CMPE 285 – Software Engineering Processes

Homework 1

First Name: Harish

Last Name: Marepalli

SJSU ID: 016707314

Book: **Software Engineering: A Practitioner's Approach (Eighth Edition)**

Problem 4.3: What process adaptations are required if the prototype will evolve into a delivery system or product?

Answer:

The concept of prototyping is beneficial, for both developers and stakeholders as it helps them gain an understanding of what to build particularly when the requirements are unclear.

One can easily understand it by looking at the figure below that is already provided.

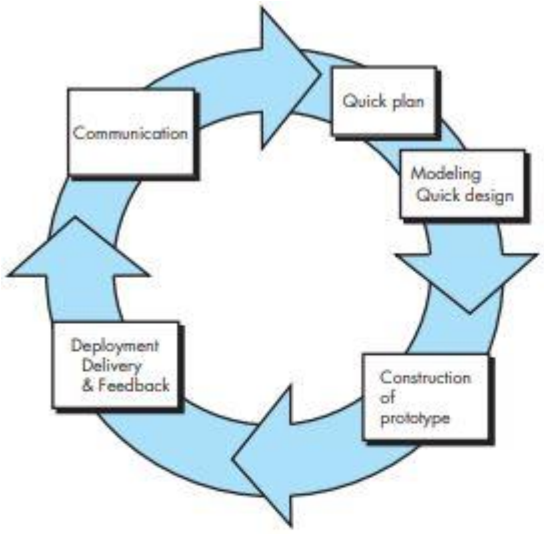

Fig: The prototyping paradigm

The process adaptations that are required if the prototype will evolve into a delivery system or product are:

1. It is crucial to implement design rules from the beginning.

2. Software Quality Assessments procedures should be applied from the start.

3. The prototype should be designed with scalability in mind serving as a foundation, for extensions that will transform it into a fledged production system.

Problem 4.6: Is it possible to combine process models? If so, provide an example.

Answer:

Yes, it is possible to combine the software process models.

Some possibilities to combine of software process models are:

1. *Evolutionary Process Model*: This model involves gradually building more complete versions of the software, with each version going through a full cycle of activities.

2. *Incremental Process Model*: In this approach, a series of releases are created, each adding more functionality to meet customer needs. Each increment is individually designed, tested, and delivered over time.

3. *Spiral Model*: Combining elements of both the prototyping and waterfall models, the spiral model allows for rapid development of increasingly comprehensive software versions.

Let's consider a real-world example, such as the Department of Defense:

i. If the project follows a strict waterfall or incremental model, that's one approach.

ii. However, if the software needs to integrate with existing features, a component-based model might be more suitable.

iii. When time is of the essence, a rapid application model can be applied.

In essence, you can combine and match evolutionary, incremental, and spiral process models to create a tailored software development approach. This flexibility is also applicable in medical and research projects.

Problem 5.10: What is a spike solution in XP?

Answer:

1. Extreme Programming (XP) is a popular agile software development method used for implementing software projects.

2. XP predominantly follows an object-oriented development approach.

3. XP consists of a set of rules and practices conducted within the framework activities of the project.

Spike Solution:

1. A spike solution is a small technique employed to explore potential solutions to challenging problems.

2. It's used to address tough problems or design challenges by creating a temporary solution.

3. When faced with a difficult problem without an obvious answer, a spike solution is created.

4. The primary objective of a spike solution is to maintain simplicity and minimize risks.

5. This involves experimenting with various coding approaches to make the correct solution more evident.

6. This is a common practice in independent development and may not always require the formality that agile programming methodologies demand.

Spike Solution in XP:

1. In XP, a spike solution is essentially a disposable prototype.

2. It's implemented and evaluated to reduce risks when moving forward with the actual implementation.

3. The goal is to validate the initial estimates for the user stories that contain the design pattern.

4. Spike solutions are created to address difficult problems or design challenges and reduce potential risks.

5. The spike approach takes a comprehensive "end-to-end" approach to problem-solving.

6. In XP, spike solutions are implemented to mitigate risks and prepare high-quality solutions for challenging problems, as required by agile programming practices.

Problem 5.11: Describe the XP concepts of refactoring and pair programming in your own words.

Answer:

Refactoring:

1. Refactoring is a constructive technique, and it's actively promoted in Extreme Programming (XP).

2. This process involves making changes to a software system in a way that doesn't affect how the code behaves externally but enhances its internal structure.

3. It's a methodical approach to cleaning up code, reducing the risk of introducing errors.

4. The main goal of refactoring is to guide modifications through small design improvements that can significantly enhance the overall design.

5. Design considerations are relevant both before and during the coding process. In essence, refactoring ensures that design evolves continuously as the system is being built.

Pair Programming:

1. Pair programming suggests that two individuals or teams collaborate at a single computer workstation to develop code for a specific task.

2. This approach facilitates real-time problem-solving and immediate quality assurance.

3. In practice, each participant typically assumes slightly different roles. For example, one may focus on coding the specifics of a particular design section, while the other ensures adherence to coding standards and that the resulting code aligns with the project's requirements.