirements have been defined and the implementation is completed. In agile development processes however, the testing is an ongoing process. The chosen development methodology will in most cases govern the type of testing implemented in a given project.

Software testing methods are traditionally divided into white- and black- box testing. They differ mainly in how the test engineer derives test cases.

White- Box Testing

White- box testing focus on the internal structures of a system. It uses this internal perspective to derive test cases. White- box testing is usually done at unit level, testing specific parts or components of the code.

Black- Box Testing

Black- box testing handles the software as a black- box, meaning it observes the functionality the system exhibits and not the specifics on how it is implemented. The tester only needs to be aware of what the program is supposed to do, he doesn't need to know the specifics on how the functionality is implemented in the code. Black- box testing checks to see if the functionality of the program is according to the agreed upon requirements, both functional and nonfunctional.

Test Driven Development

The principle behind TDD is to develop the code incrementally, along with test for that increment. You don't move on until the code passes its test. The tests are to be written before you actually implement the new functionality. The process helps programmers clarify their ideas of what a code segment is actually supposed to do. The process is often used in agile development methods. Benefits from TDD include:

- Code coverage, every code segment should be covered at least one test.
- Regression testing, check to see if changes in the code have not introduced new bugs.
- Simplified debugging, when a test fails it should be obvious where the problem lies, no need for a debug tool.
- System documentation, the tests themselves act as a form of documentation that describe what the code should be doing.

Automated Tests

Automated offers the ability to automatically do regression tests, i.e. testing to uncover if any new code has broken a test that previously passed. If we opt for manual testing regression testing will be very time consuming as every test done so far has to be done over again. With an automated testing framework this job will be a lot easier as you can run a great number of tests in a matter of seconds. Most development languages offers libraries for automated testing.

3.7.2 Testing Levels

Testing can be done at many different levels and in different stages in the development process. Following is the most common partitioning of testing levels and a description on each of them.

Unit Testing

Unit testing aims to check specific components, such as methods and objects. Typically you will be testing objects, and you should provide test coverage of all the features of that object. Its important to choose effective unit test cases, that reflect normal operation and they should show that the specific component works. Abnormal inputs should also be included to check if these are processed correctly.

Component Testing

Tests bigger components of the system, and their interfaces(communication with other components). Made up of several interacting objects. Component testing is mainly a tool to check if component interfaces behaves according to its specification.

System Testing

In a given development project there may be several reusable components that have been developed separately and COTS systems, that has to be integrated with newly developed components. The complete system composing of the different parts is tested at this level. Components developed by different team members or groups may also be integrated and tested at this stage.

Release Testing

Release testing is the process of testing a particular release of the system that is intended for use outside of the development team. Often a separate team that has not been involved in the development perform this testing. These kind of tests should focus on finding bugs in the complete system. The objective is to prove to the customer that the product is good enough. This kind of testing could either be based on the requirements of the system or on user scenarios.

User Testing

This is a stage in the testing process in which users or customers provide feedback and advice. This could be an informal process where end- users experiment with a new system too see if they like it and that it conforms to their specific needs. Testing on end- users is essential for achieving success in a software process as replicating the exact working environment the system will be used in is difficult to achieve during development. The end users can help provide feedback on how specific functionality will work in an actual work environment.

Another form of user testing involves the customer and its called acceptance testing. Its a process where the customer formally tests a system to decide whether or not it should be accepted, where acceptance implies that payment for the system should be made. Acceptance testing is performed after the release testing phase.

3.8 Code conventions

3.9 Similar solutions

Chapter 4

Requirements

4.1 Usecases/user stories

4.1.1 Planning

4.2 Sequence Diagrams

4.3 Prioritization

4.4 Functional Requirements

4.5 Nonfunctional Requirements

4.6 Test Plan

Chapter 5

Overall System Design

5.1 Database

5.2 GUI

Chapter 6

Sprints

6.1 Design

6.2 Planning

6.3 Duration

6.4 Goals

6.5 Testing

Chapter 7

Testing

Chapter 8

Conclusion

Chapter 9

Evaluation