Software Development Processes

Pengxiang (Student No: 13211223)*
Zhejiang Normal University
Software Project Management

Abstract

A software development process (also known as a software process, system development methodology, software development life cycle) is a structure imposed on the development of a software product. 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. Software Process Model is guideline and overall framework for software developments. Common methodologies include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, extreme programming and various types of agile methodology.

Keywords: Software development process, software process model, software life cycle.

1 Introduction

Software development process is a general process of software design ideas and methods, including the design function and implement algorithms of software, overall structure of modular design, programming and commissioning, testing, coding and submitting program. We address the following concepts in this report:

✔ What is software development process?
✔ How to work of software process model?
✔ What are some of the issues faced when using this model?
✔ What activities having in the software development processes?

A structured set of activities required to develop a software system：
- Specification
- Design
- Validation
- Evolution

Processes More and more software development organizations implement process methodologies.

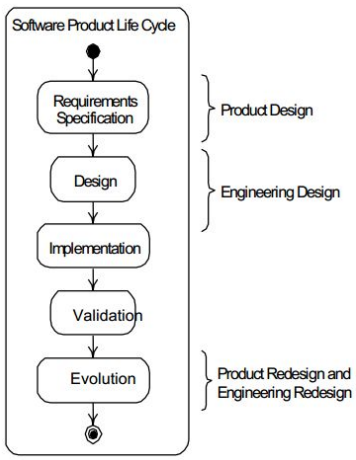

Figure 1: Software Product Life Cycle.

Process Activities/Steps Software development processes are composed of many activities, notably the following:

- Requirements Analysis
- Specification
- Software architecture
- Implementation
- Testing
- Documentation
- Training and Support
- Maintenance

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.

2 Software Process Model

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.
The goals of a process model are to be:

Descriptive

- Track what actually happens during a process
- Take the point of view of an external observer who looks at the way a process has been performed and determines the improvements that must be made to make it perform more effectively or efficiently.

Prescriptive

- Define the desired processes and how they should/could/might be performed.
- Establish rules, guidelines, and behavior patterns which, if followed, would lead to the desired process performance. They can range from strict enforcement to flexible guidance.

Explanatory

- Provide explanations about the rationale of processes.
- Explore and evaluate the several possible courses of action based on rational arguments.
- Establish an explicit link between processes and the requirements that the model needs to fulfill.

Pre-defines points at which data can be extracted for reporting purposes.

3 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. (Progress flows from the top to the bottom, like a cascading waterfall.)

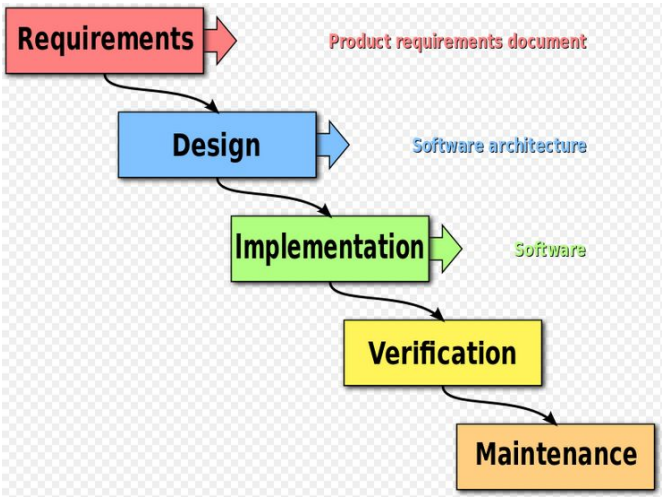

Figure 1: The unmodified "waterfall model".

4 V-Model

In software development, the V-model[2] represents a development process that may be considered an extension of the waterfall model, and is an example of the more general V-model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represents time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.

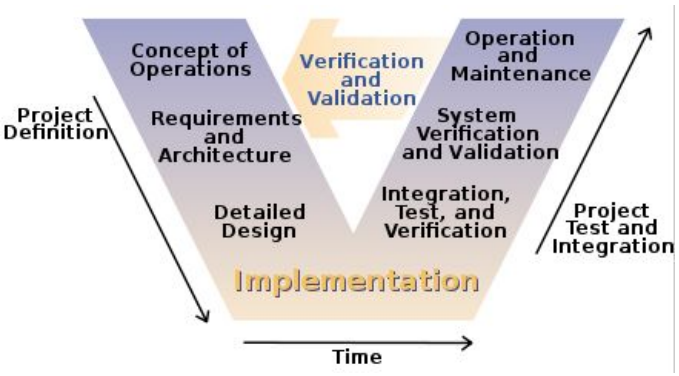

Figure 1: The V-model of the Systems Engineering Process.

5 Iterative processes

Iterative development prescribes the construction of initially small but ever larger portions of a software project to help all those involved to uncover important issues early before problems or faulty assumptions can lead to disaster. Iterative processes are preferred by commercial developers because it allows a potential of reaching the design goals of a customer who does not know how to define what he wants.

6 Software prototyping

Software prototyping is the activity of creating prototypes of software applications, i.e., incomplete versions of the software program being developed. It is an activity that can occur in software development and is comparable to prototyping as known from other fields, such as mechanical engineering or manufacturing.A prototype typically simulates only a few aspects of, and may be completely different from, the final product.

7 Dual Vee Model

When developing complicated systems, a system engineer must manage a system baseline configuration from start to finish. The baseline can include design documents, user manuals, the product itself, and should answer every What?, Why?, and Who? for a system's architecture. At each development phase, there will be changes to the system, which will change the baseline.

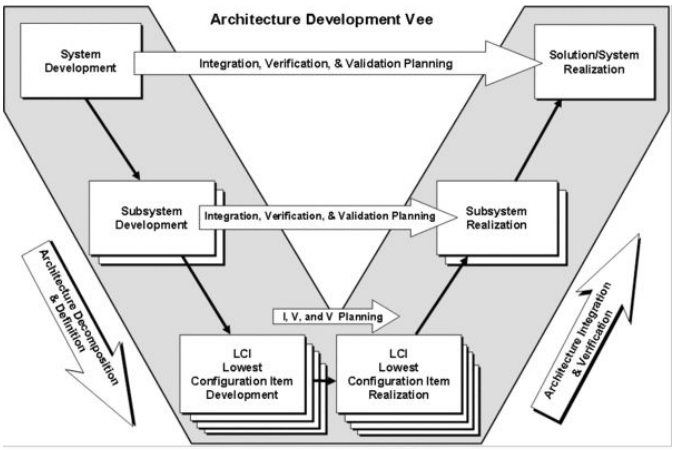

Figure 1: Architecture development vee model (provides what, why, and who).

8 Incremental build model

The incremental build model is a method of software development where the product is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping. The product is decomposed into a number of components, each of which is designed and built separately (termed as builds). Each component is delivered to the client when it is complete. This allows partial utilization of the product and avoids a long development time. It also avoids a large initial capital outlay and subsequent long waiting period. This model of development also helps ease the traumatic effect of introducing a completely new system all at once. There are, however, several problems with this model.

9 Spiral model

The spiral model is a risk-driven process model generator for software projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.

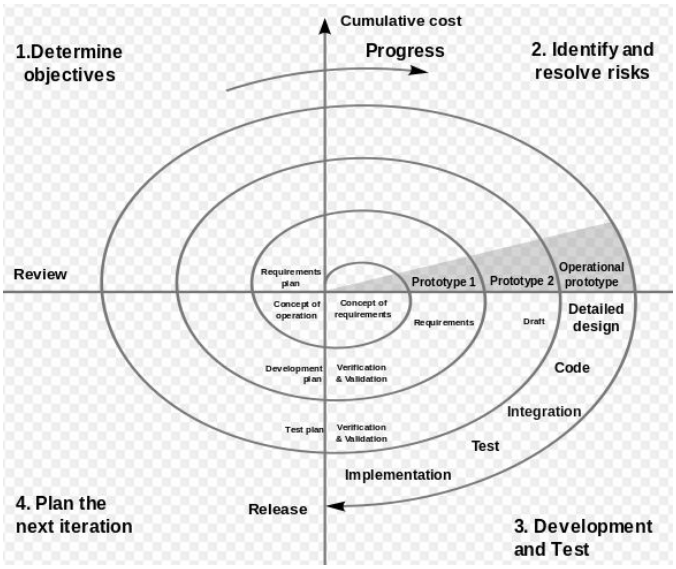

Figure 1: Spiral model of the Systems Engineering Process(Boehm, 2000).

References

http://baike.baidu.com/link?url=lgETzKA-mcQFbgFmYR_EeMo00NtusYf7LP2X5WVf4Dgtdew96YvgIG33lNuR5w20697E5kX7zHRmPH5ageugbq

Boehm, B, "Spiral Development: Experience, Principles,and Refinements", Special Report CMU/SEI-2000-SR-008, July 2000

Forsberg, K., Mooz, H., Cotterman, H. Visualizing Project Management, 3rd edition, John Wiley and Sons, New York, NY, 2005. Pages 108–116, 242–248, 341–360.

Centers for Medicare & Medicaid Services (CMS) Office of Information Service (2008). Selecting a development approach. Webarticle. United States Department of Health and Human Services (HHS). Re-validated: March 27, 2008. Retrieved 27 Oct 2008.

余金山,“软件开发过程及其模型（I）”，华侨大学，1993.