Reflective Journal A

Name: Marc Tibbs
Email: tibbsm@oregonstate.edu
Student ID: 933270515
Class: CS361 Software Engineering – Winter 2019
Assignment: Reflective Journal A

Monday, January 7, 2019

I just opened Canvas for the first time this year and checked out the info that has been released for the 2 courses that I'll be taking this term: CS344 OS & CS361 Software Development. I am excited to get back into the swing of things and take both these courses this term as I took the last term off so that I could focus on my work at my internship. I am now at my second internship and while I have been learning a lot on the job, there are many times when I do not know what is being talked about because I haven't taken certain classes. Specifically, there's usually talk about architectures that I have heard of, but know nothing about. It will be nice to finally fill in those gaps in my knowledge and get on the same page with people at my work. I suppose I am a little worried about taking 2 classes and taking the internship, but I am sure that I'll be able to handle the work.

Tuesday, January 8, 2019

Video L0-Overview:

I just watched the first lecture for this class: L0-Overview. It went over the basis for the course. It started out by explaining that software engineering is not just about the design process. In fact, the software development cycle has many steps including, developing requirements, designing solutions, implementation of the project, and testing.

The question, why are projects over budget so often? And, how do you properly estimate the needs of a project before committing to it and starting on it. For the first question, I think that under estimation of a project's budget happens for many different reasons. For example, any project has to pass under the scrutiny of the hierarchy of a business and many project leads may underestimate the project's budget in order to get past the politics inherit in any company. Another reason a project's budget may be underestimated are the unknown issues that crop up in any venture in life. Many projects have known unknowns, which are risk that you are aware of and can be mitigated by planning. However, there are also unknown unknowns which are risks that come from situations that are unexpected and as a result cannot be planned for, at least directly.

Software engineering has to deal with these types of problems of estimation and planning. We are given a definition of software engineering as "the application of a systematic, disciplined, quantifiable, approach to the development, operation, and maintenance of software." The word quantifiable is key here. In order to continually develop better software, we need to be able to measure the software's strengths, weaknesses, threats, and opportunities. Quantifiable elements of a software project include the technology's complexity, organizational skills available, available funding and workforce, the project schedule, the stakeholders interest and assistance, laws and regulations, etc.

Wednesday, January 9, 2019

Video L1-Overview

I just watched the second lecture video for this week, L1-Overview. This lecture starts out with emphasizing that software is not just about computers. We use computers as tools to solve problems. For example, a microloan website to help people start businesses.

Another example of a problem is natural disasters. Software can help in this case by locating victims, aid in the distribution of resources, and helping emergency supporters help victims.

Some other problems software is able to alleviate are pollution, repression, disease/medicine, etc.

Software engineering is basically problem solving, without making the world worse and without incurring excessive costs. Software engineering solves these problems by creating software.

Thinking about software is not enough. You also need to think about the people that will be using the software and the context that it will be used in.

A system boundary is the software being developed. People intereact with the system boundary. Is defined by the engineer/designer.

What is the difference between good software and great software?

- Reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability, interoperability.

Great software contains the right features for the right data. You cannot build a great system until you understand what it should do.

- Use cases (the activities a system supports), entities (the kinds of objects that are involved in use cases), attributes (properties of the entities).

Software engineering is a team effort and requires a lot of different skillsets. You will need analysts, designers, programmers, testers, and trainers.

Course objectives: process, requirements, documents, notations, design, validate, cost & schedule and team.

L2-Software engineering process

Quality Attributes of great software: Reliability: Consistency, Accuracy, Error to tolerate, Efficiency: Execution efficiency, storage efficiency Integrity: Access control, Access audit Usability: Operability, training, and communicativeness...

A process is a well-defined and usually involves a set of tools and techniques

- Uses resources, is subject to constraints and produces intermediate and final products
- may be composed of sub-processes
- has entry and exit criteria

Life-cycle: the process of building and maintaining the product.

Typical software task:

- figuring out what the system should do - requirements
- figuring out how the system should do it - design
- writing the code - implementation
- using the using - operation

Types of processes: Waterfall - Spiral - Agile -

Thursday, January 10, 2019

Key Questions 1.1 Why are the elements of the SDLC so important for large software systems? Look up Microsoft Secure Development Lifecycle for clues.

The SDLC is important to any software system, but is even more important for large systems because it sets out a process for the to be designed, implemented and designed in a way that they are continually and iteratively able to be improved upon. By their very nature large systems are complex. Their sheer size often makes them hard to understand after a quick glance. Th SDLC allows developers to create a system in a set convention that allows other developers to later come in to the project and more easily understand the nature of the project knowing that it originally followed SDLC principles.

Friday, January 10, 2019

1.2 SDLC sounds like a lot of writing. Why is that important?

SDLC involves a lot of writing for a variety of different reasons. One reason for it is to allow the designers, developers, and architects a well-defined view of their design and implementation processes.This type of record is important not only to the specific project that a team is working on together, but also helpful for their development system for all project that they will ever work on. With a written record of their thought processes, they can go back to see why they decided to design software in a certain way. Or, perhaps they missed something in their first design. This brings up another reason of why SDLC involves a lot of writing, which is that the writing allows teams to build upon in a iterative way. With a record of their past work, teams are able to patch up any design holes that they missed the first time around. At the same time, teams are able to see what options they have already tried and if they tried a certain design in the past that didn't work, they have a record that will discourage them fro making the same mistakes again. Finally, perhaps the most important reason for the amount of writing in SDLC is that teams need to communicate with each other effectively in order to work together effectively. Good documentation of the SDLC give a team a point of reference which team members can refer to to get insync with one another. Without this point of reference it is easy for team members to get off track with one another and make mistakes.

Sunday, January 12, 2019

I just submitted the first draft for my vision statement and am a little unsure about the topic I picked. I chose to create my project around creating a platform for students to talk to one another and contribute to each other's open source projects. I'm not too sure about it though. It doesn't seem like there will be enough to build up a term-long project around. I picked the topic because it seemed interesting. But seeing that the final word count for the final vision statement need to be around 20,000 words, I'm not sure I can write that much. Doing research on the topic didn't show too many articles or information for me to research. So I hope I am able to find more information about it. I'll give it a fresh look tomorrow and see what I can do.

Tuesday, January 14, 2019

1.3 SDLC sounds more like project management than software development. Agree or disagree? Why?

- I agree with this statement. Project management principles are part of software development process. In fact, project management principles are needed in almost any type of developmental environment. Project management principles ensure that a developmental plan is well-structured and designed to succeed. It would be a mistake to think that project management and software development can be separated. You many be able to get away without any PM principles in small, short-term projects, but for anything larger and with a longer duration, you are better off sticking to Project Management principles.

Wednesday, January 15, 2019

1.4 Computers have become ubiquitous in our lives and societies. Many tragedies have occurred because of computer malfunctions. How could software engineering have helped? Or not? HINT: look up the comp.risks archives for 30 years of malfunctions.

- In part I think that, as with any technology, the improvements that come with computer necessarily come a number of downfalls with them. With that said there is always room for improvements no matter how small they are. Of course, better adherence to software engineering standards could have mitigated some of these problems and I think that is where we should focus our attention to. Of course, nowadays more people are placing a larger emphasis on the security aspects of software and that is something that was largely overlooked in the past, as it simply did not have an impact back in the day. It usually takes something bad to happen for there to be changes. So, we have seen many security breaches and other hacks, or other catastrophes caused by the failures of our machines, but that has woken many people up and made them want to patch these glaring holes that were overlooked in the past. I think we are all headed in the right direction, but we are necessarily going to face many more malfunctions before we can fix anything.

Thursday, January 16, 2019

Lecture 3.1 - Requirements Overview

Today, I watched lecture 3.1 - requirements overview. The lecture went over the importance of well-defined and comprehensive requirements in building large projects. Simply put, A requirement is something that states the purpose of the system or, in other words, expresses the desired behavior. Requirements are very helpful for communication with customers and co-workers. they also help with keeping track of everything that needs to get done in a project. Another advantage is that they help clients and customers decide the key facets of a project and set priorities for what needs to get done first. Finally, customers often do not know what they really need and sitting down with them to go over the requirements is often the best way to clarify things.

Friday, January 17, 2019

Lecture 3.2 - Notations

Today, I watched lecture 3.1 - Notations. The lecture went over the notations for a variety of different diagrams that are useful in sketching out the requirements of a software project. Diagrams are often helpful to these projects as a way to communicate, visualize, or analyze the details of a project. They additionally give the project a sort of structure to begin with.

Some important diagrams are class diagrams, entity-relationship diagrams, data flow diagrams, sequence diagrams, and state diagrams. All these diagrams have slightly different notation, but they thankfully aren't too different from one another. This makes it easier to look at all these different diagrams, and even without looking up the details about the diagramming information, anyone can get a good idea of what is going on in the diagram.

Saturday, January 18, 2019

Lecture 3.3 - Example Requirements

I really enjoyed watching today's lecture as it really solidified some of the concepts that I've been hearing about the past couple of weeks. It's always nice to see actual examples of what is talked about in the lectures and required for our homework. After, seeing this lecture I feel ready to tackle HW1.

Sunday, January 19, 2019

Today, I worked on HW1. I wrote out one of the use cases for our application. More specifically, I wrote a use case for 'Creating a New Trip' using our mobile/web app for planning trips. I haven't seen any other use cases other than the one that was in lecture 3.3, but I think I made a pretty darn good use case. We'll see, I guess...

Monday, January 20, 2019

Lecture 3.4 - Evaluating Requirements

Today, I watched lecture 3.4, which went over the topic of evaluating requirements. There are two basic components of evaluating requirements: validation & verification. Validation ensures that the right system was built and verification ensures that the system was built correctly. There are many tiny, iterative steps to the evaluation process. Some of the more popular and tested techniques include paper prototyping, low-fidelity prototyping, high-fidelity prototyping, stakeholder review, formal analysis, and manual analysis. While it might not be feasible to complete all these different types of evaluation, the more you are able to do will likely lead to a more refined outcome.

Friday, January 26, 2019

I've missed more than a couple of journal entries over the past week because I've been interviewing for a few full-time positions. I feel like my interviews went pretty well, so hopefully I get one of these jobs. I am a little bummed that I missed making contributions over the last few days, but I really happy I had a good start on them at the beginning of the term and a little cushion to start with. I'm going to have to turn in this journal in a few days, so I'll try to make the last couple of entries good.

Sunday, January 27, 2019

So, I have to turn this journal in tomorrow. I didn't do much towards this class today because I'm currently traveling around Japan. I have the next lectures downloaded so I plan on watching those over the next few days, which will give me more to talk about in Reflective Journal B.

2.1 What options do you see for coordinating activities among hundreds of people?

- I think this is a pain point for most large companies. The solutions being used are usually Project Management software, such as Trello, Asana, Jira, and the like. I think these kinds of software platforms help a lot in coordinating many tasks across large teams and are pretty intuitive to use as well.

2.2 How do you record and distribute decisions made across the organization?

- I think that most communication about decisions can be made across organizations using chat tools such as Slack or Discord. It takes a little bit of extra work from the teams to organize all the information on those platforms, but once it is set up, it is really nice to have all the information you need on the chat service you are using at your company.