IJITKM Volume 8 • Number 1 June-Dec 2014 pp. 30-34 (ISSN 0973-4414)

Contrastive Analysis of Software Development Methodologies

Ramesh Kumar[1], Dr. Rajesh Verma[2]
[1]Research Scholar, School of Computer Scienc & IT, Singhania University, Rajasthan, INDIA
[2]Research Supervisor, Department of Comp. Sc., Govt. College, Indri, Karnal, Haryana
ramesh.banwala@yahoo.co.in

Abstract: - Software methodologies are concerned with the process of creating software - not so much the technical side but the organizational aspects. There are different software development methodologies. Each one is unique because of its emphasis on process versus data and the order and focus it places on each Software Development Life Cycle. Several software development methods and associated modeling tools have evolved to deal with issue of complexity. This paper introduces and compares software development methodologies. This information will help you recognize which methodologies may be best suited for use in various situations. The target audience is all personnel involved in software development for the DoD. No attempt has been made to cover methodologies that are most applicable to smaller projects that can be accomplished by five or fewer software engineers. You may learn about several of these in since the intended readers should consider government software standards in their use of methodologies, background on these standards is included. The selection criteria is based on the following factors; Clarity of User Requirements, Familiarity with Technology, System Complexity, System Reliability, Short Time Schedules and
Schedule Visibility.
Keywords**: -** SDLC, Structured, Agile, RAD, Methodology.

Introduction

A system development methodology refers to the framework that is used to structure, plan, and control the process of developing an information system. A wide variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses. One system development methodology is not necessarily suitable for use by all projects. A series of activities aimed to help you grasp a range of traditional and non-traditional system development methodologies, and communicate your learning to your peers. Software is used not only to provide applications on our desktop PC, or distributed business application across a network of machines, but also to control many systems around us. [1] A software development methodology is a division of software development work into distinct phases or activities with the intent of better planning and management. It is often considered a subset of the systems development life cycle. If the methodologies are applied by different groups of people, similar software will be produced the key that drives software engineering are Cost, schedule and quality [2]. Ideally all software products should be released without errors. But this is far from reality because residual errors in applications are to be expected. For example errors in applications like medical control software, car break system, and air traffic control software. For these kinds of systems, the cost of software failure is dangerously high and therefore a much higher degree of confidence in the correctness of the software is required. . This is true for large and small projects; custom built and packaged: local and global. These underlying skills remain largely unchanged over time, but the actual techniques and approaches that analysts and developers use do change often dramatically over time [3].Software engineering is concerned with addressing the challenges in Software development.

Need for Software Development Methodology

- According to the Standish group 2012 study, 39% of all projects are delivered on time, on budget, 43% were challenged which are late, over budget, and 18% failed which are cancelled prior to completion or delivered and never used." [4]
- A study conducted by the Forrester research group states that nearly one-third of all IT projects commenced would be an average of three months late. In many cases the failure is the result of either not using a methodology or using the wrong methodology [5].
- According to CIO.com, nearly three quarters of all IT projects in the Internet era that were conceived in the last seven years have suffered from one or more of the following: total failure, cost overruns, time overruns, or a rollout with fewer features or functions than promised
- Since the approach to developing complex hardware systems was well understood, it provided a model for developing software.
- These methods approached system development in a requirements/design/build paradigm with standard, well-defined processes. These methodologies are referred to as Heavy Methodologies or plan driven.

- However, newer methodologies have started making an appearance in software projects. These methodologies unlike the more classical ones are considered to be more agile and more able to adapt to change. They do not focus on a long development cycle but rather on short iterations, lightweight processes and rely heavily on close customer involvement. These types of methodologies have come to be known as Light or Agile Methodologies

System Development Methodologies

Three major categories of System Development Methodologies in terms of the progression through the SDLC phases and emphasis placed on each phase are; Structured design Structured design methodologies adopt a formal step-by-step approach to the SDLC that moves logically from one phase to the next. This design methodology introduces the use of formal modeling or diagramming techniques to describe a system's basic business processes and follows a basic approach of two structured design categories. It includes waterfall development and parallel development.

1) Waterfall development: With waterfall development- based methodologies, the analysts and users proceed sequentially from one phase to the next. The two key advantages of waterfall development-based methodologies are, the system requirements are identified long before programming begins. Changes to the requirements are minimized as the project proceeds. [6]
   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".[4].

   Advantages
   1. Simple goal, simple to understand and use.
   2. Clearly defined stages and easy to arrange tasks.
   3. Easy to manage. Each phase has specific deliverable and a review.
   4. Works well for projects where requirements are well understood.
   5. Works well when quality is more important than cost/schedule.

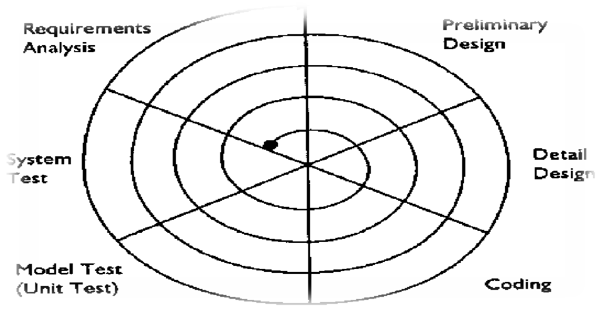
   Disadvantages
   1. The design must be completely specified before programming begins
   2. Not suitable for complex projects
   3. The deliverables are often a poor communication mechanism, so important requirements can be overlooked in the documentation
   4. Not suitable for projects of long duration because in long running projects requirements are likely to change.
   5. Users rarely are prepared for their introduction to the new system, which occur long after the initial idea for the system was introduced.

Spiral Methodology

The Spiral Lifecycle Model is a sophisticated lifecycle model that focuses on early identification and reduction of project risks. A spiral project starts on a small scale, explores risks, makes a plan to handle the risks, and then decides whether to take the next step of the project - to do the next iteration of the spiral. It derives its rapid development benefit not from an increase in project speed, but from continuously reducing the projects risk level - which has an effect on the time required to deliver it. Success at using the Spiral Lifecycle Model depends on conscientious, attentive, and knowledgeable management .It can be used on most kinds of projects, and its risk-reduction focus is always beneficial.

Basic Principal:-

1. Focus is on risk assessment and on minimizing project risk by breaking a project into hange during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project n throughout the life cycle.
2. "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." (Boehm, 1986).



Advantages of Spiral model:

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

Disadvantages of Spiral model:

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

Rapid Application Development was a response to non-agile processes developed in the 1970s, such as the Waterfall model. The problem with previous methodologies was that applications took so long to build that requirements had changed before the system was complete, often resulting in unusable systems. RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

When to use RAD model:

- RAD should be used when there is a need to create a system that can be modularized in two or three months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time.

Rapid Application Development systems commonly have these advantages: increased speed of development and increased quality. The speed increases can be achieved using a variety of methods including, rapid prototyping, virtualization of system related routines, the use of CASE tools and other techniques. Quality, as defined by RAD, is both the degree to which a delivered application meets the needs of users as well as the degree to which a delivered system has low maintenance costs. RAD increases quality through the involvement of the user in the analysis and design stages. Some systems also deliver advantages of interoperability, extensibility, and portability.

Early RAD systems had two primary disadvantages: reduced Scalability, and reduced features. Reduced scalability occurs because a RAD developed application starts as a prototype and evolves into a finished application. Reduced features occur due to time boxing, where features are pushed to later versions in order to finish a release in a short amount of time.

Agile Software Development Methodology

Agile software development is a conceptual framework for undertaking software engineering projects. There are a number of agile software development methodologies e.g. Crystal Methods, Dynamic Systems Development Model (DSDM), and Scrum. Most agile methods attempt to minimize risk by developing software in short time boxes, called iterations, which typically last one to four weeks. Each iteration is like a miniature software project of its own, and includes all the tasks necessary to release the mini-increment of new functionality: planning, requirements analysis, design, coding, testing, and documentation. While iteration may not add

enough functionality to warrant releasing the product, an agile software project intends to be capable of releasing new software at the end of every iteration. At the end of each iteration, the team reevaluates project priorities.
Agile methods emphasize real time communication, preferably face-to-face, over written documents. Most agile teams are located in a bullpen and include all the people necessary to finish the software. At a minimum, this includes programmers and the people who define the product such as product managers, business analysts, or actual customers. The bullpen may also include testers, interface designers, technical writers, and management.
Agile methods also emphasize working software as the primary measure of progress. Combined with the preference for face-to-face communication, agile methods produce very little written documentation relative to other methods.

Object –Oriented Methodologies (OOM)
The first software development methodologies termed as object-oriented were in fact hybrid partly structured and partly object-oriented. The analysis phase was typically done using structured analysis techniques (SA), producing Data Flow Diagrams, Entity-Relationship Diagrams, and State Transition Diagrams, whereas the design phase was mainly concerned with mapping analysis results to an object-oriented blueprint of the software. These methods were hence categorized as transformative. The methods prescribed by Seidewitz and Stark and Alabiso are the main methodologies in this category. Object-Oriented concept was made to balance the emphasis between process and data. It utilizes the Unified Modelling Language (UML) to describe the system concept a collection of self-contained objects, including both data and processes. Some UML tools used are;
1) Use case diagram: This is used to note the type of users of a system and what each type of user does with that system
2) Activity diagram: This shows the work flow of the system; that is, it shows the flow of control from activity to activity in the system.
3) Sequence diagram: This shows the explicit sequence of messages that are passed between objects in the defined interaction. They are helpful for understanding real-time specifications and complex use cases.
4) Class diagram: This is used to define and create specific instances or object. It reflects the classes and relationships that are needed for the set of use cases which describes the system.
5) Behavioral State diagram: This examines the behavior of one class. In order words it shows the different states that a single instance of a class passes through during its life in response to events, along with the responses and actions. A state is a set of values that describes an object at a specific point in time, and it represents a point in an object's life in which it satisfies some condition, performs some action, or waits for something to happen. Package diagram: This depicts the dependencies between the packages that make up the model.
6) Deployment diagram: This shows the physical architecture of the system. It can also be used to show software components being deployed onto the physical architecture [7].
OOM is good for modelling real-world systems. This is because in modelling real world system, processes and data are so closely related that it is difficult to pick one or the other as the primary focus. Based on this lack of congruence with the real world, new OOM have emerged that use the RAD- based sequence of SDLC phases but attempts to balance emphasis between process and data. The difference between OOM and traditional approaches like structured design is how a problem is decomposed. In traditional approaches, the problem decomposition is either process - centric or data – centric, while OOM represent the system concept in terms of process and data. Using Object Oriented Analysis and Design methods to develop real-time systems has the potential to produce safer, more reliable and maintainable code.

Comparison of Development Methodology
Selecting a methodology is not simple, as no one methodology is always best. Many organizations have their own standards. Each methodology is suitable for some context, and the main reason for studying different methodologies is to develop the ability to choose the proper model for a given project. The choice of a methodology is influenced by several factors: Clarity of the user requirements; familiarity with the base technology; system complexity; need for system reliability; time pressure; and need to see progress on the time schedule [3]

Selection Methodology Criteria

	Structured Methodology	RAD Methodology		Object oriented Methodology	Agile Methodology
To develop system with	Waterfall	Phased	Prototyping	Object oriented	XP
Reliable	Good	Good	Bad	Excellent	Good
Unclear user Requirements	Bad	Good	Excellent	Excellent	Excellent
Unfamiliar Technology	Bad	Good	Bad	Good	Bad
Schedule visibility	Bad	Excellent	Excellent	Good	Good

Conclusion: - Unfortunately when developing software there are many more variables and unknowns than in a simple taxi trip. In this strange world the Station Hotel may not be stationary hotel - it can move or even disappear or there can be a myriad of hotels all seemingly the same. The roads may be unmapped and always changing. I should also mention that there are things that can go wrong that are beyond the bounds of any development methodology - the Station Hotel may explode, who knows? There are car accidents and break downs. I hope this paper has given you insight into the different software development methodologies. Research has shown that may software projects have failed and the field of software engineering is concerned with developing and maintaining software systems that behave reliably and efficiently, are affordable to develop and maintain, satisfy all the requirements that customers have defined for them. It is important because of the impact of large, expensive software systems and the role of software in safety-critical applications.

Reference

[1] C. Quentin, and A. Kans, (2004). Formal software Development from VDM to Java. Palgrave Macmillian
[2] J. Pankaj, (2005). Software Requirements Analysis and Specification an Integrated
[3] Approach to Software Engineering, Springer Science Business Media, Inc, Third Edition, 2005.
[4]. http://www.sandishgroup.com
[5]. Hoffman,T. (2003, July 21). Value of Project Management Offices questioned. Computerworld. Vol. 37, Iss. 29, pg. 7 2. Berinato, S. (2001). The Secret to Software Success http://www.cio.com/archive /070101/secret.htm
[6]Markus Rerych, Institute für Gestaltungs- und Wirkungsforschung, TU-Wien. Accessed on line November 28, 2007.
[7]. Ambler, S., "Enhancing the Unified Process", Software Development, October 1999, pp. 33-39.
[8] A. Dennis, B. Wixom, and R. Roth, (2006). System Analysis and Design John Wiley and Sons, Inc pg 171-209.