A Documented Approach in Agile Software Development

Nitin Uikey *nitin_uikey@yahoo.com*
School of Computer Science
Devi Ahilya University
Indore, 452017,India

Ugrasen Suman *ugrasen123@yahoo.com*
School of Computer Science
Devi Ahilya University
Indore, 452017,India

A.K. Ramani *ramaniak@yahoo.com*
School of Computer Science
Devi Ahilya University
Indore, 452017,India

Abstract

Software engineers have been striving for years to improve the practice of software development and maintenance. Presently the agile approach is quickly becoming main stream in the software industry with core set of beliefs and practices called Manifesto for Agile Software Development. However, agile methods have shaken the view, arguing that more emphasis should be given on software development rather than extensive documentation. It has been observed through literature that good documentation plays a very important role in software development. It helps in increasing development speed and also helps in communication and maintaining relationship among developers and other stakeholders. In this paper, a conceptual view of documentation is proposed and a technical writer is introduced along with other scrum roles. Also, the paper presents the relationship of documentation and technical writer by which we can enhance the productivity and maintainability of software.

Keywords: Agile Methodologies, Scrum, Technical writer, Extreme Programming, Software Documentation.

1. INTRODUCTION

Agile software development methods have been adopted by many IT organizations and seem quite suitable for today's quick paced and frequently changing environment [1]. One of the main reasons for using agile methodologies is the capability of keeping overtime and expenses in check while trying to satisfy the needs of the users [1]. The various agile methodologies adopted in recent years are XP, SCRUM, ASD, TDD etc., all based on beliefs and practices defined by Manifesto for Agile Software Development (www.agilemanifesto.org) [2].

Among all agile methodologies the most widely used practices are XP and SCRUM. XP is based on planning game and the small releases coding guidelines [3][6], while maintaining the iterative and rapid feedback driver nature [4]. SCRUM approach has been developed for managing the software development process in a volatile environment based on flexibility, adaptability and productivity [4][5].

Evaluating SCRUM process and agile manifesto, which states working software over comprehensive documentation (agilemanifesto.org), there lies an importance of documentation for current and future benefits as well [7]. SCRUM consists of sprint planning, sprint meeting, SCRUM execution, product backlog, sprint burn down chart etc., but to keep a track of the path

and record of the work performed, there is a need of documentation which is currently missing in practice.

Consider a situation, where one of the team unavailable member is replaced by a new member. The new member can start working on the item but is unable to relate with past work and their connection with other activities. The scrum master can only keep an eye but now other team members have to devote some more time explaining the new member about work been performed till now to make him understand the system well [8][1].

Another aspect of SCRUM is the feedback from previous experiences. SCRUM is lacking a phase, where current experiences can be used in next similar projects. There is no record maintained about the lessons learnt or experiences gained from the current project, which can be useful as feedback in next projects. Sprint retrospective meeting determines well performed activities and problems faced during the sprint [9]. Presently, there is no documented record maintained to be used in future.

Analyzing other phases such as requirement gathering and developing an organization wide design, as performed in traditional software development life cycle, helps the team members to broadly gain knowledge about the system. SCRUM is incapable to support its team with the view of system thinking, which considers to view system as a whole and no such independent entities exist in a system.

Documenting or recording procedures and structures is necessary to increase the significance of the work. Ambler suggests two primary reasons for documentation; one is communication and other is understanding [10] [11]. The purpose of documentation is well understood as it helps to instruct those who are unfamiliar with the system and its organization and its work process [12]. Uncertainty in software environment is one of the major problems faced by the industry and its people. As uncertainty influences the managers to plan and proceed, to develop the product with its boundaries and constraints and above all, uncertainty influences the success of the project [13].

Another important aspect is requirement engineering, which is concerned with identifying, modeling, communicating and documenting the requirements of a system. It describes the final product to be produced and helps to create a vision of the outcome before actual development of the system begins [8].

Similarly, if documentation is written correctly, completely and consistently, it is regarded as a powerful tool to gain project success. It also helps software engineers to thoroughly understand a system, its components and their relationships [7]. However, the studies reveal that out of date software documentation still remains useful in many circumstances, such as inline comments and historical guidance and they are often good enough to greatly assist in detailed maintenance work [14].

One characteristic of agile methods is the informal communication among stakeholders and developers, which sometimes raises a problem of communication breakdown such as inability to scale the software, adding a new member into the development team, coping with systems complexity and rapidly changing requirements [1]. In such cases, documentation can really help to break the communication gap among stakeholders and other team members. Also, an effective communication can help understand the requirements and gain feedback from the customers. This helps in maintaining a relationship between customers and developing team which can then help in reducing further cost and time of development [15]. Finally, documentation is the best option when communicating information during development and recording information for permanent and future use [10].

Evaluating these facts, a new model of documentation, which can be incorporated with SCRUM is proposed. At the same time, a new role of technical writer is also introduced along with Product

Owner, Software Development Team and Scrum Master, who will help in documenting all the desired information for the betterment of the project as well as the SCRUM process. The new model will help reducing communication gap between customers and development team. Also it may help in recording the information and experiences for future use and in maintenance of the system.

In this paper, we have discussed a proposed documentation process in SCRUM based products and its components in Section 2. Section 3 describes pros and cons of documented approach used in software environment. Lastly, conclusion and future work is described in Section 4.

2. DOCUMENTED APPROACH IN SCRUM BASED PRODUCTS

An efficient documentation is a major factor to understand the requirement and feedback from the stakeholders. Also, it plays an essential role in communicating between developer, managers and customers. In this paper, we have proposed a documentation process in SCRUM based products, which consists of contract document, requirement specification document, design document, SCRUM process, lessons learnt document, document management system and technical writer. The components of the proposed documentation process are discussed in the following sub-sections.

Hereby, the new model will not only help developer to understand the system completely but will also help future similar projects to achieve the desired outcome. At the same time, a new role of technical writer is also proposed who may help to achieve the goal by documenting all the desired information and help the team in better communication. The conceptual view of documentation process with its sub-components is shown in Fig.1.

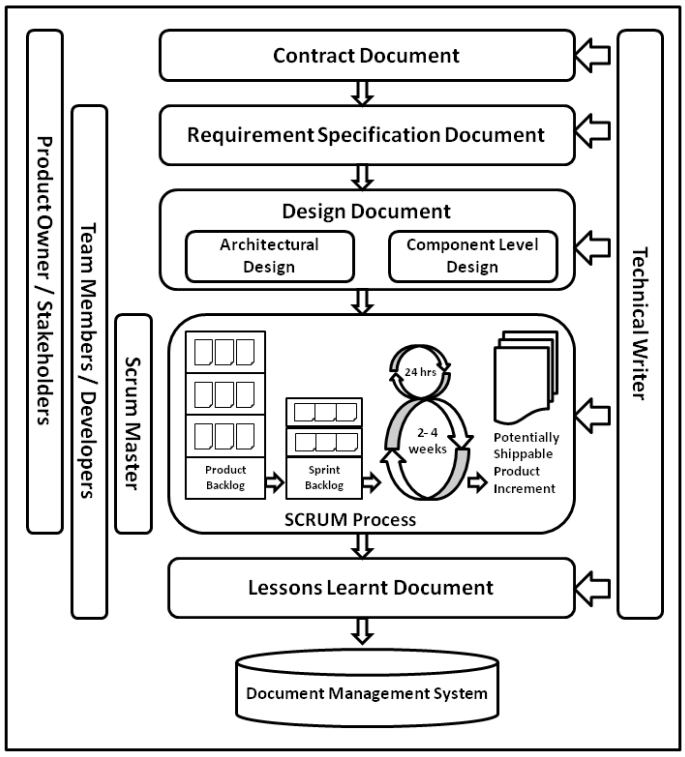

FIGURE 1: Documentation process in SCRUM based products.

2.1. Contract Document

The documentation starts with the initial phase, i.e. contract document, in precedence to project selection. Contract is a kind of agreement between two or more parties, especially one that is written and enforceable by law [30]. Contract documents help to identify the existence of the project formally along with a proper sign off of top management, staff and project team. Contract document is also helpful in maintaining relationship and creating an understanding between the project team and staff involved at various levels to show their commitment, information flow and communication levels.

In the proposed documentation process, the contract document can be created by technical writer in coordination with other team members, stakeholders and executives, to formally show the existence of the project and add to organization's portfolio.

2.2. Requirement Specification Document

After the contract sign off and identifying the stakeholders, the next phase is to collect, analyze and specify the requirements. Software Requirement Specification (SRS) forms the basis for agreement between the customer and the developer on what software will be produced. This will aid in creating a mutual understanding and clearly defining what the customers expect from the software. Also, developers clearly understand what features are to be added in order to satisfy the customer. SRS is treated as a reference for validating the final product as per the requirement.

SRS should be prepared by technical writers with assistance of product owner and development team. This will make easier for technical writers and developers to understand the system from scratch and will be useful to create good requirement documentation.

2.3. Design Document

Software design encompasses the set of principles, concepts and practices that leads to the development of a high quality system or product. A design document is a written document of a software product that provides an overall guidance of the software project architecture. A design document is intended to provide coordination among team members under a single vision. It is a stable reference, an outline view of the system and shows subsystems interrelations. The document is a complete description and maintains a high level view of the software [16]. The purpose of the design document is to express the vision of the system, describing its content and presenting a plan for implementation. Thus, design document helps to visualize in detail about the architecture, data structure, interfaces and components that are necessary to implement the system.

The purpose of architecture design is to gain a general understanding of the system decomposition and relationship with other subsystem to provide a desired functionality [17]. Today the complexity of software has increased, making it difficult to grasp and develop the whole system. The number of control paths, references, variables makes the understanding almost impossible. Thus, it is necessary to subdivide the system into manageable parts so as to increase the understanding and making things easy.

A complete set of software components are defined in architecture design. Component level design establishes the algorithmic detail required to manipulate data structures, define communication mechanism between software components via interfaces and implement the processing algorithms allocated to each component. This helps in determining whether the software will work before building it. The component level design represents the software in such a way that allows reviewing the earlier design representation for correctness and consistency [18].

Technical writers can assist developers to create good design document which can further be used as reference in scrum process. The design document will facilitate the teams to better inter-

relate the subsystems and visualize the system thinking approach while executing scrum process.

2.4. SCRUM process

Scrum is an iterative incremental methodology for software development. It is a management framework for incremental product development using one or more cross-functional, self-organizing team consisting of product backlog, sprints, reviews and burn down charts. Scrum provides a structure of team roles, meetings, rules and artifacts. Teams are responsible for creating and adapting their processes within this framework.

For any product to be built, all the features are collected from users, customers, executives and other stakeholders often written in User Story form called the product backlog. The team has certain amount of time called sprint to complete a work with prioritized sprint backlog in hand. The team plans out several sprints to complete the work. Sprints are short duration milestones that allow team to tackle and manage the chunk of projects and get it to a ship ready state. Sprints generally range from couple of days to as long as 30 days in length depending on the product release cycles. Scrum components insist on short daily standing meetings. Meeting daily, the team feels confident that everyone is on the top of their task. Team member spends a total of 15 minutes reporting to each other about the progress, what the member will do today and what impediments he faces. After the sprint execution, the team holds a meeting to demonstrate a working product increment to the product owner and other stakeholders called as sprint review meeting. After the end of each sprint, we get a subset of the product or a backlog to a ship ready state which is fully tested with all the features of that sprint. The extension of sprint is an indicator that the project is not on schedule and something needs to be performed. Therefore, it is extremely important to monitor the progress with the help of a burn down chart.

The role of product owner is to make sure that the right features get into the product backlog representing the users and the customer's requirements. Product owner helps to set the direction of the product. The role of scrum master is to make sure the project is progressing smoothly and ever team member has the resources he/she need to get his/her job done. Scrum master sets up meetings, monitor the work being done and facilitates release planning, acting more as a project manager. The team is cross-functional, self-organizing and intensively collaborative. The team pulls a few features from the product backlog which is then called as sprint backlog and decides how to proceed with the implementation of the features.

In the proposed documentation process, the product backlog can easily be generated from the requirement specification created previously. This may reduce the effort of product owner to remember the overall system while identifying the product backlog. At this stage, new features can be added which were missing in the requirement phase. New features added later in the project, can be documented and requirement specification must be updated by the technical writers with assistance of product owner. After finalizing sprint backlog, the technical writers can coordinate with development team in daily sprint meeting to document all the necessary information. To ship with the product, the technical writers can easily create a good system manual and required deliverables, as they are involved since the initial phase of the project.

2.5. Lessons learnt document

Lessons learnt document generalizes the challenges faced at the time of development. It contains the details about the project and the team. The challenges include both management and technical issues [19]. Lessons learnt are used at midpoints of the project and at project completion to record significant new understanding that have emerged as a result of the project. They are used to build the knowledge base of an organization and to establish a history of best and worst practices in project management and customer relations [20].

The technical writers document the experiences and impediments observed in scrum process, which can be used as reference in future projects and can provide a base for any process

improvement. Thus, the lessons learnt document can be useful for evolving continuous improvements.

2.6. Document Management System

It is a system used to track and store electronic document and/or images of paper document [21]. The system helps in easy storage and retrieval of the required document. This system can also help the organization to access all types of data whether it is a word file, CAD file, spreadsheets, recorded emails, video files, digital photographs, images and so on, anywhere and anytime. This type of system is well suited to store any historical data and provide a feedback or knowledge of a new upcoming project.

In SCRUM, the document management system can guide developers to choose best of the solutions or approaches based on past experiences as all the documentation resides well managed and at a single location. Incorporating previous experiences and achievements can definitely reduce the impediments and uncertainties, thus, enabling the development team with quality decision making support.

2.7. Role of Technical Writer

The mind-set of novice programmers observes the code as the key artifact in a software project and is barely able to write a coherent sentence [11]. A good documentation is regarded as a powerful tool which describes a software system, its components and relationships. Poor system documentation is the primary reason for the quality degradation in work products, leading to confusion in maintenance [7]. Here, the role of technical writers comes into picture as software developers have the knowledge and technical writers have the skills [10]. Writers judges the importance of each task to decide what should be documented immediately and what can be left for safe delay [9]. Some tasks are to be documented as they proceed while few tasks can be delayed till their completion and then documented. In the proposed conceptual view of documentation, the role of technical writer starts with the project, creating contract document or a project charter. In collecting requirement specification, technical writer can help in information gathering, detecting and articulating customer requirements.

Technical writers also help in balancing the type and amount of information extracted from customers which can improve the SRS. Technical writers can better assess and plan documentation project and better meet customer needs. Technical writers can determine the questions that are concerned with the system regarding the ease of use and reliability by learning about customer needs early in the product development process. Further, technical writers involved from the beginning and often in the process can become an information resource throughout the process [22]. In case of design document, technical writers assist in creating architecture document, short in details but thick on explanation, serving in outlining the situation describing one of more alternatives and enumerating the pros and cons of each. The primary focus is to create a common source to be used by all team members [23]. The design document can further be a part for SCRUM in producing product backlog and help indentifying various activities. In Scrum, technical writer can be an integral part of the process from sprint planning, execution, review meeting, sprint retrospective, till product shipment. It allows for comprehensive coverage of every aspect of a product as it is being developed, removing the risk of accidently leaving out something important.

Technical writer can be involved in SCRUM process by attending the scrum meeting which stands for 15 minutes with rapid fire communication with the team members [24]. Technical writers are free to help each other during sprint as long as the team metrics remains accurate and the total effort does not exceed the initial estimates of creating a document. Technical writer can also document the code, algorithms, interfaces and APIs at this level.

Technical writer should also document sprint review meeting and retrospectives, which concern the accomplishments, process improvements and determines the achievements and impediments. This knowledge is useful and important for creating lessons learnt document.

Technical writer can use lessons learnt template throughout the development process at various levels to finally come up with an overall lessons learnt document useful for future projects as a feedback.

A technical writer is a skilled professional writer, who is able to design, create and maintain technical documentation. The documentation can be of many types such as design specifications, system manuals, white papers and other related documents [25]. A technical writer is able to produce a good documentation for developers, users and customers by possessing certain skills. They acquire the ability to organize ideas in graphically useful fashion, facilitate with technology, interact with SMEs (subject matter experts), write at all levels from brief abstract to book length manuscripts, cite and critically analyze the scientific literature in written work, convey complex information in appropriate fashion to audience different levels of physiological knowledge etc. Facilitated with above skills, technical writer should possess good communication skills, flexibility to adjust in any environment and should have patience for problem solving and troubleshooting [26] [27].

3. PROS AND CONS OF DOCUMENTED APPROACH

Agile methodologies rely on an individual's communication and memory instead of documentation and believe documentation is an overhead cost and should be reduced or eliminated [28]. While documentation serves as a way to bring new members up to speed, it is useful when transitioning the project to a maintenance team. Documentation can also be used within a contracting relationship to indicate what work has been done and what progress has been made [29]. According to Ambler, software documentation responds to three basic necessities, contractual agreement, support a software project by allowing team members to gradually conceive the solution to be implemented and allow the team to communicate the implementation details with time to the maintenance team [10].

A good documentation deeply describes a software system, its components and relationships [7]. It is useful in better understanding and communication among the team members making it easy to learn and relearn the system. Documentation constitutes a collective knowledge of the organization which enhances knowledge transfer, historical information and assists in product evolution and maintenance.

At the same time, documentation process is a time consuming activity which involves a certain amount of cost. A document is a waste if written once and not updated properly to reflect the changes.

On the other hand, poor or missing documentation causes defects in evolution and maintenance phase. This can further lead to a significant loss of system knowledge if a skilled team member leaves and his knowledge and experience is not recorded. Having no documentation, the new members added late in the project will have to fully rely on other team members causing them to be distracted from their work.

Therefore, learning from established documents significantly affect activities and actions resulting in fewer mistakes and errors and increasing development speed as the members know what to do and what outcome to expect.

4. CONCLUSION & FUTURE WORK

Recently, agile software development has changed a view for traditional software documentation, proposing a model that rely more on informal communication rather than extensive documentation. However, the model does not support contract and maintenance which still has a great need for documentation.

Appropriate documentation can assist people as well as the developer to gain confidence in the system built in process. It is not only the developer being benefited by documentation but

technical writer also gains a lot. The writer, if better understands the system can write the documentation in advance, which can further help to focus the mind and prevent false starts. Despite the advantage of documentation its usefulness can be effective only if document maximize stakeholder return on investment, stakeholder knows the total cost of ownership, document fulfills the purpose and documents are sufficiently accurate, consistent and detailed. Also, the technical writer has sufficient skills to produce a good documentation.

A conceptual view of documentation is proposed, which can be useful to software community to recognize documentation as an important artifact and a new role is introduced in agile method as technical writer who can lend a hand in creating and managing documentation as per the requirement.

5. REFERENCES

[1] M.F.F. Nasution and H.R. Weistroffer. “Documentation in Systems Development: A Significant Criterion for Project Success”. Proceedings of the 42nd Hawaii International Conference on System Sciences, Jan 2009, pp. 1-9.

[2] P. McInerney and F. Maurer. “UCD in agile projects: dream team or odd couple?”. Magazine interactions, ACM, New York, USA, Volume 12 Issue 6, Nov 2005.

[3] G. Boström et al. “Extending XP practices to support security requirements engineering”. Proceedings of the 2006 international workshop on Software engineering for secure systems, ACM New York, NY, USA, 2006.

[4] P. Abrahamsson et al. “New directions on agile methods: a comparative analysis”. ICSE '03, Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society Washington, DC, USA, May 2003, pp. 244-254.

[5] M. James. CollabNet Inc., “Six Pages About SCRUM”. Available: www.open.collab.net/media/pdfs/SixPagesAboutScrum.pdf [Nov. 15, 2010].

[6] N.J. Kar. “Adopting Agile Methodologies of Software Development”. Available: www.infosys.com/Fresearch/Fpublications/FDocuments/Fadopting-agile-methodologies.pdf [Dec. 1, 2010].

[7] M. K. Mattsson. "Problems in agile trenches". ESEM '08, Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, ACM New York, NY, USA, 2008.

[8] F. Paetsch, A. Eberlein and F. Maurer. "Requirements Engineering and Agile Software Development”. WETICE '03, Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE Computer Society Washington, DC, USA, 2003.

[9] J. Baptista. “Agile Documentation with uScrum”. SIGDOC '08, Proceedings of the 26th annual ACM international conference on Design of communication, ACM, New York, NY, USA, 2008.

[10] S.W. Ambler. “Agile/Lean Documentation: Strategies for Agile Software Development”. Available: http://www.agilemodeling.com/essays/agileDocumentation.htm [Dec. 1, 2010].

[11] T. Clear. “Documentation and agile methods: striking a balance”. ACM SIGCSE Bulletin, Volume 35 Issue 2, ACM New York, USA, Jun. 2003.

[12] B. Selic. “Agile Documentation Anyone?”. IEEE Software, Volume 26 Issue 6, IEEE Computer Society Press Los Alamitos, CA, USA, Nov. 2009, pp.11-12.

[13] A. Sillitti et al. “Managing Uncertainty in Requirements: a Survey in Documentation-driven and Agile Companies”. Software Metrics, 2005. 11th IEEE International Symposium, Oct. 2005, pp. 10-17.

[14] T.C. Lethbridge et al. “How Software Engineers Use Documentation: The State of the Practice”. IEEE Software, Volume 20 Issue 6, IEEE Computer Society Press Los Alamitos, CA, USA, Nov. 2003, pp. 35-39.

[15] S. Bhalerao and M. Ingle. “Analyzing the modes of communication in agile practices”. 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Sep. 2010, pp. 391-395.

[16] Software Design Document. Available: http://en.wikipedia.org/wiki/Software_design_document [Dec. 3, 2010].

[17] T. Ryan. “The Anatomy of a Design Document, Part 1: Documentation Guidelines for the Game Concept and Proposal”. Available: http://www.gamasutra.com/view /feature/3384/the_anatomy_of_a_design_document_.php [Dec. 3, 2010].

[18] R. S. Pressman. “Software Engineering: A Practitioner's Approach”. McGraw-Hill, 2000.

[19] “Lessons Learned: Scientists, Distributed Teams, and Groupware”. Available: http://en.wikiversity.org/wiki/Lessons_Learned:_Scientists,_Distributed_Teams,_and_Grou pware [Dec. 8, 2010].

[20] “Documenting Lessons Learned”. Available: http://pmtips.net/documenting-lessons-learned [Dec. 8, 2010].

[21] “Document management system”. Available: http://en.wikipedia.org/wiki/Document_management_system [Dec. 8, 2010].

[22] “Writing Software Requirements Specifications”. Available: http://www.techwr-l.com/techwhirl/magazine/writing/softwarerequirementspecs.html [Nov. 11, 2010].

[23] “Software documentation”. Available: http://en.wikipedia.org/wiki/Software_documentation [Dec. 8, 2010].

[24] K.A. Johnson. “A technical communication intern’s first look at the agile development process”. Available: http://4378.pbworks.com/f/4378+Paper_kj.doc [Dec. 10, 2010].

[25] “Technical Writer”. Available: http://en.wikipedia.org/wiki/Technical_writer [Dec. 10, 2010].

[26] “Technical Writing Skills”. Avialable: http://www.the-aps.org/careers/careers1/gradprof/gwriting.htm [Dec. 10, 2010].

[27] “Five Skills Every Technical Writer Needs”. Available: http://idratherbewriting.com/2007/09/26/five-skills-every-technical-writer-needs [Dec.10, 2010].

[28] D. Longstreet. “The Agile Method and Other Fairy Tales”. Available: http://www.SoftwareMetrics.com/Agile [Nov. 11, 2010].

[29] M. Coram and S. Bohner. “The Impact of Agile Methods on Software Project Management”. ECBS '05, Proceedings of the 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, IEEE Computer Society Washington, DC, USA, Apr. 2005, pp. 363-370.

[30] “Contract”. Available: http://www.thefreedictionary.com/contract [Dec 10, 2010].