Process Methodology

for

Wegmans Deli Kiosk

Version 1.0

Prepared by DELI-cious Developers

Rochester Institute of Technology

September 15, 2013

Table of Contents

1. Process

1.1 Choice

The team researched and discussed many choices for our process methodology. One such methodology was the traditional waterfall model. After hearing the recommendation from the Software Engineering department, this seemed like a logical choice for our project, as it would give the project a set series of steps to follow, along with concrete deliverables at each step. However, what we didn't like was the fact that this model is very rigid and inflexible. It is expected that you've done an adequate job eliciting and refining your requirements before you can move into the architectural design phase and there isn't the opportunity to move back a phase if problems are discovered in your requirements or design. Additionally, due to the fact that our requirements are unclear and may have to change when more information is available, we feel that the traditional waterfall model is not the best fit for the project.

Another methodology that was researched and ultimately rejected by the team was an agile approach to the project's process. The project sponsor mentioned that they use the agile methodology, Scrum, for their technical projects. After researching and discussing as a group, the team decided that this would not be a great fit for this project. Although agile methodologies are growing in popularity and use in the "real-world", there are many reasons why it just would not be a good choice for this project. In the Scrum methodology, there are daily "stand-up" meetings, in which every member of the team is required to attend, in which each member discusses what they did the day before, what they will do today, and any obstacles blocking their path. Due to the fact that we are a student team with each member having very different schedules, having a daily "stand-up" meeting would not be possible and thus the process would break down. Also, the team doesn't have enough experience with the Scrum methodology to make it a viable option for this project.

A third approach that was researched was the Spiral methodology. This process was one that the team researched and almost used as our chosen methodology. We liked the fact that the process is based heavily on risk assessment. Since our requirements are unclear and likely to change, being able to assess and mitigate risks would be vital to this project. Additionally, this model produces software much earlier than a traditional waterfall. Due to the fact that this project will have a touch screen interface component, being able to produce a functional prototype early in development would allow the team to get crucial feedback about the user interface and make necessary improvements. However, the team ultimately decided that this process model would not be the best fit for this project due to the fact that none of the team members have any experience using this model. Also, risk management is very important to this methodology and would require a great deal of skill to properly determine all of the risks and mitigate them. Since, we don't have any risk management experts on the team, it was decided that the team would not use this model.

In the end, the team decided to use the **Incremental Build Model**, which will be described in the next section.

1.2 Description

An incremental build model utilizes an iterative development process, building off of previous progress. This model combines the elements of the waterfall model with the iterative philosophy of prototyping. At the end of each stage, more functionality and features are added until the product is ultimately finished. The success of this model is tied closely to the fact that it involves both development and maintenance.

Because of various unknown aspects in the project, we feel it is beneficial to utilize prototypes, build on feedback and previous experience for each revision of the project. Additionally, there are many constraints placed on the system by the sponsor. Many of these constraints involve technologies that the team has not worked with before. By doing staged releases, the team will be able to generate a working prototype early in development, which will help the team to ensure that the final product will be able to work within the specified constraints.

Unlike scrum sprints, where individual features are prioritized, the focus will be on mitigating known risks, and evolving the working prototype during each iteration. This will help both the development team, and the stakeholders to understand where the project is going, and adjust accordingly.

1.3 Lifecycle

Initial Planning

At the beginning of the project, there is an initial planning phase. During this phase, the concept of the final product is determined, along with the vision and scope for the final version of the product. Additionally, the team will discuss and agree to the documentation and artifacts that will be worked on and delivered throughout the duration of the project.

The incremental build model involves a series of waterfalls, each of which adding to the final release of the product. Each waterfall involves a combination of the following steps:

Requirements

At this stage of the iteration, the team gathers the necessary requirements from the customer and/or prospective users. There are many ways the team can go about doing this, including focus groups, customer discussions, user surveys, usability studies, etc. It is from the gathered requirements, that the team determines how to design the final deliverable for the current iteration. These requirements will be gathered in the form of a Software Requirements Document.

Planning
At the beginning of each cycle, there is a planning phase. It is here that team determines the scope and the deliverables required for this part of the project. They also determine the risks that are involved for this stage of the final product and how to mitigate these risks.

Analysis & Design
During this stage of the iteration, the team analyzes the requirements gathered in the previous stage and creates their designs to satisfy those requirements. Since the subsystems and components tend to drive development in this process model, this stage is very important. The team will assess the risks and make alterations to the design in order to address these potential risks. These designs will be gathered in the form of an Architectural Design Document, with many accompanying diagrams.

Implementation
This stage is commonly referred to as the construction phase. In this stage, the team writes code based on the designs created in the previous stage. During this stage, the team may also develop plans for testing and generate unit tests, but the main focus of this stage is write functional code.

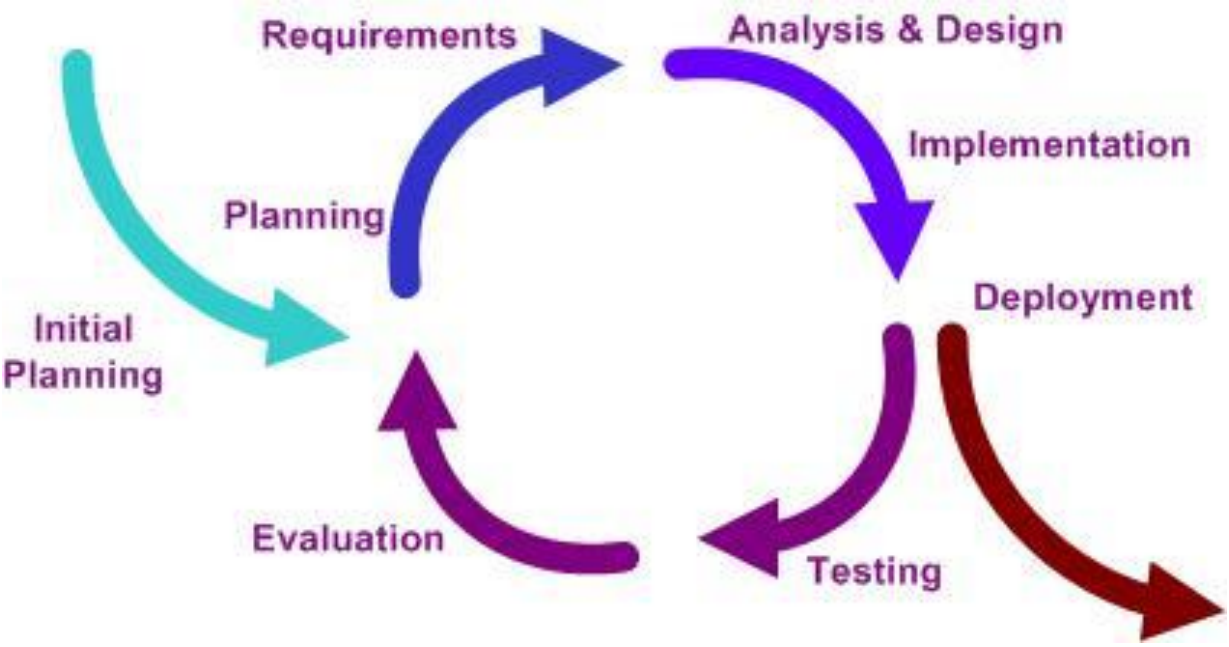
Testing
In this stage of the iteration, the focus shifts to running tests. There are many forms of testing that goes on during this stage of the iteration. These forms include unit, acceptance, usability, and many others. The results of these tests help to shape the next iteration.

Evaluation
During this stage of the iteration, the customer is given the working product for the current iteration. The customer tests out the product and gives feedback that can be used in the next iteration.

Deployment
During the last iteration of the project, the product is deployed. It is here that the final product is delivered to the customer along with any and all documentation.

1.4 Diagram

The following diagram illustrates the various stages in each iteration of the model. Each of these stages is described in the preceding section.



2. Pros

Below is a list of the advantages to using this process methodology for the project.

- Earlier visibility to customer or project progress as compared to the Traditional Waterfall Model.
- Customer can respond to features and review the product for any needful changes.
- Due to the fact that smaller changes and additions are made during each iteration, it is easier to test and debug during each iteration.
- Product is developed in iterations, with each one building on the last. This allows for a functional prototype earlier in development.
- Customer gets software earlier than a traditional waterfall.
- Test early and often
- Deliver code early and often
- Great when requirements are unclear.

3. Cons

Below is a list of the disadvantages of using this process methodology for the project.

- As additional functionality is added to the product, problems may arise related to system architecture, which may not be evident in earlier prototypes.
- Each stage builds upon the last, so making bad design choices early in development can be costly.
- Because progress is made horizontally across the product, some features may lack depth and functionality depending on the progress of the product.

4. Glossary of Terms

This section describes some of the terminology that is used in this document.

Term	Definition
Iteration	The number of the current cycle. For example, if the project hasn't completed the first cycle, then it is in the first iteration.
Stage	The current step in the cycle. For the incremental build model there are 6 stages for each cycle, Planning, Requirements, Analysis & Design, Implementation, Testing, and Evaluation.
Release	The current version of the product that is given to the customer at the end of each cycle for evaluation. In the last iteration, there will be a final release.

5. Appendix A: Rejected Process Models

Traditional Waterfall Model

Description:

The Waterfall model is a sequential design process, often used in software development. In this model, progress is seen as flowing steadily downwards, like a waterfall, through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance.

Pros:

- Very well defined structure.
- Clearly defined deliverables
- Easy to use
- Strongly endorsed by SE department for the senior project
- Good for projects that need a very rigid structure, such as government projects or highly regulated projects.
- Widely used and known
- Reinforces good habits - design before coding
- Easily defined milestones and deliverables.
- Performs well with projects that have clearly defined requirements.

Cons:

- Pure waterfall requires distinct, non-overlapping phases.
- Sometimes doesn't fit in real-world situations where the requirements are volatile.
- Not good for exploratory development
- Sometimes not easy to have all of your requirements defined that early in the project
- Software is delivered late in project cycle, so any delays can be very costly.
- Difficult and expensive to make changes to artifacts made in earlier phases.
- Significant administrative overhead, could be costly for small teams

For more information, visit: http://en.wikipedia.org/wiki/Waterfall_model

---

SCRUM

Description:

Scrum is an agile development process. Scrum models allow projects to progress via a series of iterations called agile sprints. Each sprint is typically two to four weeks, and sprint planning in the agile methodology and Scrum process is essential. While the agile Scrum methodology can

be used for managing any project, the Scrum agile process is ideally suited for projects with rapidly changing or highly emergent requirements like software.

Pros:

- Iterative
- Get software to the customer faster.
- Easier to handle volatile requirements
- Daily meetings make it easier to track individual progress of each team member.
- Issues are identified early in process due to constant contact with each other and the speed of the methodology.
- Easier to deliver quality software on-time
- Good when the project scope and requirements are not clearly defined.

Cons:

- Scrum is one of the leading causes of scope creep
- Tasks are not clearly defined so estimating time required for each one can be difficult
- If team is not fully committed, the project will fail or not get completed.
- Team members need experience with scrum for it to work well.
- Team will always be “sprinting”, to get functionality completed for current sprint.

For more information, visit: http://en.wikipedia.org/wiki/Scrum_(software_development)
Or: http://www.mountaingoatsoftware.com/topics/scrum

Spiral Model

Description:

The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. Requirements are gathered during the planning phase. In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase

Software is produced in the engineering phase, along with testing at the end of the phase. The evaluation phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

Pros:

- High amount of risk analysis
- Good for large and mission critical projects

- Software is produced early in software lifecycle
- It is iterative
- Gets the customer involved early in project, as they are needed for the evaluation phase of each iteration.
- Additional functionality can be added at a later date.
- Strong approval and documentation control

Cons:

- Can be a costly model to use, with all of the risk analysis and throwaway prototyping.
- Risk analysis requires highly specific expertise
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for small projects

For more information, visit: http://en.wikipedia.org/wiki/Spiral_model