After the Scrum: Twenty Years of Working without Documentation

Sukanya Ratanotayanon
Department of Informatics
University of California, Irvine
sratanot@uci.edu

Jigar Kotak
Department of Informatics
University of California, Irvine
jkotak@uci.edu

Susan Elliott Sim
Department of Informatics
University of California, Irvine
ses@ics.uci.edu

ABSTRACT

Agile processes enable software development projects to react to rapid changes in the development environment. However, they are often criticized for not creating and maintaining standard documentation such as requirements and design documentation. The lack of documentation can be detrimental for maintaining knowledge, especially in the long run, because there is no explicit medium for communication to new people and among existing developers. This poses an important question: whether the use of agile processes in long run is feasible. In this paper, we presented a field study of an organization that has been using an agile process for more than 20 years and has been successful in maintaining knowledge over that period. Instead of written documentation, they use living documents, well-connected communication, and working software are prioritized as mediums for maintaining knowledge. However, success is not easily achieved. There are important factors that enable the organization to use the current practices successfully. These factors are: shared values, overlapping knowledge among team members, low turnover rate, and well-understood requirements.

1. Introduction

Processes in the agile family have gained increasing popularity recently due to the competitive environment and rapid changes in both technologies and requirements. The lightweight characteristic of agile processes enables fast-paced development and rapid reaction to change. However, much skepticism toward agile processes has been shown, especially by the proponents of plan-based software processes, due to the lack of standard documentation, which can be detrimental for maintaining and distributing knowledge among project members throughout a project's life cycle.

Agile software processes do not produce and maintain any high-level documents other than source code and comments. Instead, it recommends communication and collaboration among people in the project as a means of maintaining knowledge rather than using documentation. The process suggests that documentation be created and maintained only when necessary and only to facilitate communication. For example, the process advocates documenting important knowledge that will help others understand the source code as comments. The facts that there is no explicit medium for knowledge transferring, and most important knowledge resides only within team members' heads raise an important question. Is long term use of the agile approach feasible? This concern is especially valid when the software project reaches its maintenance phase and the original developers of the system are leaving or are no longer available.

Most field studies of agile projects have not investigated this issue. They have studied projects that only recently adopted agile processes and have focused on how the agile processes were adopted [1], [2]. We have conducted a field study within a small organization that has been using an agile process for more than 20 years. In this study, we interviewed all staff members who were involved in two particular software projects. The first software project is being implemented as a computer system for the first time and is in the development phase. The second project is one of the organization's critical applications and is the third generation of computer systems performing the same task. This second application is in its maintenance phase and has already lost some of its original developers. However, even without documentation, the remaining developers are able to sustain the knowledge required to maintain their software successfully.

At first blush, one would expect that scrums, or frequent, informal face-to-face communication would be insufficient to sustain knowledge over decades. However, this was not the case. We found that mediums through which knowledge is mainly retained and distributed are: living documents, well-connected communication, and exemplar software systems. However, there are a number of factors that enable this organization to use these communication mediums effectively, and these have led to successfully sustaining and distributing the knowledge. These factors are: shared values, overlapping knowledge among team members, a low turnover rate, and well-understood requirements.

In the next section, we review related studies.. The method employed in our study and the characteristics of the organization and projects under study are presented in

Section 3. Section 4 describes in detail the mediums used to maintain and distribute knowledge in this organization. Section 5 analyzes the factors that enable them to successfully use these mediums effectively. Section 6 presents concerns that the organization has about maintaining its knowledge in the future. We conclude our paper in Section 7.

2. Related Work

Agile processes are a family of lightweight processes that share a common framework called "Agile Manifesto" established in early 2001 by the Agile Alliance [3]. The Agile Manifesto declares that agile processes value:

- Individuals and interactions over processes and tools,
- Working software over comprehensive documentation,
- Customer collaboration over contract negotiation,
- Responding to change over following a plan.

In addition, a true agile process will have the following characteristics: iterative, incremental, self-organized by team member, and emergent [4]. Examples of famous agile processes are Extreme Programming (XP), Scrum, Crystal, Adaptive Software Development (ASD), Feature-Driven Development (FDD), and Lean Development (LD).

Agile processes are popularly adopted by small teams building software that requires fast development and fast reaction to changes. Agile processes normally prioritize communication and shared tacit knowledge as a means to maintain and sustain knowledge in software projects. Documents are created only to facilitate the communication and are updated only when necessary [5]. This raises skepticism about the ability of agile processes to maintain and distribute knowledge among team members throughout the development life cycle.

Maintaining and distributing knowledge is crucial in software development processes. This is especially true in the case of maintenance, when sometimes a person who is responsible for maintaining the system is not involved in its development. Information has to be stored in some form in order to help the maintainer gain an understanding of the system. Failure to address this issue could lead to a "thin spread of application domain knowledge," and "communication and coordination breakdown" [6], which eventually lead to the failure of the project.

In traditional plan-based processes [7-9], a standard set of documents associated with software development phases is prescribed as the medium for communication to new people on the project, existing developers, users, and other related software projects. These documents are used to enforce conformity of coding standards, design documents, and requirements, as well as to provide mechanisms for tracking project features, status, and bugs.

The cost of producing and maintaining these documents, however, poses a problem, especially in small teams such as those practicing agile processes. Although there are many advocates for using documentation as a means to transfer and sustain knowledge [10-12], there is also evidence against the effectiveness of documents as a means to maintain and distribute knowledge. Studies show that, in practice, developers do not do a good job of documenting [13]. In addition, developers don't feel that it's worthwhile to update certain types of documents. Therefore, documents are not created and maintained a timely manner, and they become outdated [14, 15]. This inconsistency undermines the value of documentation as a means for maintaining knowledge in an organization.

3. Empirical Method and Field Site

In order to better understand how knowledge is maintained and transferred without the benefit of documentation, we conducted a field study of an organization that has been using an agile development process for more than 20 years. This section presents the method employed in our study and the characteristics of the organization and projects studied.

3.1 Method

We conducted interviews with 9 technical and non-technical staff members who were involved in two specific software development projects. A detailed description of the projects is presented in section 3.2. We employed semi-structured interviewing techniques, which involve a set of questions designed to cover the ranges of topics. These questions are open-ended in order to encourage subjects to talk at length about their experiences and also help us learn about other related issues. The questions, selected to promote conversation, fall into three categories: (i) their background and their roles in the project, (ii) documents and information required for their tasks, and (iii) the means used to obtain required information and to record information obtained during their tasks.

Each subject was interviewed individually for 30 to 60 minutes. For each interview, there were two interviewers: the first interviewer took the role of ensuring that all the topics in the protocol were covered, and the second interviewer, freed from focusing on the protocol, focused on following up on interesting but unanticipated remarks. In addition to the notes taken during the interview, interviews were tape recorded.

3.2 Field Site and Project Characteristics

The interviews were conducted at a small administrative office at a university, which we will call "IStar." IStar has a flat hierarchy and employs 14 individuals: 6 nontechnical staff and 8 technical staff. Non-technical staff carry out IStar's business. Technical staff implement software to support IStar's works. Each division has its own manager, who reports to the head of the organization. At the time of the interviews, there were 7 full-time developers and 1 student programmer in the technical division.

Recently, the trend of software development in IStar has been to "move everything to the web". All paper-based and client-server systems were being migrated to web applications in order to: (i) provide better service to the students and staff, and (ii) eliminate additional costs attached to paperwork. We studied two specific web application projects, WA1 and WA2. WA1 is a web application for administrators to view and perform job-related functions concerning a student's enrollment. WA2 is a web application that allows students and counselors to process requests for graduation. WA1 is a version of an existing client-server system developed in IStar that has been ported to a web application and is already in the maintenance phase. WA2, on the other hand, is a newly created application that replaces a paper-based system and recently went into production.

Although we interviewed only staff members who were involved in WA1 and WA2, the process employed by both projects is the same for all software development in IStar, and it has been used for more than 20 years. The agile processes employed are not one of the famous agile processes, but IStar's own specific process. We perceived this process as an agile process because, in addition to being a lightweight process, it presents the required agile attributes [4]:

Incremental: Developers do not elicit all the requirements up front and implement the whole system at once. They start with core requirements and then implement these first. Additional features are added later, one at a time.

Iterative: An evolutionary prototype [16] is implemented and used to clarify the requirements of the system.

Self-organizing: Developers are allowed to make their own estimates and to determine how to handle the assigned work.

Emergence: There is no predetermined plan created for the projects. Only a rough deadline is estimated for each feature. The development is carried out by the technical staff, and management tracks the status of the project via frequent informal communication.

4. Maintaining Knowledge without Paper

The common opinion about documentation shared in IStar, including the management team, is that the cost of creating and maintaining the documents is not worth its usefulness. Therefore, rather than using documentation, IStar chose alternative mediums for communicating and sustaining knowledge throughout its software life-cycle. In this section we will describe the mediums utilized by IStar.

4.1 Living Documentation

Rather than focusing on creating documentation, IStar believes in using "living documents" to maintain and transfer knowledge in software projects. The living document refers to the staff members, especially "*pioneer employees*," who are technical and nontechnical staff members who have been employed for more than 20 years.

Pioneer employees play an important role in maintaining knowledge throughout the software life cycle in IStar due to their deep understanding of domain knowledge assimilated during the years. We learned of their importance for sustaining and passing along important knowledge through interviews with WA1's developers. Since WA1 is a re-implementation of an existing system with web application technology, it shares similar characteristics with its predecessor system. Some pioneer people were involved in the development and maintenance of WA1's predecessor system. They are able to provide the developers of WA1 valuable advice on requirements, data description, design decision and solution to some known issues, which were obtained from developing the predecessor system.

There is no documentation created during the development phases, so how do existing developers transfer their knowledge to newcomers when they leave the project? This situation arose with WA1, which had some student developers who left before the project stabilized. The common belief in agile projects is that comments in the source code provide enough information for new programmers to understand the application. New developers in WA1 disagree, however. What has allowed other developers to obtain the required knowledge to continue the task lies in IStar's processes for managing the departure and arrival of employees.

Typically, a developer who is leaving IStar will suspend other work and reserve the last few days only for "explaining" and "talking" to other developers whose work is related about various aspects of the tasks. Since this office community is tightly knit, developers are aware of each other's work. This lay a foundation that allows the remaining staff to understand the information left by the departing staff. This practice is so engrained among the people who work at IStar that when people leave the office on a good note they are still considered to be part of the office. They are willing to answer telephone queries pertaining to their applications, and if need be even to come back to solve some of the problems.

IStar has a philosophy that "more familiarity leads to more knowledge." When a developer first arrives in IStar, he is usually given a task to perform low priority enhancement to the existing systems. The task allows the new employee to assimilate knowledge about the application and IStar's process. Other programmers who are aware of the assigned task "fill in" the new hire with information that is vital for the task through "informal talking" or "walk-through" discussions and meetings. This type of activity usually lasts a couple of months.

4.2 Well-Connected Communication

Like other agile projects, IStar uses communication as the primary medium for transferring and maintaining knowledge. However, what we find distinctive in the participating organization are its highly cooperative environment and the frequency of ad hoc communication among the staff. The organization's environment cultivates informal communication and a willingness to provide information and help to others. Technical and non-technical staff are on a good terms and have strong connection with each other. Due to its flat hierarchy, there are very few communication obstacles between people in different teams or even with the management team members. Each individual in the organization is approachable. Ad hoc conversation is welcomed by individuals and encouraged within the organization. Staff members are welcome to walk up to others and initiate informal communication, even with those in managerial positions.

By having frequent informal communication with non-technical staff members who are clients of the application and domain experts, developers do not have to rely on having system requirements documented. They are able to collect requirements, assimilate domain knowledge, and obtain quick feedback about the implemented system through frequent informal communication.

In addition, this connected communication facilitates knowledge transferring for various purposes, such as following up an idea presented in a meeting, distributing tasks, tracking the status of projects, and getting advice from fellow developers. It allows one staff member to be aware of and to understand the responsibilities of some of other staff, which results in overlapping knowledge, as presented in Section 5.2.

4.3 Reference Implementations and Prototyping

IStar's developers also utilize exemplar software systems as sources of information for their implementations, especially when re-implementing a system that has predecessor software. The effort spent in developing the predecessor is therefore not lost, but used in the current project. For an example, WA1's developers, with the help of pioneers, are able to use WA1's predecessors as examples for user interfaces, implementation of data processing, and some computation processes. Seeing running software also helps them to have a better understanding of business rules and descriptions of the data being processed.

Since IStar develops its software in an incremental and iterative manner, evolutionary prototypes [17] are also used to aid in communication among staff. Typically, programmers start by developing a prototype, providing core functionality of the system as a proof of concepts. Interfaces of the prototype or its screen shots are then shown to non-technical staff and fellow developers to clarify the requirement and gain design suggestions about its usability. The prototype itself is used to gain feedback and a clear requirement misunderstanding. As an example, sometimes non-technical staff members provide feedback to programmers by running the program to determine any incorrect behaviors. The management team also tracks project status by observing the current working system.

5. Success Factors

In the previous section, we described how IStar maintains and transfers knowledge without documentation. This section presents factors that enabled this organization to use these practices successfully. The reasons center around creating a positive work environment with shared values, ensuring that developers have overlapping knowledge areas, which in turn results in a low turnover rate.

5.1 Shared Values

At IStar, informal communication among peers is valued over standard documentation as a medium. All members share this value. When we asked about documentation, we were presented with the graph in Figure 1.

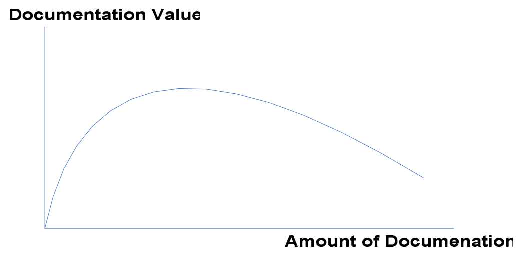

Figure 1: Amount vs. Value of Documentation

The graph shows the relationship between the value of documentation to the quantity of documentation according to IStar's staff. After a certain amount of documentation, its value decreases. The reason given was that the more documentation generated, the more time spent by existing developers to create it, and the more time needed by a new developer to read it. Another reason, given by the head of technical division, is that "a lot of people talk of creating documentation, but not many people do a good job at it." Their shared value and viewpoint on documentation cultivates informal communication and a willingness to provide information to other developers.

5.2 Overlapping Knowledge

Overlapping knowledge is key to filling the void left by the lack of documentation, for example, when a staff member leaves. Due to the closeness among the staff during their time of employment, employees discuss, explain, argue, and communicate via other informal means. In addition, various developers are involved in the same project, although at different times. As a result, overlapping knowledge areas emerge over time.

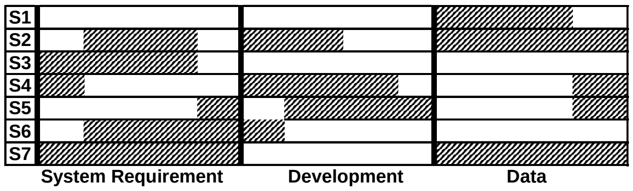

Figure 2: Overlapping Knowledge

To illustrate the overlapping knowledge areas, Figure 2 displays knowledge overlap among employees with respect to WA1. The figure shows that if any one of the employees has to leave the department permanently, people who share that employee's knowledge can combine their respective knowledge to fulfill that individual's duties. Moreover, because the employees maintain close ties with one another, if any one of the employees is leaving, the others are usually aware of this, and the last few days are spent "telling" people how they are doing and what they are doing. Their overlapping knowledge aids them in assimilating the information being passed on.

5.3 Turnover Rate

This organization has a trait that any organization would envy: a low turnover rate. Figure 3 shows that most of the employees have been working in the office for more than 10 years and some are still working even after 30 years of service to the organization.

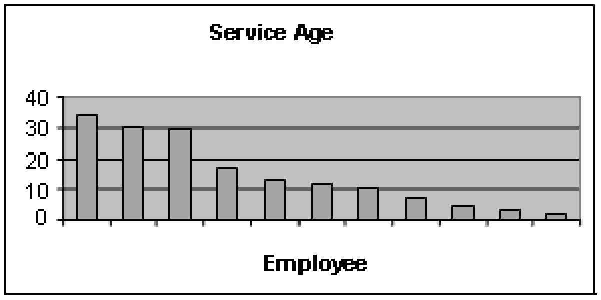

Figure 3: Service Age of Subjects

This low turnover rate aids in creating knowledge overlap among employees. Developers assimilate domain knowledge by interacting with non-technical staff members over many years. The pioneer employees can pass on their knowledge first-hand to the recent hires. As one of the non-technical staff members points out, "[she] would love to impart her knowledge if someone was ready to pick her brain."

The low turnover rate also helps to prevent the organization from losing information the staff members have. No matter how well aware the staff is toward others' responsibilities, some specific information is always lost when an existing staff member leaves the project, and it will take some time for a newcomer to pick up the knowledge. As mentioned above, the arrival and training process might take up to two months. Having a low turnover rate helps prevent the project from losing this time.

5.4 Well-Understood Requirements

Because developers are co-located with the business office and the goal of the software is to support IStar's work, developers are exposed to business logic, which helps them to grasp the domain knowledge. The non-technical team members and pioneer people who understand the business process are also there to give feedback and clarify requirements.

In addition, many critical applications have predecessors. A system is usually re-implemented to keep up with the technology; the requirements of the system behavior and workflow usually stay the same. The new system only has to mimic its predecessor behavior and computation. The system requirements were already gathered and analyzed when the predecessor system was implemented, so pioneer people who implemented it can provide this valuable knowledge of the requirements to current developers. If pioneer people forget the requirements, the working software can be run to obtain the information.

6. Moving Forward

According to a pioneer technical person, IStar has adopted this process of software development for more than 20 years. No high-level documents such as requirement or design documents were created. Informal documents created are mainly reminders, such as personal notes and online FAQs about the application's operations. It is worth noting that there is one document, "data description note," that is created and maintained regularly and is shared among developers. This note lists database fields, including their description for IStar's applications. This is not unexpected, as IStar's applications are "information processing" systems. Therefore, a description of the data to be processed is important and it is impossible, even for pioneer developers, to remember all of the data description.

With 20 years of using these processes successfully, one would expect IStar to be settled and secure with these practices. However, this is not the case. As we can see, knowledge in IStar is maintained in its staff, especially its pioneer people. As the prospect of some pioneer people retiring is near, IStar is concerned about losing important information, such as critical application behaviors and usage, domain knowledge and various kinds of organization knowledge as these people leave. Surprisingly, they are trying to deal with this problem using documentation. Approximately five years ago, the organization initiated a documentation project by hiring an external consultant to document information from all the pioneer people. The focus of that document was on the operational aspect of the applications and business processes. However, it is doubtful that the document will be as useful in terms of completeness of information and effectiveness in distributing knowledge because interchanges among the staff, who are so used to informal communication to gain required information rather than reading. In other words, despite management concerns, this agile development group is reverting to type by favoring communication over paper.

7. Conclusion

In this paper, we reported on a field study of a small organization that has been using an agile development process for over two decades. Two successful agile software projects were examined in detail in order to investigate how information has been transferred and sustained without using traditional documentation. We found that its knowledge is maintained by using living documents, well-connected communication, and working software. In other words, scrums, or frequent, informal, and intense communication, is enough to sustain knowledge within an organization for a long period of time. Furthermore, when human memory fails, the environment provides aids to recall in the form of ingrained business processes and exemplar operational software systems.

Finally, using an agile process successfully also requires deep commitment at an organizational level. IStar is no exception. We also identified important factors that enable staff to use the current practices successfully. These factors are shared values, overlapping knowledge among team members, low turnover rate, and well-understood requirements. In summary, IStar demonstrates that with the right mechanism and a hospitable culture it is possible to use agile processes and work successfully without documentation for many years.

8. Acknowledgments

We thank IStar staff for their time and patience with our questions.

9. References

[1] M. Cohn and D. Ford, "Introducing an agile process to an organization [software development]," Computer, vol. 36, pp. 74-78, 2003.
[2] H. Svensson and M. Host, "Introducing an agile process in a software maintenance and evolution organization," pp. 256-264, 2005.
[3] J. Highsmith, Agile Software Development Ecosystems, USA: Addison-Wesley Publishers, 2002, pp. 448.
[4] B. Boehm and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, USA: Addison-Wesley Publishers, 2003, pp. 304.
[5] S. Ambler, Essay: Agile Documenation, 2005.
[6] B. Curtis, H. Krasner and N. Iscoe, "A field study of the software design process for large systems," Communication of ACM., vol. 31, pp. 1268-1287, 1988.
[7] D.L. Parnas and P.C. Clements, "A rational design process: how and why to fake it." IEEE Transactions on Software Engineering, vol. SE-12, pp. 251-257, 1986.
[8] I. Jacobson, G. Booch and J. Rumbaugh, The Unified Software Development Process, 1999.
[9] W.W. Royce, "Managing the Development of Large Software Systems," Proc. WESTCON, 1970.
[10] S.C.B.d. Souza, N. Anquetil, K. Oliveira and thia M.de, "A study of the documentation essential to software maintenance," in SIGDOC '05: Proceedings of the 23rd annual international conference on Design of Communication, pp. 68-75, 2005.
[11] F.A. Cioch, M. Palazzolo and S. Lohrer, "A Documentation Suite for Maintenance Programmers," in ICSM '96: Proceedings of the 1996 International Conference on Software Maintenance, pp. 286-295, 1996.
[12] T. Sauer, "Using design rationales for agile documentation," in Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE 2003. Proceedings. Twelfth IEEE International Workshops, pp. 326-331, 2003.
[13] M. Visconti and C.R. Cook, "An overview of industrial software documentation practice," in Computer Science Society, SCCC 2002. Proceedings. 22nd International Conference of the Chilean, pp. 179-186, 2002.
[14] T.C. Lethbridge, J. Singer and A. Forward, "How software engineers use documentation: the state of the practice," Software, IEEE, vol. 20, pp. 35-39, 2003.
[15] A. Forward and T. C. Lethbridge, "The relevance of software documentation, tools and technologies: a survey," in Proceedings of the 2002 ACM symposium on Document engineering, McLean, Virginia, USA: ACM Press, pp. 26-33 , 2002.
[16] L. Horst, S. Matthias, Z. Heinz and lighoven, "Prototyping in industrial software projects: bridging the gap between theory and practice," pp. 221-229, 1993.
[17] C.Z. Jean and D.T. Peter, "An insider's survey on software development," pp. 178-187, 1982.