Volume 3, Issue 3, March 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Empirical Analysis on Conventional Model of Software Development Processes in Business Process Re-engineering

Bechoo Lal [1st ,]
Research Scholar, Dept. of Computer Science & Engineering
JJT University Rajasthan, India

Dr. Chandrahauns R Chavan [2nd,]
Associate Professor and Former Director of,
JBIMS, University of Mumbai
Mumbai, India

Abstract: ***Software engineering has emerged as a discipline where engineering principles such as modeling, prototyping, designing, developing, testing, delivery, installing and maintaining are the broad steps blend together to achieve engineering objective. Software engineering tends to be closer to science essential because of its scientific and systematic approach and use of the standards, protocols, tools and techniques in the process of development. In this paper the researcher will try to optimize the methodology of the software development process on the basis of existing development concept.***

Keywords: RAD, SDM, SSDM, CASE, DBMS, AUP, JAD

INTRODAUCTION

In the history of software engineering the software engineering has evolved steadily from its founding days in the 1940s until today in the 2010s. Applications have evolved continuously. The ongoing goal to improve technologies and practices, seeks to improve the productivity of practitioners and the quality of applications to users. There are a number of areas where the evolution of software engineering is notable:

- Emergence as a profession: By the early 1980s[1], software engineering had already emerged as a bona fide profession, to stand beside computer science and traditional engineering. See also software engineering professionalism.
- Role of women: In the 1940s, 1950s, and 1960s, men often filled the more prestigious and better paying hardware engineering roles, but often delegated the writing of software to women. Grace Hopper, Jamie Fenton and many other unsung women filled many programming jobs during the first several decades of software engineering. Today, many fewer women work in software engineering than in other professions, a situation whose cause is not clearly identified. It is often attributed to sexual discrimination, cyber culture or bias in education. Many academic and professional organizations consider these situations unbalanced are trying hard to solve it.
- Processes: Processes have become a big part of software engineering and are hailed for their potential to improve software and sharply criticized for their potential to constrict programmers.

A. Current Trend in Software

Software engineering is a young discipline, and is still developing. The directions in which software engineering is developing include:

Aspects: Aspects help software engineer's deal with quality attributes by providing tools to add or remove boilerplate code from many areas in the source code. Aspects describe how all objects or functions should behave in particular circumstances. For example, aspects can add debugging, logging, or locking control into all objects of particular types. Researchers are currently working to understand how to use aspects to design general-purpose code. Related concepts include generative programming and templates.

Agile: Agile software development guides software development projects that evolve rapidly with changing expectations and competitive markets. Proponents of this method believe that heavy, document-driven processes (like Tick IT, CMM and ISO 9000) are fading in importance [*citation needed*]. Some people believe that companies and agencies export many of the jobs that can be guided by heavy-weight processes. Related concepts include Extreme Programming, Scrum, and Lean software development.

B. Problem Statement

The current scenario of the software development process is very expensive and time taking process and it is far away from the reliability and feasibility. The researcher is trying to find out the optimization of software development process which is the essential in the development phase. The problem can be stated on behalf of the following situation:

- Due to high cast of software
- Lack of feasibility and reliability
- Time taking process

- Lack of interaction with the customer
- Complexity of existing development process

RELATED WORK

A. Experimental: Experimental software engineering is a branch of software engineering interested in devising experiments on software, in collecting data from the experiments, and in devising laws and theories from this data. Proponents of this method advocate that the nature of software is such that we can advance the knowledge on software through experiments only.

B. Model-driven: Model Driven Design develops textual and graphical models as primary design artifacts. Development tools are available that use model transformation and code generation to generate well-organized code fragments that serve as a basis for producing complete applications.

C. Software Product Lines: Software Product Lines is a systematic way to produce families of software systems, instead of creating a succession of completely individual products. This method emphasizes extensive, systematic, formal code reuse, to try to industrialize the software development process.

D. Software Development Methodology: One of the oldest software development tools is flowcharting, which has its roots in the 1920s. The software development methodology didn't emerge until the 1960s. According to Elliott (2004) the Systems development life cycle (SDLC) can be considered to be the oldest formalized methodology for building information systems. The main idea of the SDLC has been "to pursue the development of information systems in a very deliberate, structured and methodical way, requiring each stage of the life cycle from inception of the idea to delivery of the final system, to be carried out in rigidly and sequentially".[2] The main target of this methodology in the 1960s has been "to develop large scale functional business systems in an age of large scale business conglomerates. Information systems activities revolved around heavy data processing and number crunching routines".[2]

E. Specific Software Development Methodologies:

1970s

- Structured programming since 1969
- Cap Gemini SDM, originally from PANDATA, the first English translation was published in 1974. SDM stands for System Development Methodology

1980s

- Structured Systems Analysis and Design Methodology (SSADM) from 1980 onwards

1990s

- Object-oriented programming (OOP) has been developed since the early 1960s, and developed as the dominant programming methodology during the mid-1990s.
- Rapid application development (RAD) since 1991.
- Scrum (development), since the late 1990s
- Team software process developed by Watts Humphrey at the SEI

2000s

- Rational Unified Process (RUP) since 1998.
- Extreme Programming since 1999
- Agile Unified Process (AUP) since 2005 by Scott Ambler
- Integrated Methodology (QAIassist-IM) since 2007

SIGNIFICANCE OF STUDY

Every software development methodology has more or less its own approach to software development. There is a set of more general approaches, which are developed into several specific methodologies. These approaches are: [1]

- Waterfall: linear framework type.
- Prototyping: iterative framework type
- Incremental: combination of linear and iterative framework type
- Spiral: combination of linear and iterative framework type
- Rapid Application Development (RAD): Iterative Framework Type
- Extreme Programming

IV. Software Development Model

The waterfall model is a sequential development process, in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance. The first formal description of the waterfall model is often cited to be an article published by Winston W. Royce in 1970 although Royce did not use the term "waterfall" in this article.[3]

Basic principles of the waterfall model are: [1]

- Project is divided into sequential phases, with some overlap and splash back acceptable between phases.

- Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
- Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

A. Prototyping

Software prototyping is the framework of activities during software development of creating prototypes, i.e., incomplete versions of the software program being developed.

Basic principles of prototyping are: [1]

- Not a standalone, complete development methodology, but rather an approach to handling selected portions of a larger, more traditional development methodology (i.e. Incremental, Spiral, or Rapid Application Development (RAD)).
- Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
- User is involved throughout the process, which increases the likelihood of user acceptance of the final implementation.
- Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users' requirements.
- While most prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system.
- A basic understanding of the fundamental business problem is necessary to avoid solving the wrong problem.
- Mainframes have a lot to do with this sort of thing that consist of: PB&J

B. Incremental

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.

Basic principles of incremental development are: [1]

- A series of mini-Waterfalls are performed, where all phases of the Waterfall development model are completed for a small part of the systems, before proceeding to the next incremental, or
- Overall requirements are defined before proceeding to evolutionary, mini-Waterfall development of individual increments of the system, or
- The initial software concept, requirements analysis, and design of architecture and system core are defined using the Waterfall approach, followed by iterative Prototyping, which culminates in installation of the final prototype (i.e., working system).
-

C. Spiral Model

The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. Basic principles: [1]

- 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 portion 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."[4]
- 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.[5]

D. Rapid Application Development (RAD)

Rapid Application Development (RAD) is a software development methodology, which involves iterative development and the construction of prototypes. Rapid application development is a term originally used to describe a software development process introduced by James Martin in 1991.

E. Basic principles

- 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 through the use of 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.

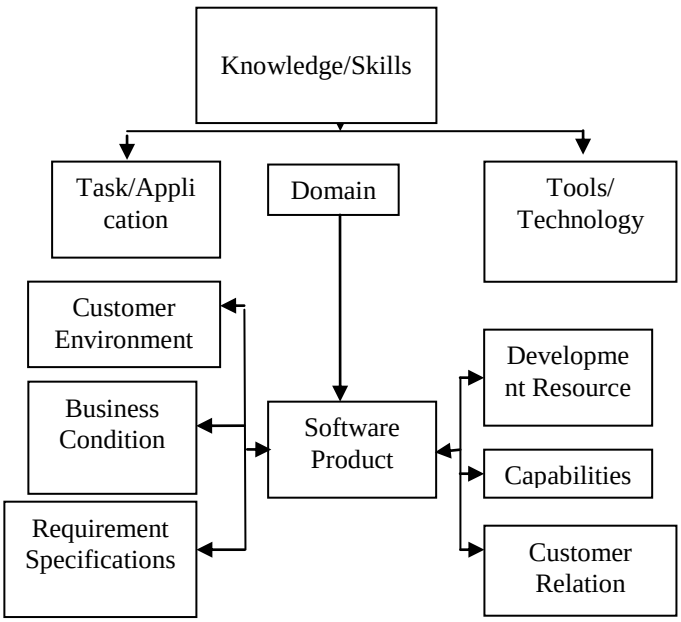
- Key emphasis is on fulfilling the business need, while technological or engineering excellence is of lesser importance.
- Project control involves prioritizing development and defining delivery deadlines or "time boxes". If the project starts to slip, emphasis is on reducing requirements to fit the time box, not in increasing the deadline.
- Active user involvement is imperative.
- Iteratively produces production software, as opposed to a throwaway prototype.
- Produces documentation necessary to facilitate future development and maintenance.
- Standard systems analysis and design techniques can be fitted into this framework.

F. Other Software Development Approaches

- Object oriented development methodologies, such as Grady Booch's Object-oriented design (OOD), also known as object-oriented analysis and design (OOAD). The Booch model includes six diagrams: class, object, state transition, interaction, module, and process.[7]
- Agile Software Development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. The term was coined in the year 2001 when the Agile Manifesto was formulated.
- Integrated Methodology Software Development refers to a group of software development practices and deliverables that can be applied in a multitude (iterative, waterfall, spiral, agile) of software development environments, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

V. Proposed Software Development Process

The proposed Software development process is a structure imposed on the development of a software product. Synonyms include software life cycle and *software* process. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process.



FigV.1: Proposed Software Development Process

This model will provide the best software solution if these key areas are managed well Further, management of these areas is effective if the organization and its developers have knowledge of the domain, application, tools and technology.

Conclusion and Future Aspects

The software development process is the essential in the development environment. There must be interaction between developers and customers. The proposed development method is optimized and fills the gap between customers and developers. The researcher is trying to simulate the software development process in the development environment. So that it can provide more flexibility, reliability for the customer as well as developers.

© 2013, IJARCSSE All Rights Reserved ***Page | 213***

ACKNOWLEDGMENT

Dr. Chandrahauns R. Chavan, I/c Professor-cum-Director , Alkesh Dinesh Modi Institute for Finacial and Management Studies, University of Mumbai and

Associate Professor and former Director , Jamnalal Bajaj Institute of Mnagement Studies, University of Mumbai,Bechoo Lal, Research Scholar Dept. of Computer Science & Engineering JJT University, Rajasthan India, currently working as a asst. professor in Western College, University of Mumbai.

REFERENCES

[1] SELECTING DEVELOPMENT APPROACH. Revalidated: March 27, 2008. Retrieved 27 Oct 2008.
[2] Geoffrey Elliott (2004) Global Business Information Technology. p.87.
[3] Wasserfallmodell > Entstehungskontext, Markus Rerych, Institut für Gestaltungs- und Wirkungsforschung, TU-Wien. Accessed on line November 28, 2007.
[4] (Boehm, 1986)
[5] (Boehm, 1986 and 1988)
[6] (Boehm, 2000)
[7] Georges Gauthier Merx & Ronald J. Norman (2006). Unified Software Engineering with Java. p.201.
[8] Edward J. Barkmeyer ea (2003). Concepts for Automating Systems Integration NIST 2003.
[9] Paul R. Smith & Richard Sarfaty (1993). Creating a strategic plan for configuration management using Computer Aided Software Engineering (CASE) tools. Paper For 1993 National DOE/Contractors and Facilities CAD/CAE User's Group.
[10] Kuhn, D.L (1989). "Selecting and effectively using a computer aided software engineering tool". Annual Westinghouse computer symposium; 6-7 Nov 1989; Pittsburgh, PA (USA); DOE Project.
[11] P.Loucopoulus and V. Karakostas. System Requirement Engineering.
[12] CASE definition In: Telecom Glossary 2000. Retrieved 26 Oct 2008.
[13] K. Robinson (1992). Putting the Software Engineering into CASE. New York : John Wiley and Sons Inc.
[14] Xiao He (2007). "A metamodel for the notation of graphical modeling languages". In: Computer Software and Applications Conference, 2007. COMPSAC 2007 - Vol. 1. 31st Annual International, Volume 1, Issue , 24–27 July 2007, pp 219-224.
[15] . www.Google.co.in
[16] www.wikipedia.org

