Jasir Massey-Campbell

Software Engineering

Assignment 2

**2.1.** In the introduction to this chapter Baetjer notes: “The process provides interaction between users and designers, between users and evolving tools, and between designers and evolving tools [technology].” List five questions that (a) designers should ask users, (b) users should ask designers, (c) users should ask themselves about the software product that is to be built, (d) designers should ask themselves about the software product that is to be built and the process that will be used to build it.

Communications between these three mediums provide the foundation for what is the evolution of a technology. As with many industries, there is a revolving cycle of communication between parties that make up that particular niche. For software there are 3 parties primarily involved. These would be the consumer, the designer, and the technology. Designers of the product choose to design their products based off a need amongst the consumers/users. This communication can occur through a various amount of ways including users telling designers about a product they would support, or designers surveying users about what they think should be created or improved.

5 questions designers should ask users

1. What feature can be added to our software easier to use (user interface)?
2. What about our software can be improved to better assist you overall (features of the service the software provides)?
3. Would you recommend this software to another party?
4. What type of software collaborations would you be interested in seeing?
5. What features do you think is outdated (what should be removed)?

5 questions users should ask designers

1. Can [this] feature be added?
2. What are the challenges to adding this feature?
3. What is the importance in reporting software errors?
4. What makes this model better than the competitor’s or the previous model?
5. What is the shelf life? How long until the next model releases?

5 questions users should ask themselves

1. How often will I use this software?
2. Does this software solve my issue?
3. How effective is this software at solving my issue?
4. How cost effective is this software? (does it makes sense to downgrade or upgrade?)
5. Do I envision myself buying this again?

5 questions designers should ask themselves

1. Will creating this software make sense financially? (is it sustainable?)
2. How quickly can we develop the software?
3. Are there legal conflicts at all with our software? (i.e. can it be deployed worldwide, nation wide or in particular states?)
4. How do we get are software out there and recognized?
5. What issue is this software solving?

**2.2.** Try to develop a set of actions for the communication activity. Select one action and define a task set for it.

Simple example: The task set for baking a cake.

Task: Baking cake

1. Find recipe
2. Buy ingredients for cake.
3. Pre-heat oven
4. Combine ingredients
5. Place in oven
6. Set timer
7. Remove from oven
8. Let cool
9. Garnish with icing or other decorations

**2.3.** A common problem during **communication** occurs when you encounter two stakeholders who have conflicting ideas about what the software should be. That is, you have mutually conflicting requirements*.* Develop a process pattern (this would be a stage pattern) using the template presented in Section 2.1.3 that addresses this problem and suggest an effective approach to it.

Pattern Name: Conflict with software direction amongst stakeholders.

Forces: stakeholders have conflicting ideas about what the software should be

Types: Stage Pattern

Initial Context: Must sort out conflicting requirements given by stakeholders

Problem: Disagreeing stakeholders on direction of software

Solutions:

1. develop two separate software that meets each of their individual requirements.
2. try to find a “middle-ground” that best attacks both requirements of the stakeholders.

Resulting Context: Can not focus on developing intended software for stakeholders

Related Patterns: n/a

**2.7.** Provide three examples of software projects that would be amenable to the waterfall model. Be specific.

The waterfall model is a linear approach to software development. This model is best for simple projects (in terms of the development, not the software itself). This could mean an enhancement to an existing project or the creation of a new one all together. This model is best when the majority of the details are already given and planned out. This method is not good for software development that will require revisions and repeated testing.

1. Software created for government agencies

2. Software focused around a database/user account system

3. Basic Information providing software (i.e weather app)

**2.8.** Provide three examples of software projects that would be amenable to the prototyping model. Be specific.

Prototyping is a method with a large focus on user feedback. This method is best when communicating with a customer that may not have the technical verbiage to articulate exactly what they what from a programming perspective.

1. Website development

2. Complex app development with focus on user interaction (i.e. social media app)

3. Video game development

**2.10.** Provide three examples of software projects that would be amenable to the incremental model. Be specific.

This model is similar to the linear model with a difference of having the capability to be released sooner than the linear model can. This model is best for linear efforts than can be accomplished in parts. File management systems are best define using an increment model. Software that has a core functionality but that intends to add additional features to it should follow this type of model.

1. Streaming services

2. Postmates/UberEats

3. Operating Systems

**2.11.** As you move outward along the spiral process flow, what can you say about the software that is being developed or maintained?

The spiral method is a risk driven process model. With this model, software is released as a series. The risk associated declines with each iteration. In other words, with the spiral model, software starts out as a prototype and through each iteration, a more complete version of the software is created.

**2.14.** What are the advantages and disadvantages of developing software in which quality is “good enough”? That is, what happens when we emphasize development speed over product quality?

Advantages:

* more likely to meet deadlines
* cleaner code
* receive feedback faster
* deploy software quicker

Disadvantages:

* sloppier code/more likely to have mistakes
* longer time till feedback is received
* longer deployment time
* lesser quality software

**3.2.** Describe agility (for software projects) in your own words.

Agility is what many software development teams seek to portray themselves as. It represents the desired mindset of a developer or development team that can appropriately and quickly respond to arising software development challenges.

**3.3.** Why does an iterative process make it easier to manage change? Is every agile process dis- cussed in this chapter iterative?

All agile process are iterative processes. It is easier to manage change because agile systems accommodate for change. One of the ways they do this is by addressing the problems in previous iterations in the next iteration. It is much easier to do this when a project is broken down into parts versus reviewing the entire project as a whole.

Is it possible to complete a project in just one iteration and still be agile? Explain your answers.

I think that it is as long as the agile method was followed correctly during that one iteration and maintains the characteristics of being agile if further increments needed to be acted upon.

**3.6.** Select one agility principle noted in Section 3.3.1 and try to determine whether each of the process models presented in this chapter exhibits the principle. [Note: I have presented an overview of these process models only, so it may not be possible to determine whether a prin- ciple has been addressed by one or more of the models, unless you do additional research (which is not required for this problem).]

The first principle of the Agility Principle is “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.” I would argue that this is the goal of all methods even the waterfall model. Although the waterfall model is best for straightforward models, all of the models have been created so that the most efficient process is acted upon in order to accomplish the development goal. In regards to the continuous delivery of the software, the method would need to be an iterative development method.

**3.7.** Why do requirements change so much? After all, don’t people know what they want?

The answer to the second question is absolutely not. Clients who hire development teams to work on projects may understand what they want from the perspective of their industry and their imagination but they may not be able to articulate that image in “tech speech” for a programmer to understand. Due to this, it is not uncommon for an initial intention in the development process becoming a revised edit. This could be something as simple as picture placement or color change. This is why certain methods were developed. Not all models accommodate for multiple changes.