Factors that Significantly Impact the Implementation of an Agile Software Development Methodology

Jeffrey A. Livermore
Walsh College/Business Information Technology/Detroit, MI, USA
Email: jlivermore@walshcollege.edu

[1]***Abstract*—The Internet economy has altered the current rules of software engineering. Traditional development methodologies have proven too cumbersome to meet the rapidly changing requirements and short product cycles demanded by business today. To meet these rapidly changing requirements, software developers have developed agile software development methodologies (SDMs) utilizing iterative development, prototyping, templates, and minimal documentation requirements.**

This research project investigated agile SDM implementation using an online survey sent to software development practitioners worldwide. This survey data was used to identify factors related to agile SDM implementation. The factors that significantly impacted agile methodology implementations included training, management involvement, access to external resources, and corporation size. Other factors such as using models, having an implementation plan, collocating the development team, and developing software for Internet or intranet use did not significantly impact the implementation of an agile software development methodology.

A number of the factors that impact the implementation of an agile development methodology are completely under the control of management. Organizations that are considering implement ting an agile methodology are able to manipulate some of these factors to increase the opportunities for success of their methodology.

***Index Terms*—agile software development, Extreme Programming, Scrum, agile methodology implementation**

I. Introduction To Agile Methologies

The growth of the Internet and the digital economy has altered the profession of software engineering. Traditional software development methodologies (SDMs) are being replaced by new light or agile SDMs. These agile SDMs are characterized by iterative development, continuous code integration, and the ability to handle changing business requirements [1].

Extreme Programming (XP) is perhaps the most popular agile methodology. XP is based on a series of coding and management concepts that include: having the business customer on-site with the development team, pair programming, collective code ownership, continuous code integration, small releases, designing tests before writing code, standup meetings, refactoring, and 40-hour work weeks [2].

Other popular agile SDMs are Scrum, Crystal Methods, and Feature Driven Development (FDD). All of these methodologies are fundamentally different from traditional SDMs and help organizations meet the challenges of today's digital economy [1].

The use of agile methodologies enable software developers to produce higher quality software in a shorter period of time. Agile methodologies were developed to improve the development process by removing barriers to accepting business requirement changes during the development process. It is not necessary to freeze or lock in business requirements and design details while developing software with an agile methodology [3]. Agile SDMs all share several qualities including prototyping, iterative development, and minimal documentation [4].

A. Extreme Programming

Extreme Programming was developed at Chrysler by Kent Beck while working on a payroll project as a member of a 15 person team. Beck continued to refine and improve the XP methodology after the project was completed until it gained worldwide acceptance in 2000 and 2001 [5]. XP can improve software quality while shortening functionality delivery schedules.

XP is based on a set of concepts and practices that include having the customer collocated with the development team, pair programming, collective code ownership, and the use of metaphors to describe business situations [1]. Having the customer collocated with the development team changes the customer's traditional role of a remote unapproachable user to being a full member of the development team. Other XP principles include: designing tests before developing code, maintaining an open workspace, daily stand-up meetings, code refactoring, and a work week of no more than 40 hours to minimize staff fatigue and loss of perspective.

Based on "Factors that Impact Implementing an Agile Software Development Methodology" by J. Livermore which appeared in Proceedings of IEEE Southeastcon 2007 Conference. © 2007 IEEE

XP contains development practices that are new to many organizations and developers. The practices of pair programming, open workspaces, and the 40 hour workweek may lead to resistance from developers and management [6]. Another practice unique to XP is holding a daily stand-up meeting. The development team meets every morning to exchange information and the team members stand during the entire meeting to help keep the meetings short [7].

B. Scrum

The Scrum methodology was specifically designed to handle rapidly changing business requirements. The Scrum name is derived from a strategy used in the sport of English Rugby. In a Rugby scrum, the ball is passed back and forth between team members to move the ball down the field. The Scrum methodology moves a project forward by improving communication between team members and breaking the work into a series of "sprints" [8]. A sprint should last between one and four weeks [8]. All development sprints should be kept to less than thirty days. Scrum focuses more on management of the development process than software coding techniques [9].

Like XP, Scrum was designed to work with small teams of ten or less members, however Scrum is a methodology that can be used effectively on both small and large projects. Individual teams can use the Scrum techniques on small or medium projects. Large projects can be broken into subprojects and a Scrum team assigned to each subproject. The communication and priority management negotiation between the subproject teams can be managed with standard Scrum techniques.

C. Crystal Methods

Crystal Methods is an agile SDM based on the premise that people impact software development projects more than tools or processes [5]. Crystal Methods is a toolkit or collection of methodology elements that organizations combine into appropriate methodologies to suit individual projects. Large projects and projects that impact public safety require more methodology elements than small non-critical projects. With Crystal Methods, organizations only create and use as large a methodology as their project and business needs demand.

According to Highsmith [5], the shade of Crystal Methods, or the amount of methodology elements used in a development project is determined by three factors. The first factor is the amount of communication necessary between the members of the development team. This factor is affected by the physical location of development team members, the office layout, and the personalities of the team members. The second factor is the presence of life-threatening implications if undiscovered software defects are present in the software when it is released. The third factor is the presence of corporate priorities that complicate the development process.

D. Feature Driven Development

The FDD methodology was developed for a bank project in Singapore [10]. The bank's development project required an iterative development process that was both easy to use and provided accurate progress reporting for management. FDD was developed by Coad and DeLuca to meet both these needs.

FDD is a five step process that does not require extensive training for a development team to use it [10]. The first three steps are: develop an overall model of the desired application, develop a list of the desired features, and prioritize that list into an implementation plan. The fourth and fifth steps are where the development iteration occurs. Each development iteration produces a deliverable for the customer. As features are developed and released, the feature list is reprioritized to keep the development team working on the highest priority features with the most value to the business customer.

FDD can incorporate agile development techniques from other methodologies. For example, FDD works very well with the XP practices of pair programming and daily standup meetings. The iterative fourth and fifth steps of FDD can also be time boxed to help manage the development process [10]. Time boxing enables the customers to maintain better control of the development priorities and determination of which functionality gets developed.

E. WISDOM

The Whitewater Interactive System Development with Object modules (WISDOM) is another agile SDM. WISDOM was developed between 1997 and 1999 for use at small companies [11]. Small companies frequently have different business requirements than large companies and may not have the large financial resources necessary to fund a large software development project. WISDOM was designed to match the needs of small companies by utilizing an iterative process of refining a prototype [12]. WISDOM has no documentation requirements outside of the use of Unified Markup Language (UML) to specify the software architecture.

F. Traditional Software Development Methodologies

Traditional methodologies such as SDM-70 or Method-1 were developed before current computing technologies such as the Internet, XML, wireless networking, or ubiquitous computing were in existence. Traditional methodologies were both innovative and effective within the context of existing technologies and relatively static business requirements. Traditional methodologies require extensive documentation, freezing or locking in business requirements during the entire development process, and require changes to existing software products and documentation produced prior to when a business requirement changes [2].

The factors that impact implementing a traditional SDM were researched by Roberts, Gibson, Fields, and Ranier in 1998 [13]. This early implementation study found that having a complete organizational transition plan for a new SDM, management involvement and commitment, using models, and providing access to

external resources all impacted the implementation of traditional SDMs. The Roberts study became a seminal work that served as the inspiration for this research project.

II. Research Hypothesis Review

The research survey instrument was designed to collect data that would answer research questions connected to eight hypotheses. The first five hypotheses were drawn from the earlier Roberts study on traditional methodology implementation with the remaining three hypotheses drawn from a literature review. The eight research hypotheses were;

H1: Training on the use of an agile SDM will have a significant impact on implementing that methodology.

H2: Active management involvement and support will have a significant impact on a methodology implementation.

H3: Having a compete methodology implementation strategy will significantly impact implementing that methodology.

H4: Selecting an agile SDM that utilizes models and templates will significantly impact the implementation of that agile SDM.

H5: Providing the development team access to external resources such as off-site training sessions, journals, consultants, books and online resources will have a significant impact on agile methodology implementation.

H6: Developing software for Internet or intranet applications as opposed to traditional computing platforms (mainframe, midrange, PC) will have a significant impact on agile methodology implementation.

H7: The size of the corporation or software development team will have a significant impact on agile SDM implementation.

H8: Collocating the development team will have a significant impact on agile methodology implementation.

III. Research Methodology

A survey instrument was developed to collect information on the eight research hypotheses regarding agile SDM implementations. The questions in the survey instrument came from the research hypotheses drawn from the Roberts et al. study and the literature review. The survey instrument was reviewed by a panel of peers for readability and then by a panel of agile SDM experts for content validity. The survey instrument was placed online for six weeks.

The survey instrument was designed to be adaptive so the questions presented to each respondent were determined by their answers to the initial series of questions that inquired about the respondent's experience with methodology implementation. For example, XP users were presented with questions about implementing and using XP as opposed to Scrum users who were asked questions about implementing and using Scrum.

Obtaining relevant response information on agile methodology implementations required identifying a population of software developers with SDM experience. This population was found in the Software Engineering Institute's Software Process Improvement Network (SPIN). SPIN consists of local chapters of individuals who are dedicated to improving the processes used to develop software [14]. An invitation to complete this research survey with a hyperlink to the survey was sent via e-mail to the presidents of all domestic and international SPIN chapters and a small number of similar organizations. The chapter presidents were asked to complete the survey and also forward the invitation to all of their members. The survey was also sent to authors who had published articles on agile SDM implementation.

IV Survey Results

A total of 112 survey responses was received. Incorrect e-mail addresses caused 23 of the survey e-mails to be rejected. Six domestic SPIN chapter presidents agreed to forward survey invitations to a combined membership of 1,803 software professionals interested in software development. Survey invitations were also sent to 143 authors who had published books or articles on agile SDM implementation or usage. There were 58 incorrect addresses for the authors that caused their invitations to be rejected. There were 112 survey responses received from the 1,946 individuals who received an invitation to participate in the survey for a return rate of 5.76%. No incentives were offered to complete and submit the lengthy 66 question survey instrument. The survey instrument was placed online for six weeks.

This survey research had a low response rate of 5.76%. A low response rate limits the applicability of the results to the larger population. Additional research should be conducted with a different research sample or with modifications to the research methodology that increase the response rate. This may be accomplished with a shorter survey instrument or survey completion incentives.

Agile methodology users provided 71 of the responses. More than third (26) of those organizations actively use XP. Responses were received from agile methodology users, traditional methodology users, and organizations that do not use any form of methodology to determine the effectiveness and benefits received from implementing agile methodologies. Scrum was implemented at eight of the responding organizations, Feature Driven Development at four, Dynamic System Development Methodology at three, Adaptive Software Development at one, and 34 organizations developed their own agile SDM. Ten of these homegrown methodologies were built on a foundation of XP practices.

A. Respondent Demographics

The responses came from a diverse group of well-educated and experienced IT professionals. The individuals that responded to the survey had an average of 14.02 years of professional experience. These individuals had a wide variety of job roles within their

organizations. The respondent job roles are contained in Table 1. The respondents came from a variety of industries with the majority coming from IT or consulting firms. Table 2 lists the industry affiliations of the respondents. Table 3 lists the education levels of the survey respondents.

Table 1.
Respondent job roles

Job Role	Entire Sample	Agile Users
Developer	34	22
IT Management	20	17
IT Senior Management	11	10
Corporate Management	13	7
Business Partner/Use	2	1
Other	11	7
No answer	21	7
Total	112	71

Table 2.
Industry demographics

Industry	Entire Sample	Agile Users
Consulting	13	11
Education	3	2
Government	4	2
IT	31	23
Financial	14	9
Manufacturing	3	2
Retail	3	2
Telecommunications	5	1
Transportation	1	1
Utility	1	0
Other	15	11
No Answer	18	7
Total	112	71

Table 3.
Respondent education levels

Highest level of education completed	Entire Sample	Agile Users
High School	6	2
Trade School	0	0
2-year degree	2	2
4-year degree	37	25
Graduate degree	49	36
No answer	21	7
Total	112	71

B. Methodology Training

The survey results established a significant correlation between successful methodology implementation and receiving training on the methodology. A correlation between two variables is a statistical measurement of their tendency to increase or decrease with each other. A statistical correlation is considered significant if it is unlikely to have occurred by chance. Organizations that provided methodology training to their development teams were more likely to have a successful implementation of that methodology than organizations that did not provide training. This is a common sense result that was predicted in the literature search. Training in a methodology enables an organization to develop expertise and be better prepared to implement the methodology.

C. Management Support and Involvement

There was a significant correlation between methodology implementation success and management support and involvement. This positive correlation was both predicted in the literature and intuitively obvious. Management involvement and support should improve the success of virtually any business project.

D. Methodology Implementation Strategies

There was not a significant correlation between developing a complete plan for implementing an agile methodology and the successful implementation of that methodology. The lack of a correlation may be explained by the adaptable and flexible nature of agile methodologies. One of the survey respondents explained this as “Having a complete and workable plan in advance is contrary to the spirit of XP. We started with a rough idea of what we needed and improved it every week.”

E. The Use of Models and Templates

There was no significant correlation between agile methodology implementation success and the use of models or templates. A correlation existed for traditional methodologies but did not exist with agile methodologies. The underlying reason is likely that agile methodologies do not rely heavily on documentation templates the way that traditional methodologies do.

F. Access to External Resources

There was a significant positive relationship between access to outside resources and the successful implementation of an agile SDM. The survey instrument collected data on external resources such as books, journals, consultants, and attendance at methodology user groups and conferences.

This correlation was predicted by the literature and makes common sense. Allocating resources to almost any IT project will increase the likelihood that the project will be successful. Allocating resources such as consultants, books, and journals will increase the development team’s knowledge of the methodology and how to exploit that methodology’s features to bring benefits to the organization.

G. Internet/Intranet Software Development

The literature indicated that there were differences between developing software for Internet or intranet usage and traditional computing platforms. The research found that there was no significant correlation between traditional software development and developing software for Internet or intranet-based applications. This

result was not expected as Internet and intranet applications both require short development schedules and frequently changing business requirements [15].

H. Company and Team Size

There was no significant correlation between implementation success and the size of the development team. This result was unexpected as a number of researchers have stated that agile methodologies do not work effectively with large development teams [1].

There was a significant negative correlation between implementation success and the size of the organization attempting to implement an agile methodology. Larger organizations have more internal inertia and find it more difficult to implement change than smaller organizations.

I. Development Team Collocation

There was no significant correlation between implementing an agile SDM and collocating the development team with business customers. The majority of the organizations surveyed had collocated their development teams are collocated so there was no basis for a comparison between agile and non-agile methodology users. Collocation provides benefits to both types of development teams and the survey results document that collocation is a standard industry practice regardless of SDM utilization.

Collocation is done to improve communications within the development team. In addition to collocating their development teams, 36 of the responding organizations installed communication tools to improve communications within the development team. Eleven of the responding organizations spent more than $999 and four of the organizations spent more than $100,000.

J. Summary of Statistical Results

All of the statistical tests were conducted with SPSS statistical software. The correlation, number of responses used in each test, and the statistical significance of each correlation are listed in Table 4.

Table 4.
Hypothesis testing results

Research Hypothesis	Pearson Correlation	Significance	Number of Responses
One	.183	.068	71/68
	.240	.032	80/102
Two	.229	.073	62/68
Three	.224	.083	61/68
	.127	.232	91/102
Four	-.180	.173	59/68
	.015	.894	85/102
Five	.319	.008	68/68
	.228	.033	88/102
Six	.103	.422	63/68
	.122	.260	87/102
Seven	.041	.375	62/68
	-.117	.183	62/68
	-.173	.049	93/102
	.167	.055	93/102
Eight	.170	.188	62/68
	.053	.614	94/102

K. Cross methodology comparison

While it is difficult to compare methodologies, the survey results showed that different methodologies provide more benefits than others. A satisfaction score was computed by assigning numerical values to the survey responses indicating agreement or disagreement on the benefits received by methodology implementation. Every benefit received added to the satisfaction score. ASD received the highest benefits score but was only implemented by one organization. XP delivered the most benefits of all the methodologies implemented by more than one organization. Table 5 contains the calculated benefits score of all the methodologies implemented by te surveyed organizations. The economics of calculating the financial value of the benefits of implementing an agile methodology are not proven [15].

Table 5.
Methodology Differences

Methodology	Number of Organizations	Benefits Score
ASD	1	31
XP	26	27.8
FDD	4	26.5
Scrum	8	25.6
Homegrown agile methodology	34	24.6
DSDM	3	23.7

V CONCLUSIONS

This research determined that there are several factors under management's control that impact the implementation of an agile SDM. Training on the methodology, active management involvement and support, access to external resources, and company size all significantly impact the implementation of an agile SDM. Having a complete methodology implementation strategy, using models and templates, developing software for either Internet or intranet use, and development team collocation did not significantly impact successful implementation.

VI RECOMMENDATIONS

The organizations that are considering implementing an agile methodology control several factors that can impact how successful that methodology implementation may be. Organizations committed to a successful implementation should consider allocating the necessary resources to help make the cultural change to agile. Resources should be dedicated to methodology training, journals, and user group memberships to help prepare the staff to use the agile SDM.

Organizations committed to a successful agile methodology implementation should also evaluate the

different agile methodologies to determine which methodology is the best fit for their organization. Different methodologies require different changes to the organization's management and software development cultures. Selecting the methodology that will bring the most benefits while requiring the fewest major cultural changes will greatly impact the methodology implementation.

There is a need for future longitudinal studies. Agile methodologies are too new to have a documented history of long-term methodology usage. The evolution of agile methodologies should also be studied and documented.

VII REFERENCES

[1] Boehm, B. & Turner, R. Management challenges to implement agile processes in traditional development organizations. *IEEE Software*. 22(5), 30-40. 2005.

[2] Theunissen, W., Boake, A., & Kourie, D. In search of the sweet spot: Agile open collaborative corporate software development. *Proceedings of the 2005 Annual Research Conference of the South African Institute of Computer Scientists and information Technologists on IT Research in Developing Countries*. White River, South Africa. 268-277.

[3] Lindstrom, L. & Jeffries, R. Extreme programming and agile software development methodologies. *Information Systems Management*. 21(13), 41-53. 2005.

[4] Holmstrom, H., Fitzgerald, B., Agerfalk, P., & Conchuir, E. Agile practices reduce distance in global software development. *Information Systems Development*. 23(3), 7-18. 2006.

[5] Highsmith, J. *Agile Software Development Ecosystems*. Addison-Wesley, Boston, MA, 2002.

[6] Jenkins] Jenkins, S.B. Musings of an "Old school programmer. *Communications of the ACM*. 49(5), 124-126. 2006.

[7] Astels, D., Miller, G, & Novak, M. *A Practical Guide to eXtreme Programming*. Upper Saddle River, NJ: Prentice Hall. 2002.

[8] Schatz, B. & Abdelshafi, I. Primavera gets agile: A successful transition to agile development. *IEEE Software*. 22(3). 2005

[9] Mann, C. & Maurer, F. A case study on the impact of scrum on overtime and customer satisfaction. *Proceedings of the Agile development Conference (ADC'05)*. Denver, CO. 70-79. 2005.

[10] Palmer, S. & Felsing, J. *A practical guide to feature-driven development.* Prentice Hall. Upper Saddle Hill River, NJ. 2002.

[11] Nunes, N., & Cunha, J. WISDOM – A UML based architecture for interactive systems. Lectures in Computer Science. 1946. 191-205. 2001.

[12] Nunes, N., & Cunha, J. WISDOM: A software engineering method for small software development companies. *IEEE Software*. 17(5), 113-119. 2000.

[13] Roberts, T., Gibson, M., Fields, K., and Rainer, R. Factors That Impact Implementing a System Development Methodology. *IEEE Transactions on Software Engineering*. 24(8), 640-649. 1998

[14] Software and Systems Process Improvement Networks. http://www.sei.cmu.edu/collaborating/spins/

[15] Williams, L. Extreme agility. *Web Techniques*, 7(1), 56. 2002

Jeffrey A. Livermore received his B.S. degree in psychology from Wayne State University in Detroit MI, his MSA degree in software engineering administration from Central Michigan University in Mt. Pleasant, MI, and his Ph.D. in information systems from Nova Southeastern University in Ft. Lauderdale, FL.

He is currently the Chair of Business Information Technology and Information Assurance at Walsh College in suburban Detroit, MI. Prior to working at Walsh College, he was the Director of District Technology in the Monroe Public Schools, the Chief Information Officer at the Barbara Ann Karmanos Cancer Institute, and a Staff Manager at American/SCI Inc.

Dr. Livermore is a member of the IEEE Computer Society, ACM, ISS, and HCTIA.