Software Development Processes

Konjoh Selabi Elvis Gerardin (2015219017)*
Zhejiang Normal University
Software Project Management

Abstract

In software engineering, a software development methodology (also known as a system development methodology, software development life cycle, software development process, 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.

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.

Keywords: software engineering, project management, software development process

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

- Requirements Analysis
- Specification
- Software architecture
- Implementation
- Testing
- Documentation
- Maintenance

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.

*e-mail:elvis.konjoh@yahoo.com

Software Project Management (2016/17)
Author Name: Konjoh Selabi Elvis Gerardin (2015219017)
University: Zhejiang Normal University
Title: Software Development Processes
Supervisor: Dr. Kenwright

2 Software Development Life Cycle

2.1 Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDP. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

2.2 Defining Requirements

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.

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through .SRS - Software Requirement Specification document which consists of all the product requirements to be designed and developed during the project life cycle.

2.3 Designing the product 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.

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification. This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity , budget and time constraints , the best design approach is selected for the product. A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

2.4 Building or Developing the Product

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

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle. Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java, and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

2.5 Testing the Product

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

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

2.6 Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations. business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing). Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

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.

3 Software Development Life Cycle Models

There are various software development life cycle models defined and designed which are followed during software development process. These models are also referred as "Software Development Process Models". Each process model follows a Series of steps unique to its type, in order to ensure success in process of software development.

Following are the most important and popular SDLC models followed in the industry:

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Mode

The other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

3.1 Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the

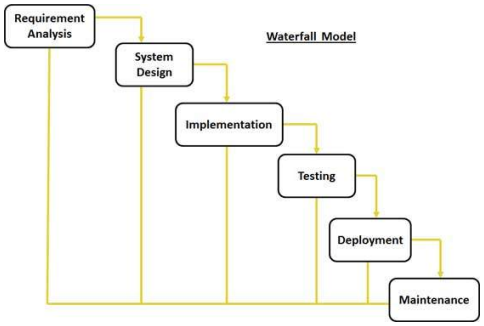

Figure 1: *Diagrammatic representation of different phases of waterfall model.*

phases. Waterfall model is the earliest SDLC approach that was used for software development . The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

Waterfall Model design Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Advantage : The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Disadvantage : The disadvantage of waterfall development is that it does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

3.2 Iterative Model

In Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed. An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Iterative Model design : Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

Advantage : The advantage of this model is that there is a working model of the system at a very early stage of development which makes

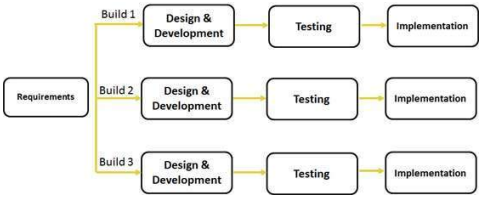

Figure 2: *Diagrammatic representation of different phases of Iterative model.*

it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

Disadvantage : The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

3.3 Spiral Model

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. Spiral model is a combination of iterative development process model and sequential linear development model i.e. waterfall model with very high emphasis on risk analysis. It allows for incremental releases of the product, or incremental refinement through each iteration around the spiral.

Spiral Model design The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals. 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.

Advantage : The advantage of spiral lifecycle model is that it allows for elements of the product to be added in when they become available or known. This assures that there is no conflict with previous requirements and design.

Disadvantage : This method is consistent with approaches that have multiple software builds and releases and allows for making an orderly transition to a maintenance activity. Another positive aspect is that the spiral model forces early user involvement in the system development effort. On the other side, it takes very strict management to complete such products and there is a risk of running the spiral in indefinite loop. So the discipline of change and the extent of taking change requests is very important to develop and deploy the product successfully.

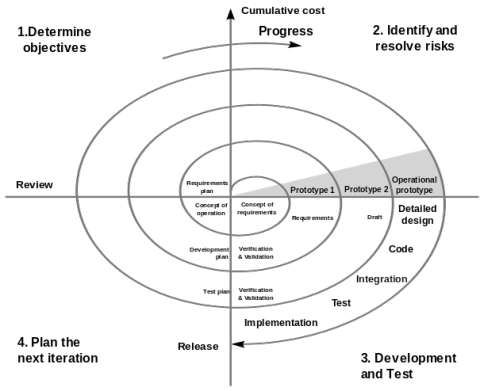

Figure 3: *Diagrammatic representation of different phases of Spiral model.*

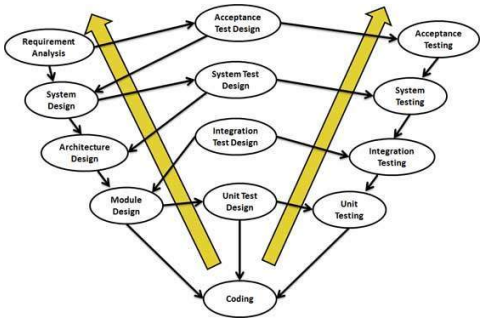

Figure 4: *Diagrammatic representation of different phases of V-Model.*

3.4 V-Model

The V - model is SDLC model where execution of processes happens in a sequential manner in V-shape. It is also known as Verification and Validation model. V - Model is an extension of the waterfall model and is based on association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle there is a directly associated testing phase. This is a highly disciplined model and next phase starts only after completion of the previous phase.

V-Model Design Under V-Model, the corresponding testing phase of the development phase is planned in parallel. So there are Verification phases on one side of the .V. and Validation phases on the other side. Coding phase joins the two sides of the V-Model.

Advantage : The advantage of V-Model is that it.s very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today.s dynamic world, it becomes very expensive to make the change.

3.5 Big Bang model

The Big Bang model is SDLC model where we do not follow any specific process. The development just starts with the required money and efforts as the input, and the output is the software developed which may or may not be as per customer requirement. Big Bang Model is SDLC model where there is no formal development followed and very little planning is required. Even the customer is not sure about what exactly he wants and the requirements are implemented on the fly without much analysis. Usually this model is followed for small projects where the development teams are very small.

Advantage : The advantage of Big Bang is that its very simple and requires very little or no planning. Easy to mange and no formal procedure are required.

Disadvantage : The Big Bang model is a very high risk model and changes in the requirements or misunderstood requirements may even lead to complete reversal or scraping of the project. It is ideal for repetitive or small projects with minimum risks.

4 Conclusion

This was about the various SDLC models available and the scenarios in which these SDLC models are used. The information in this tutorial will help the project managers decide what SDLC model would be suitable for their project and it would also help the developers and testers understand basics of the development model being used for their project.

We have discussed all the popular SDLC models in the industry, both traditional and Modern. This tutorial also gives you an insight into the pros and cons and the practical applications of the SDLC models discussed.

Waterfall and V model are traditional SDLC models and are of sequential type. Sequential means that the next phase can start only after the completion of first phase. Such models are suitable for projects with very clear product requirements and where the requirements will not change dynamically during the course of project completion.

Iterative and Spiral models are more accommodative in terms of change and are suitable for projects where the requirements are not so well defined, or the market requirements change quite frequently.

Big Bang model is a random approach to Software development and is suitable for small or academic projects.

Agile is the most popular model used in the industry. Agile introduces the concept of fast delivery to customers using prototype approach. Agile divides the project into small iterations with specific deliverable features. Customer interaction is the backbone of Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment.

RAD (Rapid Application Development) and Software Prototype are modern techniques to understand the requirements in a better way early in the project cycle. These techniques work on the concept of providing a working model to the customer and stockholders to give the look and feel and collect the feedback. This feedback is used in an organized manner to improve the product.

References

GEOFFREY ELLIOTT 2004. *Global Business Information Technology:an integrated systems approach*. Pearson Education. P.87

PANKAJ JALOTE 2005. Software Project Management in Practice P:19

Software development process,
`https://en.wikipedia.org/wiki/Software_deve~`

Software development Life Cycle,
`http://www.tutorialspoint.com/sdlc/index.htm`

What is software development process,
`http://www.selectbs.com/anal~-ment-process`