Assignment for Lecture 2: Software Processes

LAI Hui Shan
m5281022

1. Describe the problems with incremental development.

Ans:

1. The process is not visible.
    - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
2. System structure tends to degrade as new increments are added.
    - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.
3. The problems of incremental development become particularly acute for large, complex, long-lifetime systems, where different teams develop different parts of the system.
4. Large systems need a stable framework or architecture and the responsibilities of the different teams working on parts of the system need to be clearly defined with respect to that architecture. This has to be planned in advance rather than developed incrementally.
    - Water-fall is still a possible option

2. In modern software development, should there be a clear distinction between the development process and the maintenance process? Please describe your answer and discuss why.

Ans:

In modern software development, the distinction between development and maintenance is increasingly blurred. Traditional models, like the waterfall model, treat development and maintenance as separate, but with the rise of incremental development and agile processes, they are often integrated. Software evolves continually in response to changing requirements, so maintenance is not just a post-development activity. Iterative approaches such as prototyping and incremental delivery help accommodate changes more easily, allowing for continuous feedback and adaptation.

3. Explain why incremental development is the most effective approach for developing business software systems.

Ans:

1. The cost of accommodating changing customer requirements is reduced.
    - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.

2. It is easier to get customer feedback on the development work that has been done.
    - Customers can comment on demonstrations of the software and see how much has been implemented.
3. More rapid delivery and deployment of useful software to the customer is possible.
    - Customers are able to use and gain value from the software earlier.

4. Explain why change is inevitable in (complex) software systems.

Ans:

1. **Business** changes lead to new and changed system requirements
2. New **technologies** open up new possibilities for improving implementations
3. Changing **platforms** require application changes

5. Give examples of software process activities that can help predict possible changes and make the software being developed more resilient to change.

Ans:

1. **Change avoidance**, where the software process includes activities that can anticipate possible changes before significant rework is required.
    - For example, a **prototype system** may be developed to show some key features of the system to customers.
2. **Change tolerance**, where the process is designed so that changes can be accommodated at relatively low cost.
    - This normally involves some form **of incremental development**. Proposed changes may be implemented in increments that have not yet been developed.

6. Explain why systems developed as prototypes should not normally be used as production systems.

Ans:

1. It may be impossible to tune the system to meet non-functional requirements;
2. Prototypes are normally undocumented;
3. The prototype structure is usually degraded through rapid change;
4. The prototype probably will not meet normal organizational quality standards.

7. Explain why Boehm’s spiral model is an adaptable model that can support both change avoidance and change tolerance activities.

Ans:

1. The main difference between the spiral model and other software process models is its explicit recognition of risk.
2. A cycle of the spiral begins by elaborating objectives such as performance and functionality.
3. Alternative ways of achieving these objectives, and dealing with the constraints on each of them, are then enumerated.
4. Each alternative is assessed against each objective and sources of project risk are identified.

5. The next step is to resolve these risks by information-gathering activities such as more detailed analysis, prototyping, and simulation.
6. Once risks have been assessed, some development is carried out, followed by a planning activity for the next phase of the process.
7. Informally, risk simply means something that can go wrong.
8. Risks lead to proposed software changes and project problems such as schedule and cost overrun, so risk minimization is a very important project management activity.
9. The spiral model combines change avoidance with change tolerance.
    - It assumes that changes are a result of project risks and includes explicit risk management activities to reduce these risks.

8. Giving reasons for your answer based on the type of system being developed, suggest the most appropriate generic software process model that might be used as a basis for managing the development of the following systems:

- **An e-learning system where adaptive user interfaces are involved**
- **A university accounting system that replaces an existing system**
- **A monitoring system based on image recognition**

Ans:

1. **An e-learning system where adaptive user interfaces are involved:**

    - The **Incremental Development Model** would be the most suitable. Since adaptive interfaces require continuous adjustments based on user feedback, incremental development allows for delivering features in increments and adapting based on user experience. This enables the team to gather feedback before the entire system is fully developed.

2. **A university accounting system that replaces an existing system:**

    - The **Waterfall Model** is a better fit for this type of system, particularly because accounting systems have clear, well-defined requirements and involve strict processes. Since the system replaces an existing one, the requirements are often stable, and the phased design and development approach of the Waterfall Model can effectively manage this process.

3. **A monitoring system based on image recognition:**

    - The **Spiral Model** is the most appropriate due to the high technical risks involved in image recognition. The Spiral Model allows for risk assessment and reduction in each cycle, especially for technical challenges, ensuring that both functionality and performance are achieved as the system evolves.

9. Please summarize the Rational Unified Process (RUP), including basic concepts, advantages, and the evolution of its theory.

Ans:

Basic Concepts:

RUP is a modern generic software process model that integrates aspects of the **Waterfall model**, **Incremental development**, and **Reuse-oriented software engineering**. It is typically described from three perspectives:

- **Dynamic**: Shows the phases over time.
- **Static**: Focuses on the process activities (e.g., requirements, analysis, design, implementation).
- **Practical**: Suggests best practices for software development.

Four Phases of RUP:

1. **Inception**: Establishing the business case for the system.
2. **Elaboration**: Developing an understanding of the problem domain and system architecture.
3. **Construction**: System design, programming, and testing.
4. **Transition**: Deploying the system into its operational environment.

Advantages:

- **Iterative Development**: Plans increments based on customer priorities and delivers the highest-priority features first.
- **Requirements Management**: Explicitly documents customer requirements and tracks changes to them.
- **Component-based Architecture**: Organizes the system into reusable components.
- **Quality Assurance**: Ensures that the software meets organizational quality standards through rigorous verification.

Evolution of Theory:

RUP has evolved into a more flexible model that combines the strengths of waterfall, incremental development, and reuse-oriented approaches. It incorporates activities for managing change and supports cross-phase iterative development and delivery, allowing the system to adapt to changes without disrupting the overall process