Indian J.Sci.Res. 14 (1): 27-37, 2017 ISSN: 0976-2876 (Print)
ISSN: 2250-0138 (Online)

SYSTEMS DEVELOPMENT METHODOLOGIES: CONCEPTUAL STUDY

K. SARAVANAN[1]

Assistant Professor, Sree Vidyanikethan Institute of Management, India

ABSTRACT

Today, we are in the era of Mobile computing. There is an increasing trend in the use of software applications through smart phones, laptops, PDA's, tablets and other mobile devices. To fulfill this increasing demand in the market, software applications are getting developed and upgraded in rocket speed. IT Companies are employing various system development methodologies to develop quality software. Development methodologies and practices serve as one of the critical components in Systems development. Over the years, several methodologies have evolved to cater to the varying requirements of systems development and two styles of system development have emerged – the conventional closed-source development and the progressive open-source development. Today, open source development is adopted as supplement to closed source development. This paper has two objectives. The first objective is to review the literature related to system development methodologies that have evolved over the years. The second objective is to make distinction between the two styles of development, cite examples of companies which are successful in adopting open source development.

KEYWORDS: Closed Source development, Open Source development, Systems development methodology, Systems development style

Systems development methodology (SDM) is a standard process followed in an organization to conduct all the steps necessary to analyze, design, implement and maintain information systems (IS) [1]. It is highly beneficial for organizations to adopt a systems development methodology to develop IS.

Systems development life cycle (SDLC) is a framework composed of distinct steps or phases in the development of an Information System [5]. SDLC consists of five stages which include Planning, Analysis, Design, Implementation and Maintenance as shown in Figure 1.

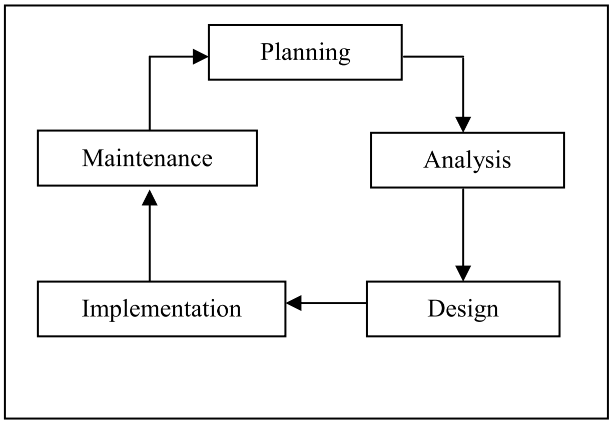


Figure 1: Systems Development Life Cycle

During 'Planning', the IS planned for development is identified and prioritized. This is followed by 'Analysis' where the requirements of the system to be developed will be determined and structured by Systems Analysts (SA). Once the requirements are structured, the Inputs, Interfaces, databases and Outputs of the IS are 'designed'. The baselined design serves as input for 'Implementation' where full fledged coding and testing takes place. Implementation also includes Documentation and Training. 'Documentation' on the system is prepared and the system is installed for use. 'Training' on how to use the system is provided to customers. This completes the Systems development and the system is moved to 'Maintenance' phase where improvement to the existing system is undertaken.

REVIEW OF LITERATURE

The evolution of Systems development methodologies dates back to 1970 when W.W.Royce introduced the traditional Waterfall model of systems development [8]. Till that time, systems were developed adopting less disciplined approaches. Only formal methods using mathematics and component based concepts of software development were adopted during those times.

With system development being made as a formalized process through the introduction of waterfall model, many models started emerging either to fix the drawbacks of waterfall model or to improve the efficiency and quality of the overall systems development process.

[1]Corresponding author

Figure 2 and table 1 provide a snapshot on the various systems development methodologies that have evolved along with the timelines. It can be observed that till late 1990s, the paradigm was conventional closed-source development and today we are witnessing progressive open-source development acting as supplement to closed-source development

Till 1970s	• Formal Methods • Component based development concepts
1970 - 1993	• Linear Sequential Models • Incremental Iterative Models • OOAD • Specialized Methods
1993 - 2000	• Agile Methods • SOA • Open Source development
2000s	• Recent Agile Methods

Figure 2: Evolution of SDM

Table 1: SDM – Year and Contribution

SDM	**Year**	**Contribution**
Formal Methods	1967	Robert W.Floyd
Component based Development	1968	Douglas Mcllroy
Waterfall Model	1970	W.W.Royce
Prototype Model	1970s	Not Available
Joint Application Design and Development	1974	Dan Gielan, Chuck Morris, Tony Crawford
V-Model	1982	Ottobrunn
Spiral Model	1986	Barry Boehm
Aspect Oriented Software Development	1990s	Gregor Kiczales
Rapid Application Development	1991	James Martin
W-Model	1993	Paul Herzlich
Scrum	1993	Jeff Sutherland and Ken Schwaber
Concurrent Development	1994	Davis and Sitaram
Dynamic Systems Development Method	1994	DSDM consortium
Rational Unified Process	1996	Philippe Kruchten
Service Oriented Architecture	1996	Gartner
Feature Driven Development	1997	Jeff de Luca
Crystal Methodology	1998	Alistair Cockburn
Open Source Development	1998	Eric S.Raymond
eXtreme Programming	1999	Kent Beck
Adaptive Software Development	1999	Jim Highsmith
Agile Unified Process	2002	Scott Ambler
Test Driven Development	2003	Kent Beck
Behavior Driven Development	2003	Dan North
Kanban Software Development	2004	David Anderson
Disciplined Agile Delivery	2009	Scott Ambler and Mark Lines
Scrumban	2009	Corey Ladas

In this paper, review of literature on SDM is done in three parts based on the evolution of development style. The first part discusses on SDM which originated in the period 1970 to 1993. During this period, the style was completely closed source development. The second part focuses on discussing SDM originated between 1994 till 2002. In this period, open source style of development emerged and agile methodologies became popular. The third part focuses on methodologies developed after 2003 where open source development is adopted as supplement to closed source development.

SDM from 1970 to 1993

Till early 1990s, six development methodologies were adopted by organization Till Linear Sequential models originated, systems development was less formalized. Concepts of. Formal methods and component based development concepts were in use but systems development became more formal after the introduction of Waterfall model in 1970 by W.W.Royce.

Linear Sequential Methods

Linear Sequential methods of Systems development advocate gathering the requirements well in advance of the Systems life cycle and baseline the same. Customers are not appreciated to change the requirements and are not involved during Systems Development. Once the System is developed, customers were allowed to use the system and requested to provide feedback. These methods are highly useful in larger and complex projects where the requirements are well known in advance and the team size is more than hundred. The problem with the traditional method is that as customers are not involved in the life cycle, maintenance cost is very high. Three variations of linear sequential methods include Waterfall Model, V-Model and W-Model.

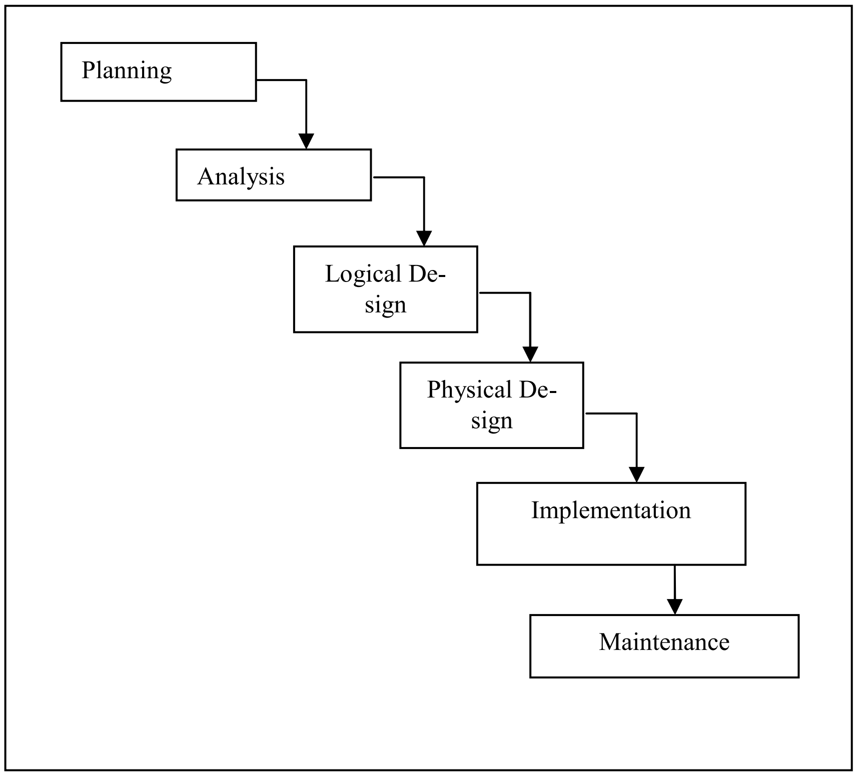
Waterfall Model

In waterfall model, each phase of development should be completed for the next phase to begin as shown in Figure 3. It follows a down-hill fashion and each phase interacts with the next phase through documentation. Proposed by W.W.Royce [1], this model is useful in situations where requirements do not change and work proceeds in a linear fashion.

V- Model

V-Model is a variation of the Waterfall model aimed at improving the quality of system by giving more focus to testing throughout the lifecycle. In this model, each activity of development in the left side has a corresponding testing activity in the right side as shown in Figure 4.

Coding activity in the left side has Unit Testing (UT) activity performed in the right side. Low level design (LLD) documents serve as input for component Testing (CT). High level design (HLD) is the input for Integration Testing (IT). System Testing (ST) is performed by considering Systems Requirement Specifications (SRS) as input. Acceptance Testing (AT) is performed by customers based on Business Requirements Specification (BRS) document. V-Model is a highly successful model and is widely adopted in companies which have a dedicated testing team



.

Figure 3: Waterfall Model

W- Model

Developed by Paul Herzlich in 1983, W-Model is an attempt to address the shortcomings of V-Model. Unlike V-Model which gave more importance to dynamic testing, W-Model focuses both on static testing as well as dynamic testing. Every development activity is mirrored by a testing activity such that static testing is focused during development and dynamic testing is focused during testing phase of systems development.

Incremental-Iterative Methods

One of the problems with traditional methods is the lack of customer involvement throughout the system life cycle which led to higher maintenance cost. The other problem is requirements were not allowed to be changed which led to customer dissatisfaction. To overcome these problems, Incremental – Iterative models were developed in which the system was evolved or incremented over a period of time with customer involvement. There are four variations of this methodology: Prototype Model, Spiral Model, Rapid Application Development Model and Concurrent development.

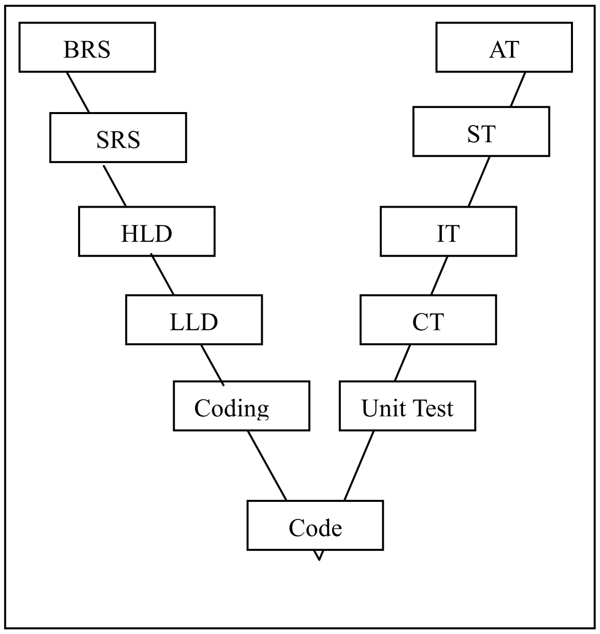


Figure 4: V- Model

Prototype Model

This model is widely used when the customers are unclear of their requirements. The initial requirements are gathered from the customers followed by a quick design. A prototype is developed and shown to the customer for evaluation. Once the customer is satisfied, full-fledged systems development will be done as shown in Figure 5.

The prototype developed is of two types – Throwaway and Evolutionary. If the prototype is developed only to get acceptance from customer and discarded further, then it is known as 'Throwaway Prototype'. The Prototype which will be further developed as the actual system is referred as 'Evolutionary Prototype'.

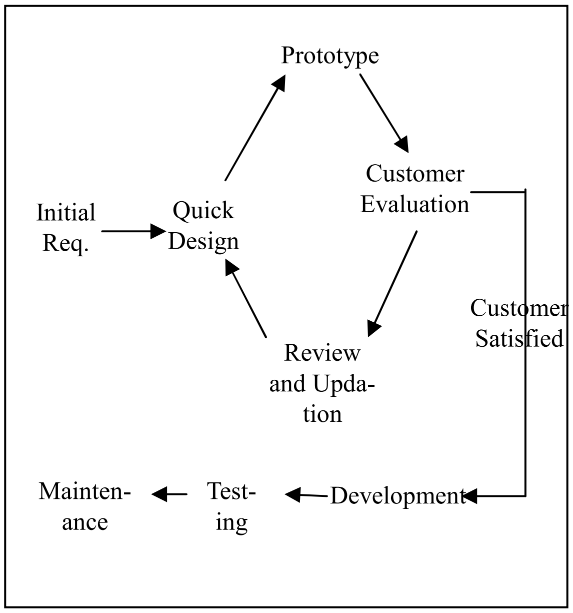


Figure 5: Prototype Model

Spiral Model

Spiral Model is a blend of traditional methodology and prototype model. It comprises of four quadrants as shown in Figure 6. The first quadrant "Planning" is the phase where requirements are gathered. This is followed by risk analysis where prototypes are developed to get acceptance from customers. Once the prototype is accepted, 'engineering and evaluation phase' continues where coding and testing happens. Once the incremental version of system is ready, it will be evaluated by customer. Next step again starts with planning and follows same sequence as explained above.

Spiral model emphasizes more on risk analysis and is used in projects which are prone to high risks like Defence, Aviation and Space related applications

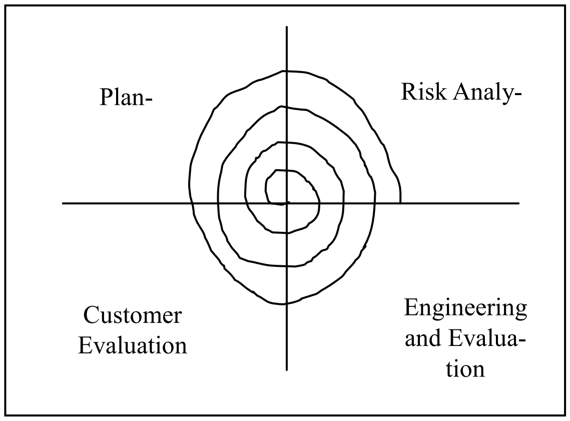


Figure 6: Spiral Model

RAD Model

Rapid Application Development Model developed in 1980s increases customer involvement, encourages development of prototypes and extensively employs computer aided software engineering (CASE) tools.

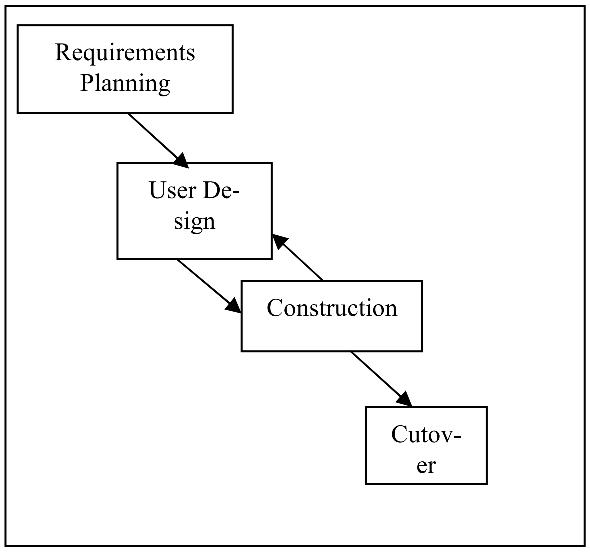


Figure 7: RAD Model

It consists of 4 stages as shown in Figure 7. The first stage 'requirement planning' is succeeded by 'design' and 'construction' which occurs iteratively till the user gets satisfied. 'Cutover' phase includes handing over the system. RAD model is suitable for applications which are to be developed very quickly.

Concurrent Development Model

Concurrent Engineering model defines a series of events that will trigger transitions from state to state for each of the software engineering activities, actions or tasks [4]. At a specific time, analysis, design, development and testing may be happening and the deliverables may be at different states which include 'under development, under review, awaiting changes' and so on. Concurrent engineering represents the overall picture of the current state of a project. Fig 12 shows the state of one of the Systems development activity.

Object Oriented Analysis and Design

OOAD methodologies combine data and processes into single entities called objects [1]. Objects correspond to real life entities and helps in improving the security and quality of systems been developed. OOAD works on the principles of abstraction, encapsulation, inheritance and polymorphism and advocates software reuse. OOAD is extensively employed in application development using object oriented programming languages like Java, C++. One of the variations of OOAD is Rational Unified Process (RUP) [1] which is explained below.

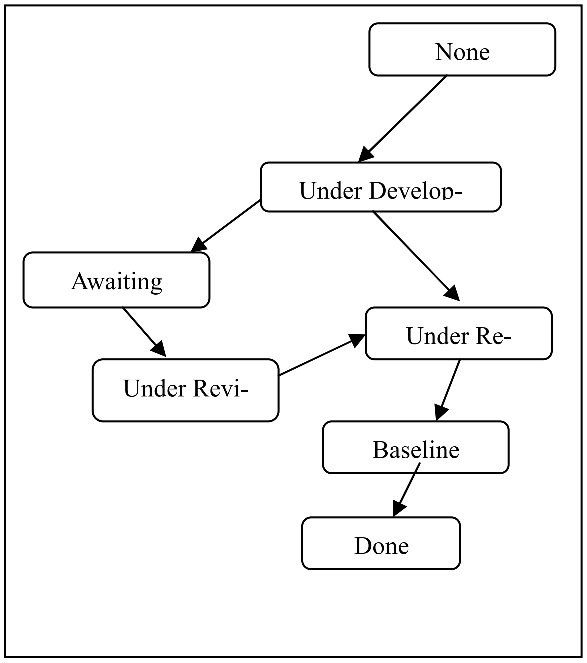


Figure 12: Concurrent Development

Rational Unified Process (RUP) Model

RUP model comprises of four phases as shown in Fig 8. During Inception, the scope of the project is decided and requirements are gathered. During the Elaboration phase, the requirements gathered are analyzed, prioritized and the architecture is developed. In the construction phase, coding and testing will happen with the beta version of the project deployed. This is followed by transition where user training is given and client approval is obtained. Each of the phases undergoes series of iterations till the next phase is reached. During the analysis phase, OOAD concepts are adopted which are further implemented in the construction phase.

Specialized Methods

Specialized models have narrow focus and are not widely adopted in all development projects. They adopt some of the characteristics of conventional models [5] but are applied to specific projects.

Aspect oriented Software Development

AOSD is based around abstractions called aspects, which implement system functionality that may be required at several different places in a program [33]. ‘Aspects’ are cross-cutting functionality that can be used in different parts of the application and woven to core functionality as and when required. This method helps in reuse of aspects without regard where the code is used. Figure 9 depicts the cross-cutting functionality.

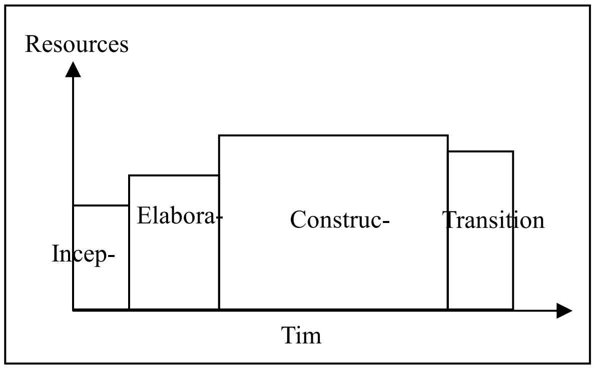


Figure 8: RUP Development

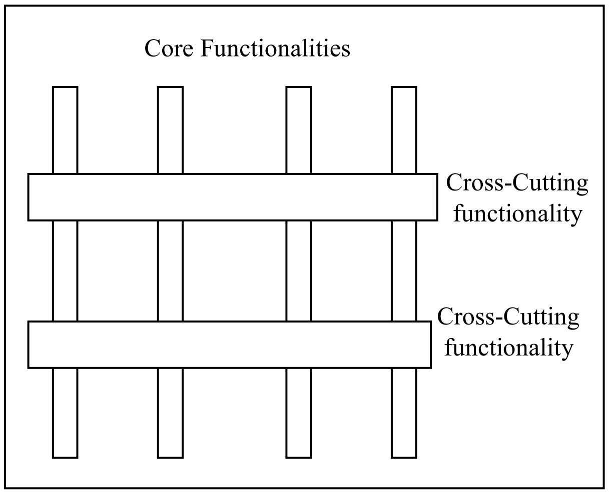


Figure 9: Aspect Oriented Development

Formal Methods

Formal methods or Cleanroom approach uses mathematical and statistical techniques to develop quality software [5]. Each step in the development employs precision of mathematics including specification of requirements, designing and testing. The ultimate result is improved reliability of the software developed.

Component based Development

Component based development focus on reuse of existing components. This method employs Commercial off the Shelf (COTS) software components which are readily available with required functionality and are suitable for integration with other software.

In component based development, initially, the architecture of the system is made ready. This is followed by searching for components in COTS. If the component that matches the requirement is available, then integration issues are considered followed by integration and testing of the entire system [4]. This method helps to improve productivity and reduce cycle time.

Joint Application Design and Development

Joint application design was first developed by Dan Gielan in 1974 and Joint application development was introduced and popularized by Chuck Morris and Tony Crawford in the late 1970s.

JAD is a team oriented approach that focuses on involving customers to understand the need of business and helps to develop a joint solution. Instead of identifying the requirements from stakeholders individually, JAD recommends facilitated workshop and partnerships to develop systems.

Though JAD is considered to be a development methodology, it is formalized for only the analysis and design phases of SDLC. In a typical JAD life cycle, four participants - Executive Sponsor, IT Representative, Scribe and user are involved. Scribe is responsible for documentation and act as facilitator to conduct JAD Sessions. JAD life cycle include the following phases: Definition, Preparation, Design and Finalization. Used effectively, JAD helps to accelerate design, enhance quality and reduces development cost.

SDM from 1993 to 2002

From 1993 to 2002, three development methodologies became popular and are being adopted by companies till today. The methodologies are Service Oriented Architecture, Agile Methodologies and Open Source Development.

Service Oriented Architecture (SOA)

Gartner defines ‘SOA as a software architecture that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations and interface calls [9]’. It is an approach of building a System by bundling various components providing generic functions. SOA advocates the concept of software reuse and promotes collaboration. It became very popular with the introduction of Web Services by Microsoft. The relationship between SOA and web services is highly influential [10]. The potential benefits of SOA include facilitation of rapid application development through service assembly, high ROI due to reuse of services and the ability to use legacy services through communication networks [1].

Agile methods

Agile is dynamic, content specific, aggressively change embracing and growth oriented [4].Agile manifesto held in 2001 defined the 12 principles to be followed to achieve agility. Typically agile development methodologies are adopted in organizations which are very dynamic and who build Agile teams.

An Agile team include the following characteristics – Common focus, decision making and fuzzy problem solving skills, competency and collaborative work environment, each and every member giving mutual respect and trust [4].

Unlike traditional methods, Agile methods appreciate changes, involves users throughout the entire life cycle, divides the development into shorter cycles of having 2-4 weeks, promotes the concept of early delivery, advocate continuous integration to improve quality and is the most highly adopted practice in industry. There are a number of variations available in agile methods. Among them, some are discussed here.

eXtreme Programming (XP)

XP started in 1996 is one of the highly successful agile development models. Extreme Programming emphasizes teamwork. Managers, customers, and developers are all equal partners in a collaborative team [11]. XP comprises of 4 phases as shown in Figure 10.

During the Planning phase, user stories are built with customer involvement in such a way, each story can be completed within 2 weeks. An Estimate is made and a schedule is prepared to develop the identified user stories and the team commits for development.

This is followed by design phase where 'Keep it Simple' philosophy is followed. XP encourages reuse by adopting 'Class – Responsibility Collaborator' concepts and whenever a risk is seen, prototypes (i.e., 'spike solutions') are built to get customer consensus. 'Refactoring' is another characteristic of XP.

During coding, unit test scripts are developed first followed by code generation. XP advocates 'Pair Programming' where both developers and testers work together in the same terminal. The developer concentrates on code design and tester concentrates on code standards. 'Continuous integration' is another aspect of XP. At the end of the cycle, an incremental version of software is released and 'project velocity' is computed. Project Velocity helps to understand whether the estimation was optimal.

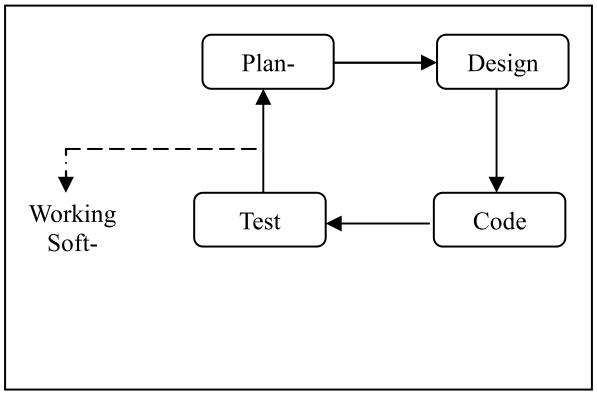


Figure 10: eXtreme Programming (XP)

Scrum

Scrum, developed in 1993 is an agile methodology employed for completing complex projects [12]. In scrum, the requirements of the system to be developed are termed as 'product backlog'. From the product backlog, requirements are prioritized for doing a 'Sprint' – a 30 days scrum cycle which ultimately delivers incremental software. The backlog of sprint is referred as 'Sprint backlog'. The requirements of sprint backlog is analyzed, designed, coded and tested by a sprint team as a collaborative effort. Every 24 hours, a scrum meeting coordinated by Scrum Master with the participation of entire team will happen as shown in Figure 11.

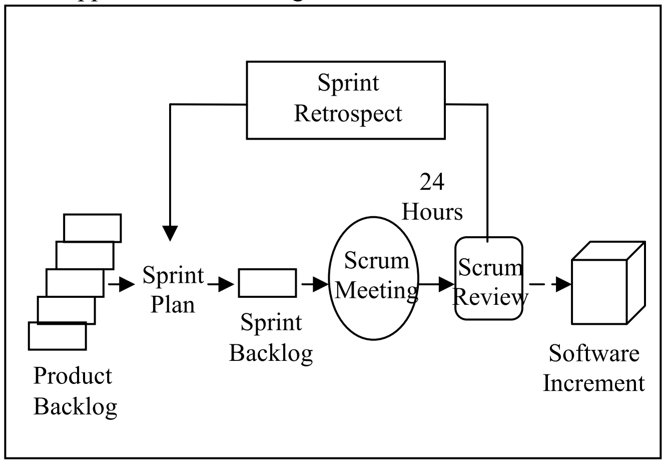


Figure 11: Scrum Framework

The purpose of the Scrum Meeting is to understand the achievements made since last scrum meeting, discuss and resolve any obstacles for the work to be accomplished and also get commitment of deliverables before next scrum meeting. It is a collaborative exercise and is a proven successful model.

Adaptive Software Development (ASD)

Adaptive Software development developed by Highsmith comprises of three phases – speculate, collaborate and learn as shown in Figure 12. During Speculation, project is initiated and basic requirements are defined [4]. Duringcollaboration, the requirements are analyzed, designed, coded and tested as a collaborative team and during learning, formal technical reviews are done, focused groups get feedback from customers and post-mortem is done with the intent of learning and improving. ASD emphasizes in the formulation of self organizing teams, interpersonal collaboration, individual and team learning

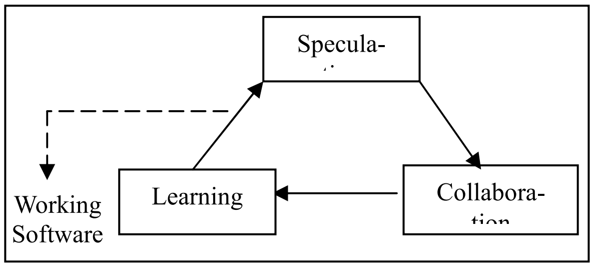


Figure 12: Adaptive Software Development

Feature Driven Development (FDD)

Feature driven development introduced in 1999 is a client-centric, architecture-centric, and pragmatic software process [13].

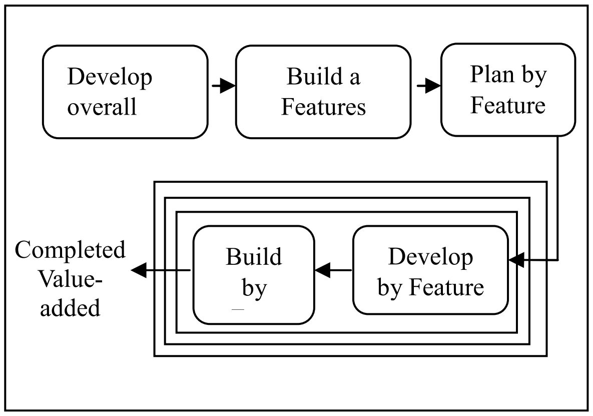


Figure 13: Feature Driven Development

In FDD, a 'feature' is a client valued function that can be implemented in two weeks or less'. It consists of 5 collaborative framework processes as shown in Figure 13. It emphasizes on project management and defines six milestones during the design and implementation of a feature – design walk-through, design, design inspection, code, and code inspection, promote to build [4].

Dynamic System Development Methodology

Developed in 1994, DSDM is an agile development methodology and adopts Pareto Principle such that 80% of project comes from 20% of requirements [14]. Follows the concept of MoSCoW for prioritizing requirements which stands for Must, Should, Could, Won't have requirements. DSDM suggests an iterative software process and consists of three iterative cycles and two life cycle activities. The iterative activities include functional model iteration, design and build iteration and Implementation. Life cycle activities include Feasibility Study and Business Study.

Crystal Methodologies

Developed in mid-1990s by Alistair Cockburn, Crystal methods are referred as 'lightweight methodologies' [14]. It focuses on people, interaction, skills, community, talents and communications. Processes are given secondary focus and people's interaction, talent are given a major focus.

The methodology uses colors to denote the 'weight' of methodology to use. The different colors in the family include Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Red, Crystal Maroon, Crystal Diamond and Crystal Sapphire. The larger a project gets, darker the color. The seven properties of crystal methodology include frequent delivery, reflective improvement, close communication, personal safety, focus, easy access to expert users and technical environment with case tools.

Open Source System Development

It is the process in which the source code of the developed software is publicly available for study, change and improvement. This new style of development became popular after Tim Berners-Lee made his HTML code as the platform for the development of World Wide Web. Typically an open source project will be initiated by anyone who senses that there is a need for Software to be developed. The initial code is shared with public and it is followed by identification of volunteers to chalk out the development plan.

Full-Fledged code development, review, code documentation, testing and code commitment is then made. The software is released and enters into continuous improvement.

There are different types of open source development projects. In Garden variety, standalone software programs are developed for a specific purpose. In Distribution project, the common source program is distributed which can further be customized. In BSD model, the software development will be done using one revision control system developed by a single team. Standalone documentation projects develop documentation for open source software which has already been developed.

SDM from 2002 onwards

From 2002 onwards, six development methodologies became popular and are being adopted by companies till today. These include Test Driven Development, Behavior Driven Development, Agile Unified Process, Kanban Software Development, Scrumban and Disciplined Agile Delivery.

Test Driven Development

Introduced in 2003 by Keny Beck of eXtreme programming, TDD advocates preparation of tests before code is written. It emphasizes specification more than validation which is the traditional focus of testing.

It comprises of four steps – add a test, run the test, make a little change in code to make the tests pass and again run the test to ensure the tests passes and again repeat the cycle. It focuses on refactoring - tuning the code without any changes in the appearance of the feature. In simple words, TDD is test first development added with refactoring.

Behavior Driven Development

Introduced in 2003 by Dan North, BDD is an extension of Test Driven Development. North defines BDD as "a second-generation, outside–in, pull-based, multiple-stakeholder, multiple-scale, high-automation, agile methodology". It describes a cycle of interactions with well-defined outputs, resulting in the delivery of working, tested software that matters. It works on the principle of getting the behavior of software from stakeholders and uses 'should' to describe the behaviour and 'ensure' for assigning the responsibility. It implements examples to describe the behaviour and uses automation to provide quick feedback and regression testing.

Agile Unified Process (AUP)

Developed by Scott Ambler, AUP is a simplified version of Rational Unified Process consisting of four phases and seven disciplines. AUP advocates small increments over 'big bang' approach by releasing the system in portions into production as shown in Figure 14.

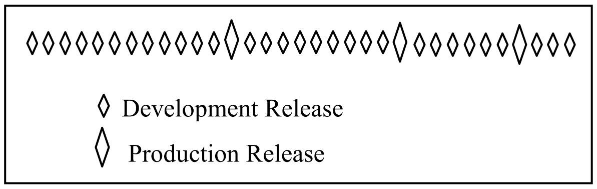


Figure 14: Agile Unified Process

The first production release may take twelve months, the second release may take nine months and subsequent releases take six months. Continuous learning, experience makes the system to be developed quickly. In AUP, phases are large and disciplines are iterative and small. AUP works on the philosophy of agility, simplicity, trust in staff, focused on high value items and can be tailored as per the need. AUP is not for everyone and should be chosen as per the requirement of the development team.

Kanban Software Development

Inspired by the Toyota Production System and Lean manufacturing, Kanban software development originated in 2004 is a visual process management system that aids decision making concerning what to produce? When to produce? and How much to produce? [16]. It works on four key practices: Visualize the workflow, Lead using a team approach, reduce the batch size of your efforts, learn and improve continuously. It uses a Kanban board for visualization and control mechanism.

Scrumban

Scrumban is a combination of Scrum methodology and Kanban methodology as shown in Fig 15. Scrumban is a Scrum or Scrum-like process which is being improved by Kanban [18]. Fundamentally, Scrumban is a management framework that emerges when teams employ Scrum as their chosen way of working and use the Kanban Method as a lens through which to view, understand and continuously improve how they work

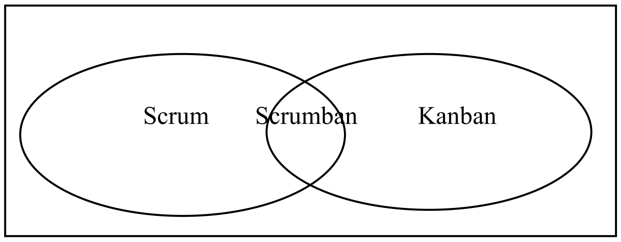


Figure 20: Scrumban

Disciplined Agile Delivery (DAD)

Scott Ambler started to work on DAD form 2009 and defines 'Disciplined Agile Delivery (DAD) process as a decision framework concerned with people-first, learning-oriented hybrid agile approach to IT solution delivery. It has a risk-value delivery lifecycle, is goal-driven, is enterprise aware, and is scalable DAD is a process decision framework for Lean Enterprises'. DAD is a hybrid approach of many development methodologies like Scrum, Kanban, XP, and RUP and so on. The focus of DAD is on delivery consisting of three phases which results in an incremental system. There are four versions of DAD which includes Agile/basic version, Advanced/Lean Version, Lean continuous delivery cycle and an Exploratory Lean startup cycle. The development team can choose any of the versions and tailor it as per the requirement. DAD advocates the philosophy of goal-driven, enterprise aware team and scaling Agile.

OPEN-SOURCE DEVELOPMENT VS CLOSED-SOURCE DEVELOPMENT

Eric Steve Raymond in his work, "The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary" discusses on the two styles of development – Conventional Closed Source development and Progressive Open Source development.

Raymond compares closed source development to the building of a cathedral where the entire cathedral is built before the doors are open and he says that the software is developed over the internet through crowd sourcing in open source development,

In Cathedral model, the source code is only available during release of the software and it is available with only exclusive group of developers during development. But in case of Bazaar model, the code is available to the public during development itself.

He emphasizes that software developed through bazaar model is more reliable as the source code is available for public testing and scrutiny. In case of open source development, the source code is available only for selected developers in the organization and so it may not be reliable to the extent of the software developed using open source development model.

Raymond lists 19 lessons to create open source software in his work.

Open Source development involves volunteers and advocates collaborative work. The following lists the characteristics of open source development:

- The Source code is available for public to view, modify and distribute
- The community of developers are volunteers and are spread across the globe
- The focus is collaborative development of software to improve the quality and reliability of software
- Release early and release often

The following table provides the difference between Open Source development & Closed Source Development:

Criteria	**Open Source Development**	**Closed Source Development**
Software Access	Freely available	Using License
Software Acceptance	In the growth phase of acceptance	Accepted by all
Philosophy	Improve Quality of software through collaborative open community	Holding intellectual copyrights within the company
Motivation	Contribution to Society	Commercial business
Evolution of Style	1998 onwards	1960s onwards
Market Focus	Wider Market	Narrow Market
Business Model	Revenue through Support Services	Revenue through software licenses
Source code View	Can be Viewed	Can be Viewed
Source code Modification	Can be modified	Trade secret, cannot be modified by public
Development community	Across globe	Company specific developers
Software	Allowed	Not Allowed

Distribution		
Vendor Lock-in	No	Yes
Bug fix	Based on open source community	As per SLA
Security	Better than Closed Source	Secured
Software Stability	Stable	Better than Open Source
Customization	Highly Possible	Possible with request to vendor
General Preference	Small and Medium Enterprises	Large Enterprises
Documentation	Online documentation	Guides & Help files along with software
Ease of use	Good	Good
Support	Available through online blogs	Tailored to the requirement
Cost	Free	Expensive
Risk	Moderate	Less risk

Today, Open source development is a proven style of systems development. The Open source development era started in 1989 when World Wide Web was born and Tim Berners-Lee gave his HTML code for development to the public. This was followed by Linus Torvalds bringing this style of development to the world through his Linux Operating system in 1991. Linux is widely accepted, stable and highly popular open source operating system used worldwide today.

The next stage of recognition to open source development came when Raymond inaugurated the open source community in 1998 and Netscape Communication Corporation announced starting its open source Mozilla project. This is the period when Apache group's open source web server became the market leader grabbing 65% of market share in 1999. Even today, apache runs several open source projects successfully catering to different aspects of software development process.

Another major breakthrough of open source development is given by Google. Google's open source mobile operating system, Android started in 2003 is the market leader in mobile market. It is the highly reliable and adopted mobile operating system today. In the browser world, Google Chrome, the open source browser, is the widely used browser and is the market leader. Started in 2008, Chrome, the open source development browser from Google has reached the number one position out beating Microsoft's Internet Explorer.

Today, most of the companies have started adopting open source development as a supplement to their closed source development to improve the reliability of the systems developed and to employ crowd sourcing. They have also started adopting open source software tools to aid in their development efforts. With development community spread across the globe, to grab a wider market and to improve collaborative systems development, open source development style is embraced as a supplement to closed source development.

CONCLUSION

This paper gave an overview of all the system development methodologies that have evolved till date. It attempted to distinguish the two major styles of development – closed source, open source and also highlighted the characteristics of open source development. The change in landscape from closed source development to open source development was discussed. It also cited several examples of systems developed adopting open source development style. Today open source development is being adopted as supplement to closed source development is also explained.

REFERENCES

Chen W.K., 1993. Linear Networks and Systems. Belmont, Calif.: Wadsworth, pp. 123-135. (Book style)

Poor H., 1986. "A Hypertext History of Multiuser Dimensions," MUD History, http://www.ccs.neu.edu/home/pb/mud-history.html.

Coming D.S. and Staadt O.G., 2008. "Velocity-Aligned Discrete Oriented Polytopes for Dynamic Collision Detection," IEEE Trans. Visualization and Computer Graphics, **14**(1):1-12, doi:10.1109/TVCG.2007.70405. (IEEE Transactions)

Williams J., 1993. "Narrow-Band Analyzer," PhD dissertation, Dept. of Electrical Eng., Harvard Univ., Cambridge, Mass. (Thesis or dissertation)

Hubert L. and Arabie P., 1985. "Comparing Partitions," J. Classification, **2**(4):193-218.

Vidmar R.J.,1992. "On the Use of Atmospheric Plasmas as Electromagnetic Reflectors," IEEE Trans. Plasma Science, **21**(3):876-880, available at http://www.halcyon.com/pub/ journals/ 21ps03-vidmar.