The Software Development Process

Chen Jiaqi[*]
Zhejiang Normal University
Software Project Management

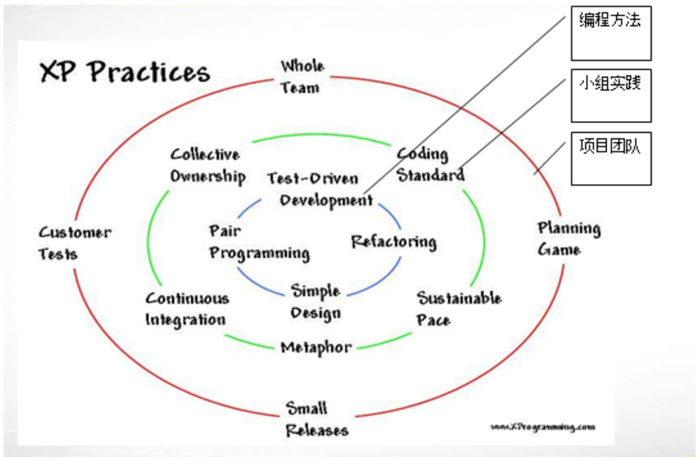

Figure 1: *The systems development life cycle (SDLC), also referred to as the application development life-cycle, is a term used in systems engineering, information systems and software engineering to describe a process for planning, creating, testing, and deploying an information system. The systems development life-cycle concept applies to a range of hardware and software configurations, as a system can be composed of hardware only, software only, or a combination of both.*

Abstract

The article gives a brief and concise explanation to the eight steps of software development process. Weather or not you are studying about the software project manager, this article will give you a clear view about the software development process. These article show the eight different types of the approaches of software development. Common methodologies include?waterfall,?prototyping,?iterative and incremental development,?spiral development,rapid application development,?extreme programming?and various types of?agile methodology. If you're experienced, you'll find this article a light refresher to the subject, and if you're deciding whether or not to delve into the field of Software development , this article may help you make that significant decision. For the sake of practicality, we discuss a variety of every steps in the process and every advantage or disadvantage of the different types of approaches.

Keywords: software engineering,waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, extreme programming

1 Introduction

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.

- What is software development processes?
- What are the software development approaches?
- What is the advantage and disadvantage of each approaches?
- How to control the process?

What's the article for: This article is for the introduce the different types of the methodologies.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.

2 Requirements analysis:

Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system

[*]e-mail:278831372@qq.com

Copyright 2016. The material in this article is copyrighted by the respected authors.
The article is based on work to support Software Project Management.
Software Project Management (2016/17)
Author Name: Chen Jiaqi
University: Zhejiang Normal University
Title: The Software Development Process
Supervisor: Dr. Kenwright

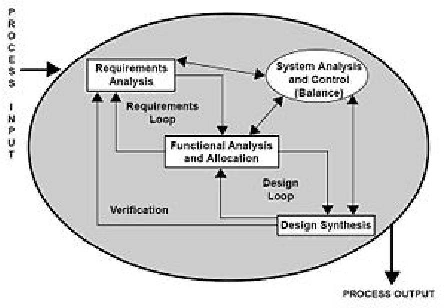

Figure 2: *A systems engineering perspective on requirements analysis.*

requirements.[Kotonya and Sommerville 2008]

3 High-level design:

High-level design (HLD) explains the architecture that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces. The HLD uses possibly nontechnical to mildly technical terms that should be understandable to the administrators of the system. In contrast, low-level design further exposes the logical detailed design of each of these elements for programmers.

4 Low-level design:

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work. Post-build, each component is specified in detail.[1]

The LLD phase is the stage where the actual software components are designed.

During the detailed phase the logical and functional design is done and the design of application structure is developed during the high-level design phase.

5 The Waterfall Model:

The waterfall model is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, production/implementation and maintenance.

The waterfall development model originates in the manufacturing and construction industries: highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Because no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.[Benington 2011]

In Royce's original waterfall model, the following phases are followed in order:

System and software requirements: captured in a product requirements document Analysis: resulting in models, schema, and business rules Design: resulting in the software architecture Coding: the development, proving, and integration of software Testing: the systematic discovery and debugging of defects Operations: the installation, migration, support, and maintenance of complete systems Thus the waterfall model maintains that one should move to a phase only when its preceding phase is reviewed and verified.

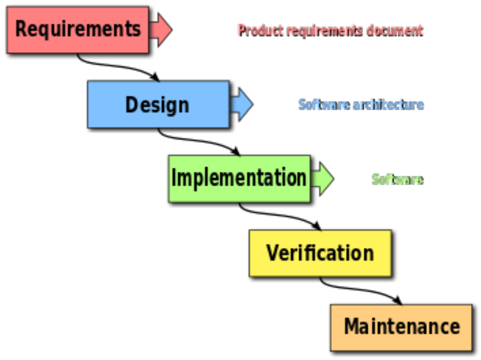

Figure 3: *The unmodified "waterfall model". Progress flows from the top to the bottom, like a cascading waterfall.*

Various modified waterfall models (including Royce's final model), however, can include slight or major variations on this process.[United States 1956]

6 Prototyping:

The original purpose of a prototype is to allow users of the software to evaluate developers' proposals for the design of the eventual product by actually trying them out, rather than having to interpret and evaluate the design based on descriptions. Prototyping can also be used by end users to describe and prove requirements that have not been considered, and that can be a key factor in the commercial relationship between developers and their clients.[MF 1991] Interaction design in particular makes heavy use of prototyping with that goal.

An early example of large-scale software prototyping was the implementation of NYU's Ada/ED translator for the Ada programming language.[Dewar 1990] It was implemented in SETL with the intent of producing an executable semantic model for the Ada language, emphasizing clarity of design and user interface over speed and efficiency. The NYU Ada/ED system was the first validated Ada implementation, certified on April 11, 1983.[SofTech Inc. 2003]

7 Incremental development:

Various methods are acceptable for combining linear and iterative systems development methodologies, with the primary objective of each being to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.

The basic principles are: A series of mini-Waterfalls are performed, where all phases of the Waterfall are completed for a small part of a system, before proceeding to the next increment, or Overall requirements are defined before proceeding to evolutionary, mini-Waterfall development of individual increments of a system, or The initial software concept, requirements analysis, and design of architecture and system core are defined via Waterfall, followed by iterative Prototyping, which culminates in installing the final prototype, a working system.

8 Iterative and incremental development:

Iterative and Incremental development is any combination of both iterative design or iterative method and incremental build model for software development. The combination is of long standing [SofTech Inc. 2003] and has been widely suggested for large development efforts. For example, the 1985 DOD-STD-2167 mentions (in section 4.1.2): "During

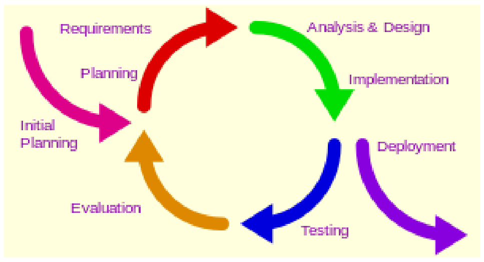

Figure 4: *The unmodified "waterfall model". Progress flows from the top to the bottom, like a cascading waterfall.*

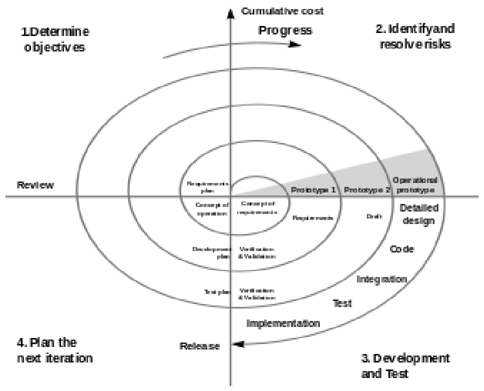

Figure 5: *The unmodified "waterfall model". Progress flows from the top to the bottom, like a cascading waterfall.*

software development, more than one iteration of the software development cycle may be in progress at the same time." and "This process may be described as an 'evolutionary acquisition' or 'incremental build' approach." The relationship between iterations and increments is determined by the overall software development methodology and software development process. The exact number and nature of the particular incremental builds and what is iterated will be specific to each individual development effort.

Iterative and incremental development are essential parts of the Modified waterfall models, Rational Unified Process, Extreme Programming and generally the various agile software development frameworks.

It follows a similar process to the plan-do-check-act cycle of business process improvement.

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. Authentic applications of the spiral model are driven by cycles that always display six characteristics. Boehm illustrates each with an example of a "hazardous spiral look-alike" that violates the invariant.["Boehm 2000]

- Define artifacts concurrently
- Perform four basic activities in every cycle
- Risk determines level of effort
- Risk determines degree of details
- Use anchor point milestones
- Focus on the system and its life cycle

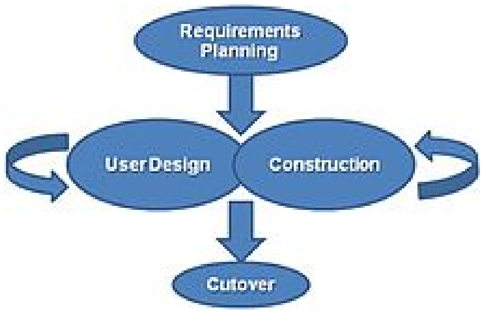

Figure 6: *The unmodified "waterfall model". Progress flows from the top to the bottom, like a cascading waterfall.*

10 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.

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".[Whitten 2003]

11 Agile software development:

Agile software development is a set of principles for software development in which requirements and solutions evolve through collaboration between self-organizing,[Collier 2011] cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change.[Alliance. 2013] Agile itself has never defined any specific methods to achieve this, but many have grown up as a result and have been recognised as being 'Agile'.

The Manifesto for Agile Software Development,[Beck 2011] also known as the Agile Manifesto, was first proclaimed in 2001, after "agile methodology" was originally introduced in the late 1980s and early 1990s. The manifesto came out of the DSDM Consortium in 1994, although its roots go back to the mid 1980s at DuPont and texts by James Martin and James Kerr et al.[Kerr 1993]

12 Lightweight methodologies:

A lightweight methodology is a software development method that has only a few rules and practices, or only ones that are easy to follow. In contrast, a complex method with many rules is considered a heavyweight methodology."

Examples of lightweight methodologies include:

- Adaptive Software Development by Jim Highsmith, described in his 1999 book Adaptive Software Development
- Crystal Clear family of methodologies with Alistair Cockburn,
- Extreme Programming (XP), promoted by people such as Kent Beck and Martin Fowler
- Feature Driven Development (FDD) developed (1999) by Jeff De Luca and Peter Coad

- ICONIX process, developed by Doug Rosenberg: An UML Use Case driven approach that purports to provide just enough documentation and structure to the process to allow flexibility, yet produce software that meets user and business requirements

Most of these lightweight processes emphasize the need to deal with change in requirements and change in environment or technology by being flexible and adaptive.

13 Conclusion

Software development is the process of computer programming, documenting, testing, and bug fixing involved in creating and maintaining applications and frameworks involved in a software release life cycle and resulting in a software product. The term refers to a process of writing and maintaining the source code, but in a broader sense of the term it includes all that is involved between the conception of the desired software through to the final manifestation of the software, ideally in a planned and structured process.Therefore, software development may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products.

Acknowledgements

I would like to thank my teacher B.Kenwright

References

ALLIANCE., A. 2013. *"What is Agile Software Development?"*. 3

BECK, K. 2011. *"Manifesto for Agile Software Development"*. 3

BENINGTON, H. D. 2011. *IEEE Annals of the History of Computing,"Production of Large Computer Programs"*. 2

"BOEHM, B. 2000. "spiral development: Experience, principles,and refinements". *" Special Report CMU/SEI-2000-SR-008"*. 3

COLLIER, K. W. 2011. *" Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing. "*. 3

DEWAR, ROBERT B. K.; FISHER JR., G. A. S. E. F. R. B. S. G. C. F. B. M. 1990. *"The NYU Ada Translator and Interpreter"*. ACM SIGPLAN Notices - Proceedings of the ACM-SIGPLAN Symposium on the Ada Programming Language 15. 2

KERR, JAMES M.; HUNTER, R. 1993. *"Inside RAD: How to Build a Fully Functional System in 90 Days or Less"*. 3

KOTONYA, G., AND SOMMERVILLE. 2008. *Requirements Engineering: Processes and Techniques Chichester*. John Wiley and Sons. 2

MF, S. 1991. *Software Prototyping: Adoption, Practice and Management.* 2

SOFTECH INC., WALTHAM, M. 2003. *"Ada Compiler Validation Summary Report: NYU Ada/ED, Version 19.7 V-001"*. 2

UNITED STATES, N. M. C. A. P. 1956. *Symposium on advanced programming methods for digital computers*. 2

WHITTEN, JEFFREY L.; LONNIE D. BENTLEY, K. C. D. 2003. *"Systems Analysis and Design Methods. "*. 3