Generalizing Agile Software Development Life Cycle

S. Bhalerao[1], D. Puntambekar[2]
Master of Computer Applications
Acropolis Institute of Technology and research
Indore, India
[1]Bhalerao.shilpa@gmail.com,
[2]d_puntamberkar@rediffmail.com

M.Ingle
School of computer Science and Information Technology
Devi Ahilya University
Indore, India
maya_ingle@rediffmail.com

Abstract— **In last decade, various agile methods have been introduced and used by software industry. It has been observed that many practitioners are using hybrid of agile methods and traditional methods. The knowledge of agile software development process about the theoretical grounds, applicability in large development settings and connections to establish software engineering disciplines remain mostly in dark. It has been reported that it is difficult for average manager to implement agile method in the organization. Further, every agile method has its own development cycle that brings technological, managerial and environmental changes in organization. A proper roadmap of agile software development in the form of agile software development life cycle can be developed to address the aforesaid issues of agile software development process. Thus, there is strong need of agile software development life cycle that clearly defines the phases included in any agile method and also describes the artifacts of each phase. This generalization of agile software development life cycle provides the guideline for average developers about usability, suitability, applicability of agile methods.**

Keywords-Agile software Development; extreme Programming; Adaptive software developmen; Scrum; Agile Method;story.

I. INTRODUCTION

Agile Methods (AMs) have been adopted by many IT organizations and have generated many quality products of software industry. These methods have gained higher edge on traditional software development by accommodating frequently changing requirements in high tight schedules [1]. AMs have promised higher customer satisfaction, low defect rates, higher usability and a solution to higher changing requirements [2]. AMs include mainly; Extreme Programming (XP), Scrum, Feature Driven Development (FDD), Crystal methodology, Dynamic System Driven Development (DSDM), Adaptive Software Development (ASD), Open Source (OS), Agile Modeling (AM), and Pragmatic Programming (PP) [3]. It has been observed that all aforesaid methods are based on agile manifesto and have their own software development life cycle for improving productivity and quality of software [4]. It has been noticed that applicability of these methods is mainly in small software with low life critical systems. Many opponents have claimed that agile software development is set of ad-hoc practices and does not have sound principles behind it. Further, it has been stated by many software researchers that it is hard for average software developer/ manager to understand and manage entire agile approach to development [5]. Attempts have been made to reconcile the AMs with plan driven methods [6]. Still, there is lack of a generalized Agile Software Development Life Cycle (ASDLC) for AMs that include complete agile principles and practices as whole. Therefore, in this paper, we have proposed ASDLC and also discuss the documents or artifacts required to produce in particular phase. It is highly beneficial to identify the activities and practices associated with particular phase of software development. Knowledge of ASDLC is also useful to reduce the ratio of experienced member and inexperienced members in team. This will be highly useful for generating trust in industry about Agile Software Development Process (ASDP).

In this paper, firstly, we will discuss ASDP and research in this area with their pros and cons in Section 2. Secondly, we will propose the generalized ASDLC in Section 3. Section 3 also includes the activities and document produced in various phases. Lastly, conclusion and future scope is drawn in ASDLC in Section 4.

II. BACKGROUND

Many software development methods/ models have been proposed since the evolution of software. Some development models had shown remarkable success in stable and predictable environment. At the same time, these models have proven to be one of the major causes of failure in disruptive software development. In internet and mobile technology, frequent changes in requirements, technology and staff have been observed [7]. Thus, software development process has become more cumbersome in such environment. Traditional Software Development Methods (TSDMs) are proven to be unsuccessful and software success rate of TSDMs is less than 40% in such environments [8]. A new way of software development i.e. agile software development is outcome of the frustration of many practitioners using TSDMs. In last decade, a number of AMs have been evolved based on Agile Manifesto established in 2001[www.agilemanifesto.org]. It has been observed that agile principles and practices ensure the customer satisfaction by involving the customer in all the phases of software development. It emphasizes mainly; accommodating last minute changes, delivering working software, individual interactions etc.

ISSN : 0975-3397

TABLE I. SUMMARY OF SOFTWARE DEVELOPMENT LIFE CYCLE OF POPULAR AGILE METHODS

Sr. no	Method	Phases	Description
1	XP	Exploration	Write story for current iteration
		Iteration Panning	Prioritize Stories, effort and resource estimates
		Iteration to release	Analysis, design, coding, testing
		Production	Rigors testing,
		Maintenance	Customer supports, release for customer use
		Death Phase	No more requirements
2	Scrum	Pre-game	Preparation of product backlog list ,effort assessment, high level architectural design
		Development	Sprints, analysis, design, delivery,
		Post game	System testing, integration testing, documentation releases
3	FDD	Develop over all model	Scope, features, model, use cases are decided in various iterations
		Build the feature list	Feature list is prepared
		Plan by feature	Not clearly specified
		Design by feature	Not clearly specified
		Build by feature	
4	DSDM	Feasibility Study	Feasibility of the system is assessed
		Business Study	Essential business and technology characters are analyzed
		Functional model iteration	Analysis, functionality prioritization, nonfunctional requirements and risk assessment.
		Design and build iteration	Build and testing of system
		Implementation	Actual production of the system
5	ASD	Speculate	Project initiation, adaptive cycle planning
		Collaborate	Concurrent component eng.
		Learn	Review, F/A, Release

AMs are people centric and believe in short iterations and small releases to get feedback on the working software. This feedback is useful in improving the quality of the software. It has been noticed that each AM has individual software development life cycle and characteristics. For example, XP possesses five phases namely; exploration, iteration plan, iteration to release, production phase and death phase. On other hand, DSDM and other methods follow different phases of life cycle. XP emphasizes on customer involvement in every activity of software development and lacking in management practices whereas Scrum mainly deals with the project management activities [2]. Although, all these methods use perform analysis, design, coding, and implementation in iterative and incremental manner. Table 1 represents the popular AMs with phases and details. It is clear from the Table 1 that DSDM not only stresses on development but also includes the feasibility and business study. Further, it has been noticed by many researchers that AMs do not follow all the phases of software development life cycle [3]. Some researchers have attempted to include missing phases of SDLC in existing AMs [9]. However, there is strong need to define generalized agile software development life cycle to increase the understandability of agile practices and principle to increase the use of these methods.

III. AGILE SOFTWARE DEVELOPMENT LIFE CYCLLE (ASDLC)

Proposed generalized Agile Software Development Life Cycle (ASDLC) is designed on the basis of common practices and principles used in all existing AMs. We have defined various phases in ASDP and activities performed in each phase along with artifacts required in each phase. Complete ASDLC is shown in Fig. 1 and discussed as follows:

A. Vision and Project Approval

ASDLC starts with the vision or inception phase that deals with the need of new system by analyzing problems in existing system. Management, product manager, users and team members establish the scope and boundary conditions of proposed system. At this level, objective is apparent but the features fulfilling the objectives may be uncertain. Main objective of this phase is to identify critical uses of the system, level of uncertainty of the system, overall estimation of size and duration of the system using algorithmic or non-algorithmic approach. Further, systematic analysis is performed to identify the feasibility of the system at operational and economical level with clear specified requirements. It is concerned with technical possibility of the system with incurring risk associated with it. At same level, feasibility of particular AM is assessed. This assessment is based on project type, and personnel and organizational issues etc. Business study of the system is required to analyze the essential characteristics of the business and technology. For example, a website for income tax submission must require its technicalities involved in it. Major objective of business study is identification of class of affected users. This affected class of users is useful source of information in software development cycle. It has been noticed that early estimation is useful in project approval.

It is a non iterative phase and generally completed in two-three weeks time. High level description of the system, early estimates are mandatory documents produced in this phase.

B. Exploration Phase

Exploration phase is an iterative and incremental phase to reduce the uncertainty and ambiguities in requirements by continuous meeting of stakeholders in the form of workshops and brainstorming. Some of the AMs have preferred customer as team member but proposed ASDLC recommends the maximum communication between team and customer to resolve the requirement related issues by using any preferred mode of communication between customer and team [9]. Requirements may be captured in form of stories and documented in story cards that can be referred for future references. Typical format of story cards contains information about author, story id, story description, further changes in story and details of related stories etc. [9]. Artifacts produced are informal requirements description in the form of stories. Team starts with selected experienced team members on agile software development. Selected team members start communicating with the customers to understand the problems and requirements of the proposed system. Generally, while experienced team members are working on requirements,

ISSN : 0975-3397

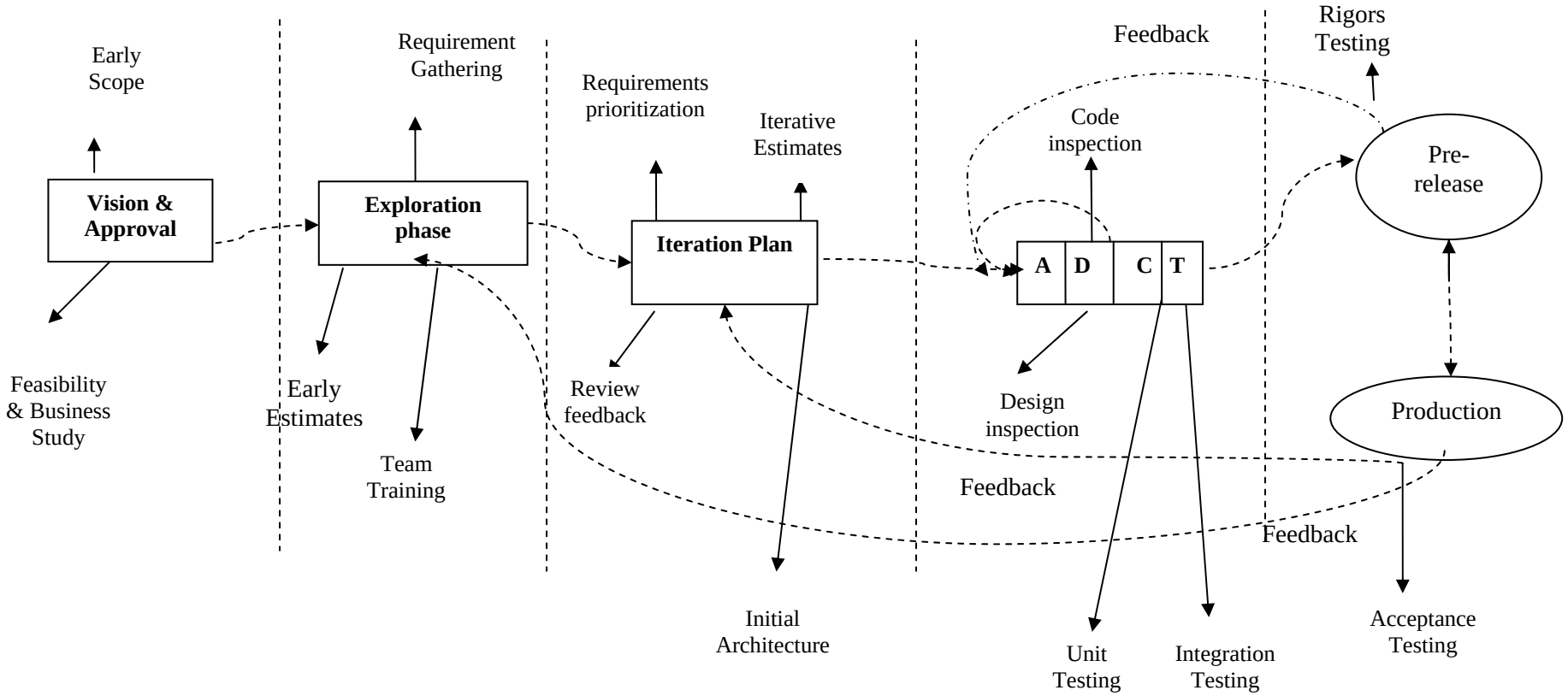

Fig. 1 Agile Software Development Life Cycle

ISSN : 0975-3397

inexperienced team members have been trained on agile process and technology used for training and enhance the ways to improve quality of product being developed. Further, feedback of the last release is also accommodated in this phase and major changes in the last releases are defined as new requirements.

C. Iteration Planning

Iteration planning is most important phase of ASDLC and possesses many activities of software development required to schedule the project. First activity in this respect is review of the working software released in last iteration. Participants assess the progress and increment of the work product and discuss the future plan of the project. At the same time, requirement prioritization is performed to get maximum ROI from working software. In iteration planning, list of requirements in stack is updated depending on the feedback and requirements received from customers. This list is reviewed for prioritization of requirements. Prioritization is based on various factors mainly; value, knowledge, financial returns etc. For example, a feature that requires team to improve their technical skills has been developed in later stage but a feature that has higher financial returns must be kept at higher priority. This prioritization stack is useful in increasing ROI and producing working software in shorter time period. Prioritization has been done for only those features that are clear and unambiguous. Project manager, customer representative and team members sit together to decide the priority of requirements. Moreover, iteration plan phase possesses iterative estimation activity to estimate size, cost and duration of the project. It also re-estimates efforts depending on team velocity [13].

This phase also ensures the resource requirements of the system. Artifacts produced in this phase are prioritized stack requirements and set of requirements from the stack is selected for current iteration.

C. ADCT Phase

This phase is an iterative phase that deals with Analysis, Design, Coding, and Testing (ADCT). In this phase, functionality of the system is produced and enhanced in new increments. It requires several iterations before releasing the product. Decided schedule in iteration planning is decomposed in several small iterations of one to four weeks. First iteration develops the architecture of whole system by enforcing the selection of stories that form the system. In successive iterations, designing and coding along with testing is performed. In last iteration, product is ready to deploy at customer site. It incorporates designing and coding with unit testing using the concept of pair programming. ASDP always possesses simple design to incorporate changes in the requirements. Design guidelines include metaphors, CRC cards, Spike solution and re-factoring. CRC card is an index card that is used to represent responsibilities, relations of classes used in designing a particular story. Spike solution is small focused effort used to explore solution to the problem. It has been observed that adding more functionality in early stages of the software leads to a poor design document. Any one of aforesaid practices leads to just enough, simple and understandable documents. System has been inherently designed to change. For example, system can work for any database. This type of independency of code and design provide lesser burden when changes are triggered. Thus, ASDP use design patterns to maintain low coupling and high cohesion among modules. Functionality testing and rigorous integration testing is performed by team of customer and developers before release the product. Main activities of this phase are simple designing, maintaining coding standards and rigors testing by Test Driven Development (TDD) and functional testing. Extra care is taken to design a code simply by code and data re-factoring.

Major artifacts in this phase are design documents and codes of system.

D. Release Phase

This phase can be decomposed in two sub-phases namely; pre-release and production as shown in Fig 1. Pre-release phase recommends extra testing (i.e. integration and acceptance testing) and checking of functional and non-functional requirements of the system to be released. It has been advised to include some minor changes expected by the user in the release and major changes are expected to accommodate in next iteration. On the other hand, production phase deals with releasing the product for customer use. At this time, training for users of the system is provided for operation ease. It has been observed that team handles two responsibilities after first release of the system. Firstly, team is involved in enhancing the functionalities of product. Secondly, team has to take responsibility of system in running state thereby providing customer helpdesk.

We have attempted to define the ASDLC after reviewing the all phases of software development of existing AMs. We have also included the phases introduced by other researchers thereby increasing the trust and faith on agile software development.

IV. CONCLUSION

ASDP is process to handle the disruptive software environment by incorporating practices and principles established in 2001. It has been observed that agile practices such as delivering working software, short iterations and feedback etc. increase the internal and external software quality. Some practitioners stated that agile practices are collection of best practices of the software development. Although, there are many success stories of ASDP in last decade, but knowledge of implementing these practices in a particular project is very scared. Therefore, we have analyzed software development life cycle of all existing AMs and proposed a generalized ASDLC. Proposed ASDLC is essence of all existing AMs and represents all phases required in a software development cycle in iterative and incremental manner. It also encourages the practices of simple design, re-factoring to maintain the simplicity. Thus, proposed work is useful for adoption of AMs with following benefits:

ISSN : 0975-3397

- ASDLC is a step towards resolving the misconceptions about AMs that these methods are ad-hoc coding practices.
- A systematic approach to define all the phases of ASDP that is useful to average project manager to understand the principles and practices behind agility.
- ASDLC represents the activities and document required in each phase thereby providing the developer and user view for better understanding of AMs.
- ASDLC provides flexibility to handle phases in concurrent and iterative manner.
- Feedback in ADCT phase improves internal quality whereas feedback in iteration planning improves external quality of the product.

Thus, proposed ASDLC is a step towards improving agile software development which will leads to fast accessibility of AMs. However, this is preliminary work and needs verification on large projects.

REFERENCES

[1] Aoyamma, M., "Agile Software Process and its Experiences", In *IEEE Transaction 1999.*

[2]Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J., *Agile Software Methods Rreview and Analysis*, Espoo, Finland: Technical Research Centre of Finland, VTT publication 478 available http://www.inf.vtt.fi/pdf/publications/2002/478.pdf.2002

[3] Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J., "New Directions on Agile Methods : A Comparative Analysis", In Proceeding of 25th *International conference on Software engineering 2003.*

[4] Beck, K., *Extreme Programming Explained*, Pearson Education Low price Edition Asia.

[5] Bergin, J., and Grossman, F., "Extreme Construction: Making Agile Accessible", In *Proceedings* of *IEEE Agile 2006 Conference.*

[6] Bhalerao, S., and Ingle, M., "Formalizing Communication Channel in Agile Methods", *In Proceedings of International Conference on Trends in Information Science and Computing (TISC07)*, Dec. 2007.

[7] Bhalerao, S., and Ingle, M., "Mapping SDLC phase with Various Agile Methods", In *Proceedings of International conference on Advances in Computer Vision and information Technology*, Nov. Aurangabad, pp. 318-325.

[8] Cohn, M., and Ford, D., "Introducing an Agile Process to an Organization", In *IEEE Computer Society 2003*, pp.74-78.

[9] Cockburn, A., and Highsmith, J., "Agile Software Development: The People Factor", In *Computer, Nov. 2001*, pp. 131-133.

[10]Cockburn, A., *Agile Software Development*, Pearson Education, Asia, Low Price Edition.

[11] Dingsøyr, T., Dybå, T., and Abrahamsson, P., "A Preliminary Roadmap for Empirical Research on Agile Software Development", In *Proceedings of Agile Conference 2008*

[12] Pressman, R., *Software Engineering A Practitioner Guide*, McGraw- Hill 6th Edition.

[13] Bhalerao, S., and Ingle, M., A Comparative Study of Agile Projects Estimation using CAEA, In *Proceedings of International Conference on Computer Engineering and Application, June 2009, Philippines.*

[14] Cao L. and Balasubramanium R., " Agile Software Development: Ad-hoc Practices or Sound Principles ?", IEEE ITPRO, March- April 2007, pp. 41-46.

[15] Iacovelli A. and Souveyer C., "Framework for Agile Method Classification", *Proceedings of Model Driven Information System Driven Engineering- Enterprise, User and System Model (MoDISE –EUS) 2008, pp. 91-102.*

AUTHORS PROFILE

Prof. S. Bhalerao is MCA from Banasthali Viyapeeth, Rajasthan and persuing her research in standardization of agile methods to increase trust and faith amongst practitioners under the guidance of Dr. M. Ingle Professor, DAU, India. She has published more than 15 papers in national and international confereces and journals. She got IT excellence award for improving education standards in central India

Prof. D. Puntambekar is MCA from DAVV, Indore with about 20 years of Industrial and Academic experience. He has published more than 25 research papers in national and internation Conferences. He is also consultant to many Govt. agencies for E-Governance implementation. He is being appointed as Director on the board of M.P. State Electronics Development Corporation (MPSEDC) a State Govt. Enterprise.

Prof. M. Ingle is Ph. D in Computer Science. She is renowned Professor and Senior System Analyst in a prestigious Devi Ahilya University (DAU), Indore, India. She has published more 70 research papers in national and international confereces and journal in area of software engineering, Web Engineering, NLP and Usability Engineering etc. She has been awarded for best teacher and imporving education standards in field of IT in central India.

ISSN : 0975-3397