Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 45 (2012) 7797-7800

ISSN: 2229-712X

Descriptive approach to software development life cycle models

Shaveta Gupta and Sanjana Taya

Department of Computer Science and Applications, Seth Jai Parkash Mukand Lal Institute of Engineering & Technology, Radaur, Distt. Yamunanagar (135001), Haryana, India.

ARTICLE INFO

Article history:

Received: 24 January 2012;

Received in revised form:

17 March 2012;

Accepted: 6 April 2012;

Keywords

Software Development Life Cycle,
Phases and Software Development,
Life Cycle Models,
Traditional Models,
Recent Models.

ABSTRACT

The concept of system lifecycle models came into existence that emphasized on the need to follow some structured approach towards building new or improved system. Many models were suggested like waterfall, prototype, rapid application development, V-shaped, top & Bottom model etc. In this paper, we approach towards the traditional as well as recent software development life cycle models.

© 2012 Elixir All rights reserved.

Introduction

The Software Development Life Cycle (SDLC) is the entire process of formal, logical steps taken to develop a software product. Within the broader context of Application Lifecycle Management (ALM), the SDLC is basically the part of process in which coding/programming is applied to the problem being solved by the existing or planned application.

The phases of SDLC can vary somewhat but generally include the following:

- conceptualization;
- requirements and cost/benefits analysis;
- detailed specification of the software requirements;
- software design;
- programming;
- testing;
- user and technical training; and
- Maintenance.

There are many methodologies or models that can be used to guide the software development lifecycle either as a core model to the SDLC or as a complementary method. Software Development Life Cycle also known as SDLC is the process of analysis, estimation, design, development, integration, testing and implementation of software system. Proper management of Software Development Life Cycle is critical to success of any software development project. Software Development Life Cycle can also be thought as a concept of providing complete support to a product, all the way through its phases of evolution.

A software life cycle model depicts the significant phases or activities of a software project from conception until the product is retired. It specifies the relationships between project phases, including transition criteria, feedback mechanisms, milestones, baselines, reviews, and deliverables. Typically, a life cycle model addresses the following phases of a software project: requirements phase, design phase, implementation, integration, testing, operations and maintenance. Much of the motivation behind utilizing a life cycle model is to provide structure to avoid the problems of the "undisciplined hacker".

Software Development Life Cycle Models

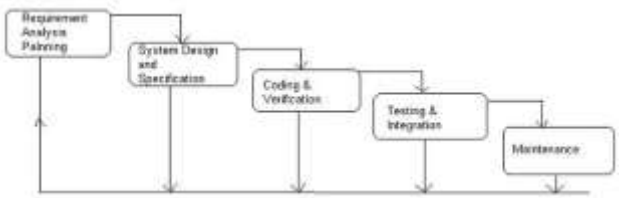
Software Development Life Cycle Model is an abstract representation of a development process. SDLC models can be categorized as:

- Traditional Software Models (Waterfall, V-Model, CMM, RUP Model)
- Recent Software Models (Prototyping software life cycle model, Incremental Model ,Spiral Model, Rapid Application Development Model, Joint Application Development Model(JAD), Dynamic Systems Development Method (DSDM), Object-Oriented Model, Agile Model etc.)
- **Traditional Software Models**

Traditional software development methodologies include Waterfall model, V-Model, Capability Maturity Model and Rational Unified Process (RUP).

➢ **Waterfall Model**

This is most popular traditional model, disciplined and sequential approach to software development. In this model, each phase must be completed in its entirety before the next phase can begin. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. Waterfall discourages revisiting and revising any prior phase once it's complete. This "inflexibility" in a pure Waterfall model has been a source of criticism by supporters of other more "flexible" models.



Waterfall Life Cycle Model

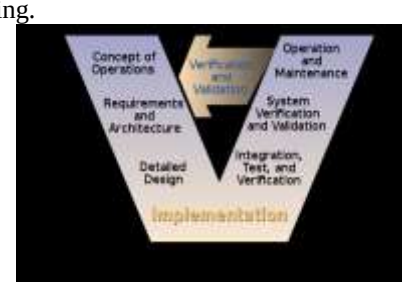
➢ **V-Shaped Model**

The V-model can be thought of as an extension of the waterfall, mapping test phases to the phases of the development cycle .Instead of moving down in a linear way, the process steps

E-mail addresses: goel.shaveta03@gmail.com , sanjana.taya@gmail.com

© 2012 Elixir All rights reserved

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.



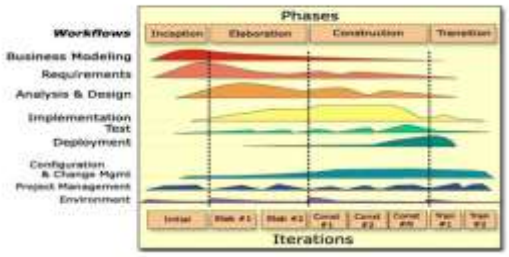
V-Shaped Life Cycle Model

➢ **Capability Maturity Model-**

A maturity model can be viewed as a set of structured levels that describe how well the behaviors, practices and processes of an organization can reliably and sustainable produce required outcomes. A maturity model can be used as a benchmark for comparison and as an aid to understanding - for example, for comparative assessment of different organizations where there is something in common that can be used as a basis for comparison. In the case of the CMM, for example, the basis for comparison would be the organizations' software development processes. There are five levels defined along the continuum of the CMM and, according to the SEI: "Predictability, effectiveness, and control of an organization's software processes are believed to improve as the organization moves up these five levels

➢ **Rational Unified Process Model-**

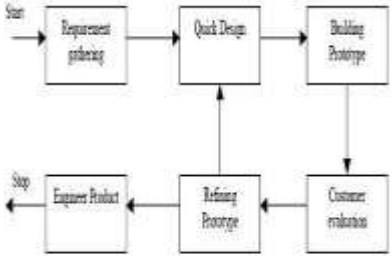
RUP is an adaptable process framework allowing organizations to select elements of processes that are most relevant for their project. Rational Unified Process is an iterative adaptive software process. Iteration consists of nine disciplines. Out of these nine disciplines, six are engineering disciplines like Business Modeling, Requirements, Design and Analysis, Implementation, Test, Deployment and remaining there are supporting disciplines which consist of Configuration and Change Management, Environment and Project Management. These iterations have to be followed with guidelines and templates at each stage. This way RUP provides a set of standards to be adhered to for all the stages of the System Development Life Cycle (SDLC). The RUP consists of a life cycle with four distinct phases as shown below in two dimensional diagram of RUP.



Recent Software Development Models-

➢ **Prototyping Software Life Cycle Model**

Instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. Development of the prototype obviously undergoes design, coding and testing but each of these phases is not done very formally or thoroughly.



There are two main versions of prototyping model:

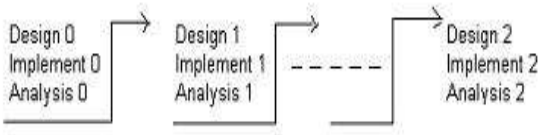
Version I: Prototyping is used as a requirements technique.

Version II: Prototype is used as the specifications or a major part thereof.

If the first version of the prototype does not meet the client's needs, then it must be rapidly converted into a second version.

➢ **Incremental Model**

The incremental model is an intuitive approach to the waterfall model. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into easily managed iterations. A working version of software is produced during the first iteration, subsequent iterations build on the initial software produced during the first iteration.



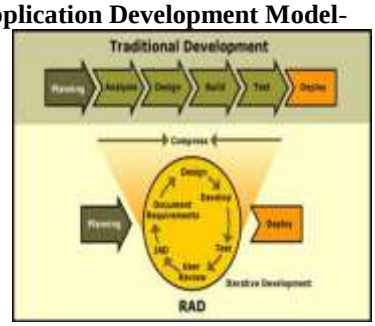
Incremental Life Cycle Model

➢ **Spiral Model**

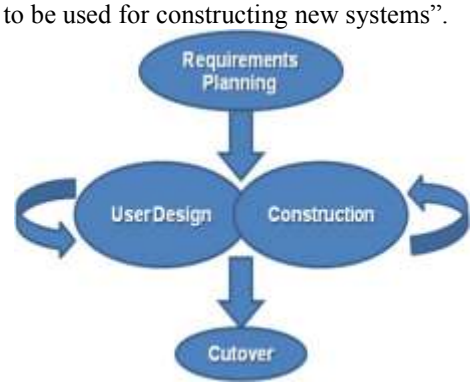
The activities in this model can be organized like a spiral, with more emphases placed on risk analysis. Four phases involved here are shown below and software project repeatedly passes through these phases in iterations. The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. A prototype is produced at the end of the risk analysis phase. Software is produced in the engineering phase, along with testing at the end of the phase. The evaluation phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral. In the spiral model, the angular component represents progress, and the radius of the spiral represents cost.

Spiral Life Cycle Model

- **Rapid Application Development Model-**

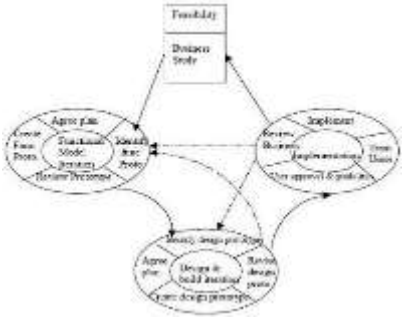


Rapid application development is a software development methodology, which involves iterative development and the construction of prototypes. It is a merger of various structured techniques, especially the data driven Information Engineering with prototyping techniques to accelerate software systems. The development process starts with the development of preliminary data models and business process model 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".



- **Dynamic System Development Method (DSDM)-**

The Dynamic System Development Method (DSDM) is dynamic as it is a Rapid Application Development method that uses incremental prototyping. This method is particularly useful for the systems to be developed in short time span and where the requirements cannot be frozen at the start of the application building. Whatever requirements are known at a time, design for them is prepared and design is developed and incorporated into system. In Dynamic System Development Method (DSDM), analysis, design and development phase can overlap. Like at one time some people will be working on some new requirements while some will be developing something for the system. In Dynamic System Development Method (DSDM), requirements evolve with time. Dynamic System Development Method (DSDM) has a five-phase life cycle.



- **Joint application development (JAD)-**

Joint Application Development, or JAD, is a process originally developed for designing a computer-based system. It brings together business area people (end users) and IT (Information Technology) professionals in a highly focused workshop. The advantages of JAD include a dramatic shortening of the time it takes to complete a project. It also improves the quality of the final product by focusing on the up-front portion of the development lifecycle, thus reducing the likelihood of errors that are expensive to correct later on.

JAD is a requirements-definition and software system design methodology in which stakeholders, subject matter experts (SME), end-users, software architects and developers attend intense off-site meetings to work out a system's details. JAD focuses on the business problem rather than technical details. It is most applicable to the development of business systems. It produces its savings by shortening the elapsed time required to gather a system's requirements and by gathering requirements better, thus reducing the number of costly, downstream requirements changes. Its success depends on effective leadership of the JAD sessions; on participation by key end-users, executives, and developers; and on achieving group synergy during JAD sessions. JAD works best when combined with an incremental-development lifecycle model such as Rational Unified Process.

- **Agile Model-**

Agile software development is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross- functional teams. The Agile methods are focused on different aspects of the software development life-cycle. Some focus on the practices (extreme programming, pragmatic programming, agile modeling), while others focus on managing the software projects (Scrum). Yet, there are approaches providing full coverage over the development life cycle (DSDM, RUP), while most of them are suitable from the requirements specification phase on (e.g. FDD). In the Agile method the customer and developers are in close communication, where as in the traditional method, the "customer" is initially represented by the requirement and design documents.

Conclusion

The "one size fits all" approach to applying SDLC methodologies is no longer appropriate (Lindvall & Rus, 2000). Each SDLC methodology is only effective under specific conditions. Traditional SDLC methodologies are often regarded as the proper and disciplined approach to the analysis and design of software applications (Rothi & Yen, 1989) like waterfall, staged and phased development, transformational, spiral, and iterative models. The new methods like Agile, Dynamic System Development Method, Rational Unified Process were developed to efficiently manage software projects subjected to short timelines and excessive uncertainty and change. These methodologies include their simpler processes and easier acceptance by developers who are only familiar with code and fix techniques. They are most appropriate when there are uncertain and volatile requirements, responsible and motivated developers, and customers who wish to become involved.

Lightweight methodologies re-examine the traditional assumptions that have historically been made about the commitment of resources to requirements analysis and process improvement (Yourdon, 2000). Traditional SDLCs operate on the fundamental assumption that it is worth investing resources to identify a flaw in a process because the process will be used over and over again. On the other hand, lightweight SDLCs recognize that when everything is changing and there is no assurance that processes will be reused that it makes little sense to expend the effort.

Bibliography

Lindvall, M., & Rus, I. (2000, July/August). Process diversity in software development. *IEEE Software, 17*(4), 14-18.

Rothi, J., & Yen, D. (1989). System Analysis and Design in End User Developed Applications. *Journal of Information Systems Education*. Retrieved April 7, 2001, from the World Wide Web: http://www.gise.org/JISE/Vol1-5/SYSTEMAN.htm.

Yourdon, E. (2000, October). The Emergence of "Light" Development Methodologies. *Software Productivity Center*. Retrieved March 11, 2001, from the World Wide Web: http://www.spc.ca/resources/essentials/oct1800.htm#3.