Software Development Processes

Al-alas Mohammed

Zhejiang Normal University

Software Project Management

Abstract

According to architecture science which has a concept to build such as a house project begins with planned steps to reach the goal of the project, so in software Engineering has also the same concept, starts with planning ends by maintenance, and via this article, I navigate you to see this steps in variant models .

1. Introduction

Software Development 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.[1]

A software process is a set of related activities that leads to the production of a software product. These activities may involve the development of software from scratch in a standard programming language like Java or C. [2]

2. Software Process Models

Common methodologies include waterfall, prototyping, iterative, and incremental development, spiral development, rapid application development, extreme programming and various types of agile methodology. Some people consider a life-cycle "model" a more general term for a category of methodologies and a software development "process" a more specific term to refer to a specific process chosen by a specific organization. For example, there are many specific software development processes that fit the spiral life-cycle model.

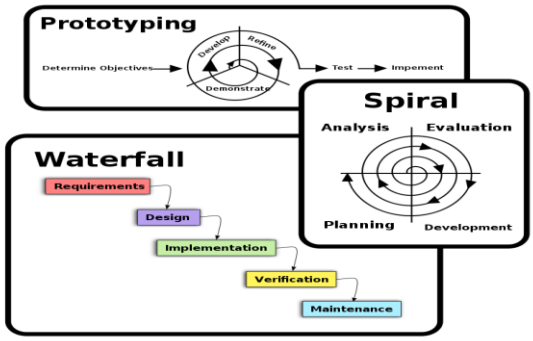

Figure 1

The three basic approaches applied to software development process

2.1 Waterfall Model

The first published model of the software development process was derived from more general system engineering processes. [2] This model is illustrated in Figure 2. Because of the cascade from one phase to another, this model is known as the 'waterfall model' or software life cycle. The waterfall model is an example of a plan-driven process — in principle, you must plan and schedule all of the process activities before starting work on them. [2]

The waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases, typically:

- Requirements analysis resulting in a software requirements specification
- Software design
- Implementation
- Testing
- Integration, if there are multiple subsystems
- Deployment (or Installation)
- Maintenance

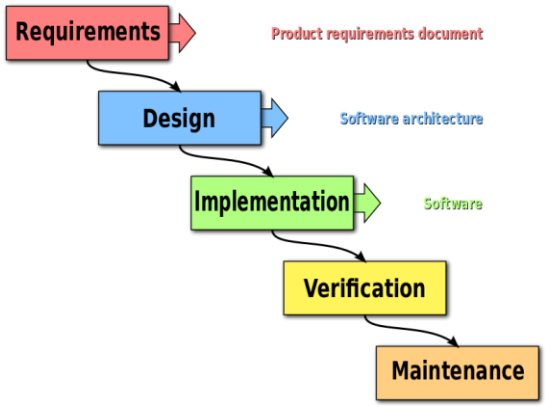

Figure 2 the activities of the software development process represented in the waterfall model.

However, the waterfall model reflects the type of process used in other engineering projects. As is easier to use a common management model for the whole project, software processes based on the waterfall model are still commonly used.

2.2. Prototyping

Often, a customer defines a set of general objectives for software, but does not identify detailed requirements for functions and features. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human-machine interaction should take. In these, and many other situations, a prototyping paradigm may offer the best approach. [3]

A software prototype can be used in a software development process to help anticipate changes that may be required:
1. In the requirements engineering process, a prototype can help with the elicitation and validation of system requirements.
2. In the system design process, a prototype can be used to explore particular software solutions and to support user interface design.

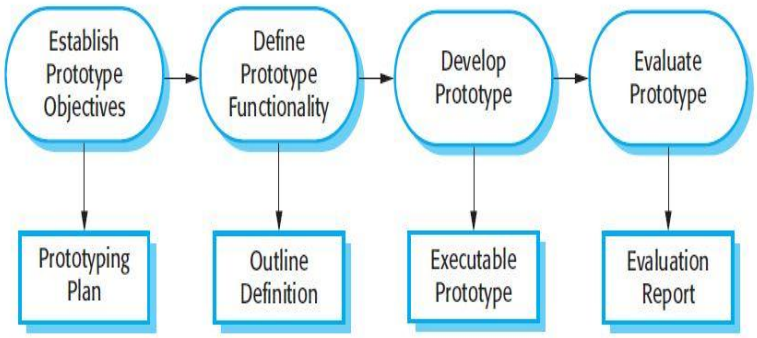

Figure 3 the process of prototype development

System prototypes allow users to see how well the system supports their work. They may get new ideas for requirements, and find areas of strength and weakness in the software. They may then propose new system requirements. Furthermore, as the prototype is developed, it may reveal errors and omissions in the requirements that have been proposed. A function described in a specification may seem useful and well defined. However, when that function is combined with other functions, users often find that their initial view was incorrect or incomplete. The system specification may then be modified to reflect their changed understanding of the requirements. [2]

A system prototype may be used while the system is being designed to carry out design experiments to check the feasibility of a proposed design. For example, a database design may be prototyped and tested to check that it supports efficient data access for the most common user queries. Prototyping is also an essential part of the user interface design process. Because of the dynamic nature of user interfaces, textual descriptions and diagrams are not good enough for expressing the user interface requirements. [2]

2.3. Rapid application development

Rapid application development (RAD) is a software development methodology, which favors iterative development and the rapid construction of prototypes instead of large amounts of up-front planning. The "planning" of software developed using RAD is interleaved with writing the software itself. The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements. [1]

The rapid development process starts with the development of preliminary data models and business process models using structured techniques. In the next stage, requirements are verified using prototyping, eventually to refine the data and process models. These stages are repeated iteratively; further development results in "a combined business requirements and technical design statement to be used for constructing new systems". [1]

The term was first used to describe a software development process introduced by James Martin in 1991. According to Whitten (2003), it is a merger of various structured techniques, especially data-driven Information Engineering, with prototyping techniques to accelerate software systems development. [1]

The basic principles of rapid application development are:

Key objective is for fast development and delivery of a high quality system at a relatively low investment cost.

Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.

Aims to produce high quality systems quickly, primarily via iterative Prototyping (at any stage of development), active user involvement, and computerized development tools. These tools may include Graphical User Interface (GUI) builders, Computer Aided Software Engineering (CASE) tools, Database Management Systems (DBMS), fourth-generation programming languages, code generators, and object-oriented techniques.

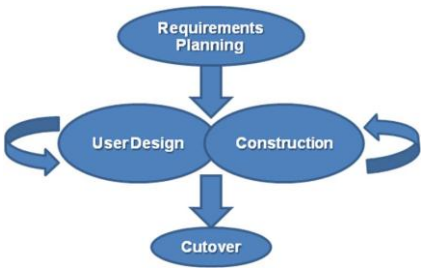

Figure 4 Rapid Application Development (RAD) Model

2.4. Spiral development

Originally proposed by Barry Boehm, the *spiral model* is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model. It provides the potential for rapid development of increasingly more complete versions of the software. Boehm describes the model in the following manner:
The spiral development model is a *risk*-driven *process model* generator that is used to guide multi-stakeholder concurrent engineering of software intensive systems. It has two main distinguishing features. One is a *cyclic* approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk. The other is a set of *anchor point milestones* for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions. [3]
In Figure 5 Here, the software process is represented as a spiral, rather than a sequence of activities with some backtracking from one activity to another. Each loop in the spiral represents a phase of the software process. Thus, the innermost loop might be concerned with system feasibility, the next loop with requirements definition, the next loop with system design, and so on.

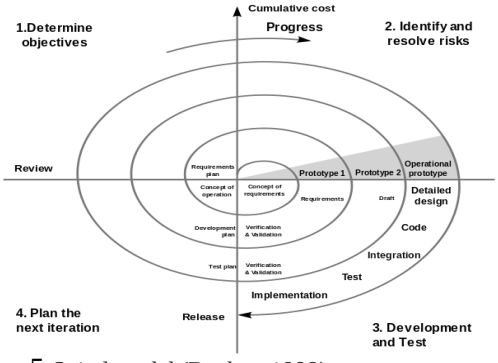

Figure 5 Spiral model (Boehm, 1988)

The main difference between the spiral model and other software process models is its explicit recognition of risk. A cycle of the spiral begins by elaborating objectives such as performance and functionality. Alternative ways of achieving these objectives, and dealing with the constraints on each of them, are then enumerated. Each alternative is assessed against each objective and sources of project risk are identified. The next step is to resolve these risks by information-gathering activities such as more detailed analysis, prototyping, and simulation.

2.5. Agile development

"Agile software development" refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve via collaboration between self-organizing cross-functional teams. The term was coined in the year 2001 when the Agile Manifesto was formulated.

Agile software development uses iterative development as a basis but advocates a lighter and more people-centric viewpoint than traditional approaches. Agile processes fundamentally incorporate iteration and the continuous feedback that it provides to successively refine and deliver a software system.

There are many agile methodologies, including:

- Dynamic systems development method (DSDM)
- Kanban
- Scrum

3. Conclusion

A variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses. One software development methodology framework is not necessarily suitable for use by all projects. Each of the available methodology frameworks are best suited to specific kinds of projects, based on various technical, organizational, project and team considerations.

4. References

[1]. https://en.wikipedia.org/wiki/Software_development_process

[2] Ian Sommerville, Software Engineering, 8 edition.

[3] Pressman, roger s. Software engineering: a practitioner's approach, seventh edition.