ISSN:0975-9646

Explore 10 Different Types of Software Development Process Models

S.Thulasee Krishna [1],Dr. S.Sreekanth[2] ,K.Perumal[1,] K.Rajesh Kumar Reddy[1]
[1]*Dept of CSE*
Kuppam Engineering College, Kuppam,
[2]*Dept of MCA*
Sitams, Chittoor, AP,India

Abstract— **The development lifecycle of software Comprises of four major stages namely Requirement Elicitation, Designing, Coding and Testing. A software process model is the basic framework which gives a workflow from one stage to the next. This workflow is a guideline for successful planning, organization and final execution of the software project. Generally we have many different techniques and methods used to software development life cycle. Project and most real word models are customized adaptations of the generic models while each is designed for a specific purpose or reason, most have similar goals and share many common tasks. This paper will explore the similarities and difference among these various models.**

Keywords*—Rapid application development model, concurrent development process model ,formal model ,CSDM*

I. INTRODUCTION

Software processes performed during software Development and evolution are becoming rather complex and resource-intensive. They involve people who execute actions with the primary goal to create quality software in accordance with the previously set user requirements. only structured, carefully guided and documented software processes can lead to the stated goal. Constant monitoring and improvement of software processes is therefore of a significant interest for organizational performing software development and maintenance. in order to improve the process an objective description and evolution of the existing process is needed.

What is a software process models:

In contrast to software life cycle models, software process models often represent a networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution. Such models can be used to develop more precise and formalized descriptions of software life cycle activities. Their power emerges from their utilization of a sufficiently rich notation, syntax, or semantics, often suitable for computational processing. Software process networks can be viewed as representing multiple interconnected task chains [1]. Task chains represent a non-linear sequence of actions that structure and transform available computational objects (resources) into intermediate or finished products. Non-linearity implies that the sequence of actions may be non-deterministic, iterative, accommodate multiple/parallel alternatives, as well as partially ordered to account for incremental progress. Task actions in turn can be viewed a non-linear sequences of primitive actions which denote atomic units of computing work, such as a user's selection of a command or menu entry using a mouse or keyboard. Winograd and others have referred to these units of cooperative work between people and computers as "structured discourses of work" [2]. While task chains have become popularized under the name of "workflow" .Task chains can be employed to characterize either prescriptive or descriptive action sequences. Prescriptive task chains are idealized plans of what actions should be accomplished, and in what order. For example, a task chain for the activity of object-oriented software design might include the following task actions:

- Develop an informal narrative specification of the system.
- Identify the objects and their attributes.
- Identify the operations on the objects.
- Identify the interfaces between objects, attributes, or operations.
- Implement the operations.

Clearly, this sequence of actions could entail multiple iterations and non-procedural primitive action invocations in the course of incrementally progressing toward an object-oriented software design. Task chains join or split into other task chains resulting in an overall production network or web. The production web represents the "organizational production system" that transforms raw computational, cognitive, and other organizational resources into assembled, integrated and usable software systems. The production lattice therefore structures how a software system is developed, used, and maintained. However, prescriptive task chains and actions cannot be formally guaranteed to anticipate all possible circumstances or idiosyncratic foul-ups that can emerge in the real world of software development. Thus, any software production web will in some way realize only an approximate or incomplete description of software development. Articulation work is a kind of unanticipated task that is performed when a planned task chain is inadequate or breaks down. It is work that represents an open-ended non-deterministic sequence of actions taken to restore progress on the disarticulated task chain, or else to shift the flow of productive work onto some other task chain. Thus, descriptive task chains are employed to characterize the observed course of events and situations that emerge when people try to follow a planned task sequence. Articulation work in the context of software evolution includes actions

people take that entail either their accommodation to the contingent or anomalous behavior of a software system, or negotiation with others who may be able to affect a system modification or otherwise alter current circumstances. This notion of articulation work has also been referred to as software process dynamism.

II. TEN DIFFERENT SOFTWARE PROCESS MODELS

A. Waterfall Process Model

The Classical Life Cycle or the Waterfall Process Model [3] was the first process model to present a sequential framework, describing basic stages that are mandatory for a successful software development model. It formed the basis for most software development standards and consists of the following phases: Requirements elicitation, Designing, Implementation and Testing.

Advantages of waterfall model:

- Simple goal.
- Simple to understand and use.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are Well documented.
- Easy to manage. Each phase has specific deliverable and a review.
- Works well for projects where requirements are well understood.
- Works well when quality is more important then cost/schedule.
- Customers/End users already know about it.

Listed below are some flaws:

- Rigid design and inflexible procedure[4].
- Restricting back & forth movement from a later stage to a former one. when new requirements surface accommodating those with existing ones become difficult due to restrictions in looping back to prior stages.
- Waterfall Model faced ***"inflexible point solutions"*** which meant that even small amendments in the design were difficult to incorporate later in design phase.
- As the requirements were frozen before moving to the design phase, using the incomplete set of requirement, a complete design was worked on. Such an approach worked normally well for a small project requiring average amendments. In case of a large project, completing a phase and then moving back to reconstruct the same phase, incurred a large overhead [4].
- Once a phase is done, it is not repeated again that is movement in the waterfall goes one to the next and the vice versa is not supported. deadlines are difficult to meet in case of large projects [9].

B. Prototype Model

In Prototype Model [6], the user is given a "look and feel" of the system using a prototype. The prototype for the system to be developed is built, tested and reworked as necessary. Prototype process model is suitable for dynamic environment where requirements change rapidly. The process begins with gathering main functional requirements; this is followed by a quick design leading to the development of a prototype. The prototype is then evaluated by users and customers. Developers rework on the prototype until the customer and users are satisfied.

Advantages of prototype model:

- Users/customers own requirements.
- Instills customer confidence that the "right" product is being built.
- Provides a good way to determine requirements when t here uncertainty about what is needed.

The prototype can face the following limitations:

- The main limitation of this model includes lack of information about the exact number of iterations and the time period required to upgrade the prototype in order to bring it up to the satisfaction of the user and the customer.
- Developers are in such a rush that they hardly consider all the functionalities of the prototype. In order to release the product as soon as possible, the prototype with some additions is released on or before the target release date. This happens due to lack of user analysis activities; the end product contains features the user is hardly aware how to use.
- Often the developers make implementation compromises in order to make the prototype work quickly, which will lead to the use of inappropriate operating system or programming language [7].
- The premature prototypes lack key consideration like security, fault tolerance, distributed processing and other such key issues [9].

C. Incremental Development Model

In incremental development process [5], customers identify, in outlined the services to be provided by the system. They identify which of the services are most important and which are least important to them. A number of delivery increments are then defined which each increment providing a subset of functional requirements. The highest priority functional requirements are delivered first.

Advantages of incremental development model:

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.

- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during an iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment operational product is delivered.
- Issues, challenges & risks identified from each increment can be utilized/applied to the next increment.

The disadvantages of the model are:

- It is difficult to map requirements directly to different increments. Include excessive user involvement. Poorly defined scope as scope of the product may vary increment to increment.
- An overhead in the model is rapid context switching between various activities. Each iteration is followed by an evaluation ensuring that user requirements have been met [8]. This evaluation after each iteration is time consuming.

D. Spiral Model

In Spiral model [10], instead of presenting a sequence of activities with some backtracking from one activity to the other, the process model followed a spiral organization of activities. It combines characteristics of both prototype and waterfall process model. The model is divided into some task regions, which are as follows: Customer Communication, Planning, Risk Analysis, and Engineering, Construction and release and Customer evaluation. The distinctive feature of this model is that each stage is controlled by a specific risk management criteria ensuring decision making using critical factors. .

Advantages of spiral model:

- Changing requirements can be accommodated.
- Allows for extensive use of prototypes
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided in to smaller parts and more risky parts can be developed earlier which helps better risk management.

The following disadvantages are identified in this model:

- A number of risks, constraints, alternatives, models etc. need to be analyzed but never are these risks or objectives listed and no specific risk analysis technique is mentioned. If risk analysis is poor the end product will surely suffer.
- Another difficulty of the spiral model is adjustment of contract deadlines using the spiral model.Risk analysis expertise is vital. For large projects expert software developers can produce efficient software products but in case of a complex large project absence of specific risk analysis techniques and presence of varying expertise can create a chaos [10].

E. Rapid Application Development Model

The RAD model [11] is an adaptation of the classical model for achieving rapid development using component based construction. If requirements are well understood with a well constrained project scope, the RAD process enables delivery of the fully function system. The model is considered to be incremental development model and that have emphasis on short development cycle.

Advantages of rapid application development model:

- Time to deliver is less.
- Changing requirements can be accommodated.
- Progress can be measured.
- Cycle time can be short with use of powerful RAD tools.
- Productivity with fewer people in short time.
- Use of tools and frameworks.

Rapid Application Development has following drawbacks:

- Reduced features occur due to time boxing, where features are delayed to later versions in order to deliver basic functionality within abbreviated time.
- Reduced scalability occurs because a RAD developed application starts as a prototype and evolves into a finished application using existing component and their integration.
- RAD, for large projects, requires a Sufficient number of human resources also requiring existence of components for reuse. Also RAD is not suitable for all types of application development [11].
- High technical risks discourage RAD use. This is because use of new technology in a software is difficult in a changing global software market [11].

F. Rational Unified Process Model

The RUP [5] provides dynamic, static and practice perspectives of a product. The RUP provides each team member with the guidelines, templates and tool mentors necessary for the entire team to take full advantage of the best practices. The software lifecycle is broken into cycles, each cycle working on a new generation of the product.

This phased model identifies four discrete phases:

- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

Advantages of Rational unified process model:

- This is a complete methodology in itself with an emphasis on accurate documentation.
- It is proactively able to resolve the project risks associated with the client's evolving requirements requiring careful change request management.

The identified drawbacks of the process are:

- Each phase has a milestone which needs to be satisfied for the next particular phase to start.

- If the respective milestone of the particular phase is not satisfied the entire project might get cancelled or re-engineered before proceeding further.
- The satisfaction criteria of a particular milestone has its own constraints and are not listed specifically [12].

G. The V-Model

The V-Model [13] is an extension to the Waterfall Model in that it does not follow a sequential mode of execution rather it bends upward after the coding phase to form V shape.

Advantages of V-model:

- The users of The V-Model participate in the development and maintenance of The V-Model. A change control board publicly maintains the V-Model. The change control board meets once a year and processes all received change requests on The V-Model.
- At each project start, the V-Model can be tailored into a specific project V-Model, this being possible because the V-Model is organization and project independent.
- The V-Model provides concrete assistance on how to implement an activity and its work steps, defining explicitly the events needed to complete a work step: each activity schema contains instructions, recommendations and detailed explanations of the activity.

It has the following drawbacks;

- It addresses software development within a project rather than a whole organization.
- The V-Model is not complete as it argues to be, as it covers all activities at too abstracts level.

H. Concurrent Engineering Model

The concurrent development model sometimes called concurrent engineering model can be represented schematically as a series of frame work activities, software engineering action and task, and their associated status . Provide a schematic representation of one software engineering task with in the modeling activities for the concurrent process model. The activity-modeling may be in any one of the states noted at any given time. Similarly, other activities or task can be represented in an analogous manner. All activities exist concurrently but reside in different states .its first iteration and exist in the waiting changes state. The modeling activities which existed in the none state while initial communication was completed, now makes a transition into the under development state. if, however, the customer indicates that changes in requirements must be made, the modeling activities moves from the under development states into the awaiting changes states. The concurrent process model defines a series of events that will trigger transition from state to state for each of the software engineering activities, actions, or tasks. The concurrent model is applicable to all types of software development and provides an accurate picture of the current state of a project. Rather than confining software engineering activities, actions, or task on the network exists simultaneously with other activities, actions, or tasks. Events generated at one point in the process network trigger transitions among the states.

Advantages of concurrent engineering model :

- The concurrent development model, some times called concurrent engineering. It's can be represented schematically as a series of frame work activities, software engineering actions, software engineering task and their associated states.
- The concurrent process model defines a series of events that will trigger transition from state to state for each of the software engineering activities and action or task.
- The concurrent process model is applicable to all types of software development and provides an accurate picture of the current state of a project.

The identified drawbacks of the process are:

- The SRS must be continually updated to reflect changes.
- It requires discipline to avoid adding too many new features too late in the project.

I. Confident Software Development Process Model

The Confident process model which we have proposed has seven phases, namely; Feasibility study/Requirement, Requirement Based Analysis, Logical Design, Confident Code, Logical Testing, Implementation & Deployment, and Maintenance. It is a flexible model not restricting the developers enabling them to move both Front and back from any given stage to any other stage during its lifecycle. Each phase is further divided into sub phases, each specifying a criterion which has to be met to move to the next phase.

Advantages of Confident development process model:

- all upcoming requirements to be frozen and to be accommodated in later versions of the product.
- we haven't frozen requirement phase, one can move easily from design phase to requirement phase if a new requirement(s) surfaces.

These identified drawbacks of the confident development process are:

J. The Formal model

The formal methods model encompasses a set of activities that leads to formal mathematical specification of computer software formal methods enable a software engineer to specify, develop, and verify a computer-based system by applying a rigorous. When formal methods are used during development, they provide a mechanism for eliminating many of the problem that are difficult to overcome using other software engineering cardiogram

Advantages of formal model :

- Formal specification encourage rigour. Often , the very process of construction of arigorou. Specification is more important then formal specification .
- Formal methods usually have a founded mathematic basis.
- Formal methods have well defined semantics therefore , ambiguity in specification automatically avoided when one formally specifies a system.
- Formal specification can be executed to obtain immediate feedback on the future of the specified system [14].

These identified drawbacks of the process are:

- The development of formal models is currently quite time consuming and expensive
- It is difficult to use the models as a communication mechanism for technically unsophisticated customers.

III. CONCLUSION

The paper has demonstrated ,it proved useful for process comprehension and analysis through simulation and validation. Model simulation can be used to identify processes flaws deficiencies and bottlenecks, to estimate the impact of potential changes to the process and to compare alternative process models without putting the new process into practice. This paper is very very useful to the developers

REFERENCES:

[1] Kling, R., and W. Scacchi, The Web of Computing: Computer Technology as Social Organization, Advances in Computers, 21, 1-90, Academic Press, New York, 1982.

[2] Winograd, T. and F. Flores, *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishers, Lexington, MA, 1986.

[3] W.W. Royce, "*Managing the Development of Large Software Systems: Concepts and Techniques*", IEEE, IEEE Computer Society, August 1970, pp. 1-9.

[4] B.W. Boehm, "*A Spiral Model for Software Development and Enhancement*", IEEE, IEEE Computer Society, vol. 21, issue 5, May 1988, pp. 61 – 72.

[5] Ian Sommerville, " Software Engineering", 8th Edition, 2006, pp. 89.

[6] R.J. Madachy, "*Software Process Dynamics*", New Jersey: Willey Inter science, 2007, pp. 31.

[7] R.S. Pressman, "*Software Engineering, A Practitioner's Approach*", 5th ed. New York: McGraw-Hill, 2001, pp. 32.

[8] E.I. May, B. A. Zimmer, "*The Evolutionary Development Model for Software*", Hewlett-Packard Journal, Article 4, August 1996, pp. 1-8.

[9] B.W. Boehm, "*Anchoring the Software Process*", IEEE, IEEE Software, vol. 13, issue 4, July 1996, pp. 73-83.

[10] R.J. Madachy, "*Software Process Dynamics*", New Jersey: Willey Inter science, 2007, pp. 33.

[11] R.S. Pressman, "*Software Engineering, A Practitioner's Approach*", 5th ed. New York: McGraw-Hill, 2001, pp. 34.

[12] P. Kruchten, "*Rational Unified Process Best Practices for Software Development Teams"*, Canada: rational Software, 2001.

[13] Jeff Tian, Southern Methodist University, Dallas, "Software Quality Engineering", IEEE, Computer Society Publication, Willy Inter Science, 2005.

[14] Rajib Mall, *"Fundamental of software engineering " 2 nd edition . prentice- hall, pp. 101-102.*

AUTHORS:

S.THULASI KRISHNA received the B.Tech. Degree in Computer Science and Engineering from Jawaharlal Nehru University ,Hyderabad, India in 2005, M.E from Sathyabama University Chennai ,and Ph.D pursing from Rayalaseema University ,Kurnool. He joined as Asst.professor in VIST Engineering College in august 2005, Hyderabad. He was worked as Asso.Professor in Vidyanikethan Engineering College Tirupathi. And currently working as Associate Professor in Kuppam Engineering College. He is member of International Association of Engineering.

Dr. S. SREEKANTH has obtained Ph.D. Degree from S.V.University, Tirupathi. He is working as a Professor & Director in MCA Department, SITAMS, Chittoor, Andhra Pradesh, with the experience of 16 years. He has published 13 research papers both in international and national journals of computer science.

K.RAJESH KUMAR REDDY received the B.Tech. Degree in Computer Science and Engineering from Jawaharlal Nehru University, Anantapur in 2009, M.Tech from Jawaharlal Nehru University, Anantapur in 2011 and currently working as Assistant Professor in Kuppam Engineering College. He is member of International Association of Engineering.

K.PERUMAL received the B.E degree in Information technology from Anna University, Chennai in 2010,and M.E pursing from Anna University. He is working as Asst.Professor in Kuppam Engineering College.