CSC 505: Spring 2014
Quiz Number 2
March 6, 2014

Meeting Time: 2:00 - 5:30 pm.
Location: Tyler 126 (Graduate Lounge)
Scribe: Andrew Poirier

Team Members:

1. Omar Rivera
2. Andrew Poirier
3. Daven Amin
4. Rick Rejeleene
5. Qutaiba Albluwi
6. Younghun Chae
7. Tripti Garg

What are the main features of rapid software development and how does rapid development differ from other, more heavyweight processes?

The main features of the rapid software development are specification, design and implementation. In this there is no detailed specifications but it does include a very non-detailed design document. With this being said, the system is spread out and developed in a series of increments with end users and stakeholders interacting and evaluating each increment step of the process. In the user interface development phase, the design is created quickly with drawings and icons. Once these designs are created, the system has the ability to create a web based interface for a browser or an interface for a specific environment. When compared to heavyweight processes, the rapid development has smaller incremental steps which involves the customer interaction and input on each increment where in heavy weight the projects are being worked on in different teams where it is hard for the customer to sit down with all of the teams in the same place and have a demonstration. This makes it harder for the developers to completely meet the customers specifications on each step.

Discussion :
Does anyone use this in real life ?

- Has performed this with scrum and each week there were goals for every week. With something simple it can be solved in one - two weeks. In a huge project
- Proof of concept by end of summer with limited funds, time and labor. Used micro milestones. Tuesday : what we are going to do and thursdays: what actually got done. Product was out of the door and to the customer by the end of the summer. Working on a 10 year process now and if the user does not like the final product in 2020 then the
- New idea pops up you have to develop it as soon as possible. If waiting too long then you might lose it or not have copyright rights to the software because it took too long.

- Do something everyday. Make a website and show everyday what little tasks that were completed so the customer sees the process day by day.

What are the strengths and weaknesses of rapid development?

The main strengths of rapid development are twofold: the delivery of services to the customer is accelerated and system end-users are engaged throughout the iterative development process. The accelerated delivery of services delivers high-priority functionality to customers earlier than would be the case in a traditional software development process. This allows customers to get results and make changes to the requirements in the early stages of development. By engaging system end-users throughout the development process, users interact with the software as it is being developed and provide feedback to developers. This increases the likelihood of the system meeting all requirements. It also puts the developers and end-users on the same page so that there are no surprises at the end of the process.

The main weaknesses of the rapid development process are related to management problems, contractual problems, validation problems, and maintenance problems. Management is difficult due to the lack of regular deliverables and the possible use of unfamiliar technologies to meet the rapid delivery of software. This may pose an issue if the existing staff of a project manager lack skills in these technologies. The second set of issues are related to contractual problems. Most contracts are based off of the system specifications and it is a lot harder to create a contract off of a system design. Validation problems can also cause issues because they are concentrated towards the demonstration of the system. In systems developed using interactive development processes, the documentation may be minimal. This makes independent validation of the system difficult. The final weaknesses of rapid development are related to maintenance problems. The software may be difficult for anyone apart from the original developers to understand due to the rapid and continual change.

Discussion :

- Academia work you need to develop and use your own tools. Approach will be very different and would not use an agile approach.
- Creates a problem and sometimes does not fix the methods we are developing the software. Sometimes all of the steps do not need to be taken when developing software and this is why most of the time the systems fail.
- Three projects. traditional waterfall process and we can work on all three at one. With agile you have to work on only one project at a time.

For what types of projects is it best suited?

The rapid software development process best suits the development of small or medium-sized business systems as well as personal computer software products. When getting into multi-level larger systems the teams are often in different places which makes it harder to make interactions with different parts of the system like hardware and software since they are in different places working on the same system.

Discussion :

- Smaller and medium companies are a lot better because it is more consumer friendly

- If you start working on a project that was long term and dig in half way though there is not much documentation and explanation to how the code actually works. When it is on smaller scaled and on a smaller time schedule then there is quick delivery and you have no choice and have to roll out the product which would be easier because it was smaller based.

How is testing different in rapid environments?

Testing in rapid environments is difficult due to the lack of defined specifications how ever Extreme Programming uses four main features to focus on testing. These four features are test-first development, incremental test development from scenarios, User involvement in the test development and validation, and the use of automated test harnesses.

Discussion :

- Not much testing really happens unless it is in XP. In reality there is not many specifications to test on and that there is not much time to test.

How about documentation, how is that different?

1. It is not cost-effective to produce lots of system documentation while the developed systems are changing so quickly. Thus, Rapid Development spends time for implementation rather than documentation.
2. Customers can confirm the requirements by using prototyping instead of documentation.
3. There is no detailed specification and design documentation is minimised.
4. Prefers face-to-face, over written documents.
5. Very little written documentation during the development, but requires a significant amount of post-project documentation.
6. Progress can be hard to judge and problems hard to find because there is no documentation to demonstrate what has been done.
7. There is a validation problem because there is no specification. Thus, it is hard to know what the system is tested against.
8. There is a maintenance problem when it needs new requirements because there is no specification.
9. In Extreme Programming, user requirements are expressed as scenarios or user stories.
10. The scenarios or user stories are written on cards and the development team break them down into implementation tasks.
11. The documentation in Rapid Programmings does not mean only the documents, but also means prototyping. There is a type of documents that is called compound document. This is a document with active elements, such as a spreadsheet, pictures, digital videos, digital audio, and other multimedia features, that allow user interactions. Each active element has an associated application which is invoked when that element is selected. The document itself is the integrator for the different applications. A prototype can be created by developing a compound document.

Discussion :

- #7 & #8 - Since there is no documentation or specification when testing a system it is very hard because there is nothing to go against or test against while the test process is happening. It is getting documented while prototyping so when there is a test phase there is very little to go against.

What extreme programming practices do you think are useful or promising? Why?

1. The full-time and throughout involvement of the customer in the development process grants the development team to receive instant feedback. It also cuts off the overhead of communicating with the customer. Further, it contributes to quick and more effective design of testing scenarios. Such developer-customer model might be integrated in other standard software development schemes, depending on the type of the software and availability of budget.
2. Putting emphasis on scenarios and using "story cards" enhances the customer readability of the software documentation which in return contributes to better understanding of the system by the developers. In some software projects, the extensive use of computer science jargon by the developers tend to cause ambiguities to the customers, and using the "story cards" seems to offer a simple and effective solution to the problem.
3. The model of using small releases leads to better customer satisfaction, especially if the customer is "demanding".
4. In software companies, ownership and empowerment is an issue that face many programmers/developers. When the system fails or a difficult bug is found, it is not unusual to point fingers at specific programmers. At the same time, at the successful release of the software, the senior developers/administrators normally get much of the credit. In XP, the concept of collective ownership overcomes this issue. It empowers programmers at all levels to propose and immediately implement creative solutions.
5. The "Test-first development" concept which is an innovative and powerful feature of Extreme Programming can be effectively adopted in other software development schemes. Whenever there is a clear relationship between a requirement and the code, "test-first development" offers an effective solution that would reduce the cost of the testing process.

Discussion :

- Full time customer very difficult. there was a person that was put in charge that is in charge of the whole project is a huge help if the budget allows.

- Sometimes a lot of people do not know what you are trying to say so when the testing happens.

- Ownership - when the software fails the bug is pinpointed to it. When it is created all together then everyone will get an even blame.

- Having little things on your website on a daily update gives the company a really good customer satisfaction because the company is reporting to the customer every day.

What is the role of prototyping in rapid development environments?

Prototyping involves construction of models of the system in order to discover new features. It is an initial version of a system to demonstrate concepts and try out designs. Prototypes architects building are generally used to help uncover and confirm customer requirements.

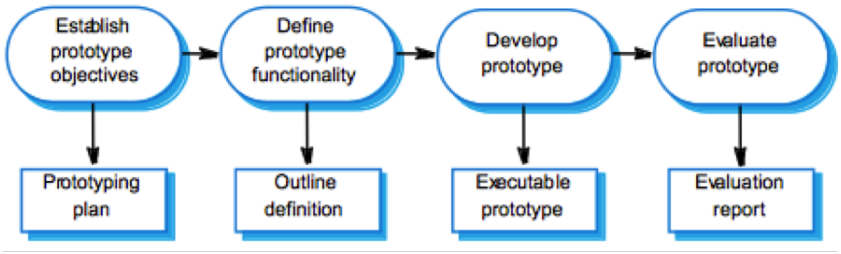

Figure 1: Prototyping
Figure 1
Cited : http://www.cs.ccsu.edu/~stan/classes/CS530/Slides/SE-17.pdf

Prototypes can be working and nonworking models.

- Working models can include working code in the case of software systems and simulations or to scale prototypes for non software systems. The code in working model can be deliberately designed to throwaway or it can be reused i.e non-throwaway.
- Non working models can include storyboards and models for user interface.
- In Agile methodology mostly throwaway Prototype is used.

Throwaway Prototype :
Prototypes should be discarded after development as they are not a good basis for a production system or get corrupted by frequent changes.

- It may be impossible to change the system to meet non-functional requirements of the system
- Prototypes are normally undocumented
- The prototype probably will not meet the organizational quality standards.

Advantages of Prototype :

- Give better clarity of concept of system to developers and customers.
- Most accurately fulfills customer's needs. During requirement process, help with requirement elicitation and validation.
- Help in reducing development effort. While developing software system, prototype is referred regularly.
- In testing process, can be used as to run back-to-back tests.
- Help in improving design quality.

Discussion :

- Throw away prototype - there are somethings that are just prototyped to showed and then throw them away. When working with a project based off of networking you would want to work on something that was hard coded and only have a few simple basic tests that the product

will work.

Other aspects discussed? Explain?

Is pair programming effective or redundant?

- Pair programming is usually when working with a mentor and mentee. The mentor is there to help the mentee to find the bugs or major issues that occur in the code. There would be two people working since the start.
- Solves the problem of someone just sitting around and having only one person programming. Switch off every ten minutes. There is a silent agreement between the pairs and ended up like single programming.

What is the benefit of compound documents(multimedia) compared to prototyping?]

- You would be showing something to them that would have media in it would be a lot quicker than making a correct working prototype. The customer can redo any idea that they might see in the compound document. Some projects you can not simulate and make a prototype

When prototyping what is the best approach to take based on the different aspects of the system (ie. database, gui ?)

- On a database you want to prototype how the data is sent, stored, and processed when it is sent and retrieved from the system. On a gui system the prototype should basically just execute and navigate to the pages or simple pop up messages that they system might make if a button is pressed. Ie. If a cancel button is pressed a dialog might pop up asking the user if they really want to exit but they system might not exit but the button and navigation options are all working. .