Software Development Processes

LiZhengYing 13211206
ZheJiang Normal University
Software Project Management

Abstract

In software engineering, a software development process (also known as a system development methodology, software development life cycle, software development methodology ,software process) is a splitting of software development work into distinct phases (or stages) containing activities with the intent of better planning and management. It is often considered a subset of the systems development life cycle. The methodology may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application.

Keywords: software, development methodology, system, activities, life cycle

1 Introduction

A software development process or life cycle is a structure imposed on the development of a software product. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process.

2 Processes

More and more software development organizations implement process methodologies.

The Capability Maturity Model (CMM) is one of the leading models. Independent assessments can be used to grade organizations on how well they create software according to how they define and execute their processes.

There are dozens of others, with other popular ones being ISO 9000, ISO 15504, and Six Sigma.

3 Process Activities/Steps

Software Engineering processes are composed of many activities, notably the following:

- **Requirements Analysis**

Extracting the requirements of a desired software product is the first task in creating it. While customers probably believe they know what the software is to do, it may require skill and experience in software engineering to recognize incomplete, ambiguous or contradictory requirements.

- **Specification**

Specification is the task of precisely describing the software to be written, in a mathematically rigorous way. In practice, most successful specifications are written to understand and fine-tune applications that were already well-developed, although safety-critical software systems are often carefully specified prior to application development. Specifications are most important for external interfaces that must remain stable.

- **Software architecture**

The architecture of a software system refers to an abstract representation of that system. Architecture is concerned with making sure the software system will meet the requirements of the product, as well as ensuring that future requirements can be addressed.

- **Implementation**

Reducing a design to code may be the most obvious part of the software engineering job, but it is not necessarily the largest portion.

- **Testing**

Testing of parts of software, especially where code by two different engineers must work together, falls to the software engineer.

- **Documentation**

An important task is documenting the internal design of software for the purpose of future maintenance and enhancement.

- **Training and Support**

A large percentage of software projects fail because the developers fail to realize that it doesn't matter how much time and planning a development team puts into creating software if nobody in an organization ends up using it. People are occasionally resistant to change and avoid venturing into an unfamiliar area, so as a part of the deployment phase, its very important to have training classes for the most enthusiastic software users (build excitement and confidence), shifting the training towards the neutral users intermixed with the avid supporters, and finally incorporate the rest of the organization into adopting the new software. Users will have lots of questions and software problems which leads to the next phase of software.

- **Maintenance**

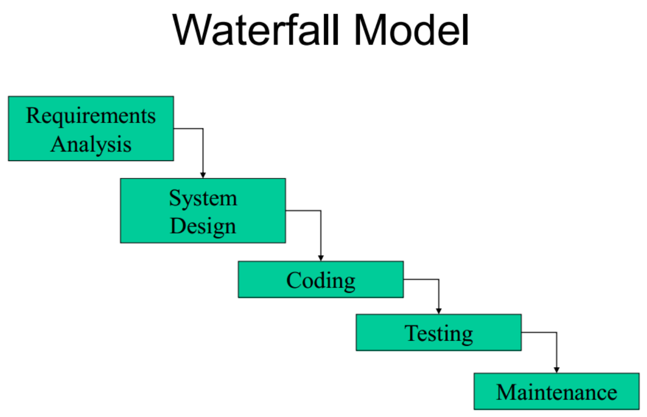
Maintaining and enhancing software to cope with newly discovered problems or new requirements can take far more time than the initial development of the software. Not only may it be necessary to add code that does not fit the original design but just determining how software works at some point after it is completed may require significant effort by a software engineer. About 60% of all software engineering work is maintenance, but this statistic can be misleading. A small part of that is fixing bugs. Most maintenance is extending systems to do new things, which in many ways can be considered new work.

4 Process Models

A decades-long goal has been to find repeatable, predictable processes or methodologies that improve productivity and quality. Some try to systematize or formalize the seemingly unruly task of writing software. Others apply project management techniques to writing software. Without project management, software projects can easily be delivered late or over budget. With large numbers of software projects not meeting their expectations in terms of functionality, cost, or delivery schedule, effective project management is proving difficult.

Waterfall Model

The best-known and oldest process is the waterfall model, where developers follow these steps in order. They state requirements, analyze them, design a solution approach, architect a software framework for that solution, develop code, test, deploy, and maintain. After each step is finished, the process proceeds to the next step.



V Model

The V Lifecycle is an Application Development methodology that supports new product development or large enhancement projects when the size and complexity of the system is large enough to have a multi-layered design.

The V Lifecycle is suitable when:

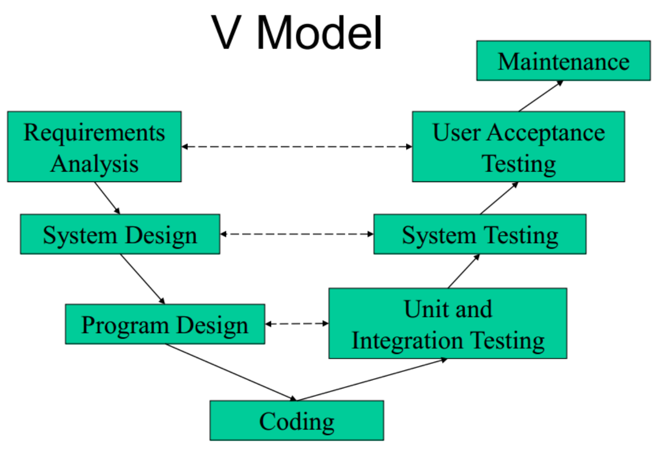
The system is large and complex and requires a multi-layered design.

The system can be broken down into sub-systems, modules, and units.

The system requirements are clear and complete.

The system requires high maintenance and support.

The system requires high reliability and accuracy.



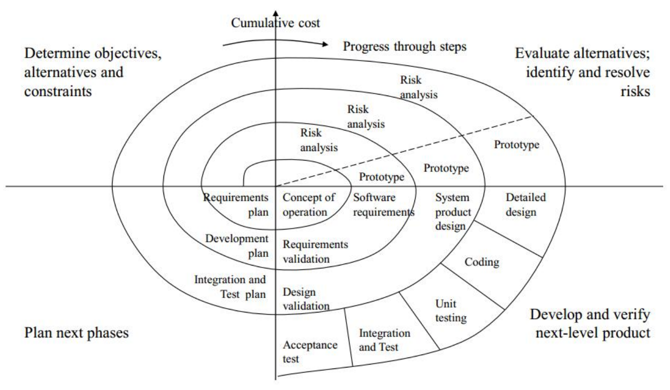
Spiral Model

In 1988, Barry Boehm published a formal software system development "spiral model," which combines some key aspect of the waterfall model and rapid prototyping methodologies, in an effort to combine advantages of top-down and bottom-up concepts. It provided emphasis in a key area many felt had been neglected by other methodologies: deliberate iterative risk analysis, particularly suited to large-scale complex systems.

The basic principles are:

- Focus is on risk assessment and on minimizing project risk by breaking a project into smaller segments and providing more ease-of-change during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle.
- "Each cycle involves a progression through the same sequence of steps, for each part of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program."
- Each trip around the spiral traverses four basic quadrants: (1) determine objectives, alternatives, and constraints of the iteration; (2) evaluate alternatives; Identify and resolve risks; (3) develop and verify deliverables from the iteration; and (4) plan the next iteration.
- Begin each cycle with an identification of stakeholders and their "win conditions", and end each cycle with review and commitment.

Spiral Model (adapted from Boehm 1987)



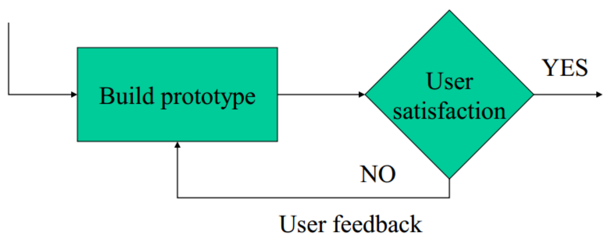
Prototyping Model

Software prototyping, is the development approach of activities during software development, the creation of prototypes, i.e., incomplete versions of the software program being developed.

The basic principles are:

- Not a standalone, complete development methodology, but rather an approach to handle selected parts of a larger, more traditional development methodology (i.e. incremental, spiral, or rapid application development (RAD)).
- Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
- User is involved throughout the development process, which increases the likelihood of user acceptance of the final implementation.
- Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users' requirements.
- While most prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system.
- A basic understanding of the fundamental business problem is necessary to avoid solving the wrong problems.

Prototyping Model



5 Conclusion

In this class, I had known Software Engineering processes and many process models. Waterfall Model, V Model. Spiral Model, Prototyping Model. The sequence of events for the waterfall model is analysis,design,coding,testing,maintenance. The most important feature of spiral model is risk management. The V-model is considered to be an extension of the Waterfall model. Instead of a downward linear path the process moves downward until it reaches the coding stage whereupon it begins moving upward until it passes user acceptance. Significant difference between V- and Waterfall models is that the former includes well-defined Verification and Validation phases.
The prototyping model of software development is a useful approach when a customer cannot define requirements clearly.

6 References

https://en.wikipedia.org/wiki/Software_development_process#cite_note-CMS08-1

http://www.selectbs.com/analysis-and-design/what-is-a-software-development-process

http://www.acquisitionofknowledge.org/moodle/pluginfile.php/864/mod_resource/content/3/020%20--%20Models.pdf