An Evaluation of Software Development Methodology Adoption by Software Developer in Sri Lanka

C.D. Manawadu[1], Md Gapar Md Johar[2], S.S.N. Perera[3]
[1] *Advanced Technology Center, Zone24x7 Private Limited, Colombo, Sri Lanka*
[2] *Faculty of Information Science & Engineering, Management and Science University, Selangor, Malaysia*
[3] *Department of Mathematics, University of Colombo, Colombo, Sri Lanka*

ABSTRACT:
Software development methodology usage and the adoption by software developers is an area which is crucial to understand. Currently available research in this area is limited and finding the evolution of software development methodology in the Sri Lankan context is vital for this booming industry. In this research the main objective is to find out how the evolution of software development methodology took place in history in Sri Lanka and what are the current methodologies adopted by the software developer in Sri Lanka. The research will be carried out by both qualitative and quantitative methods with the facilitation of the software development companies and software developers in Sri Lanka. The outcome of this research will be a stepping stone for future research on this area and the impact for future software development practitioners while adopting software development methodologies.

KEYWORDS: *Adoption, Methodologies, Software Developer, Software Development, , Sri Lanka, ,*

I. INTRODUCTION

Software development is a booming industry and it employs a massive number of knowledge workers in the current society. Evolution of this great industry has only taken few decades however it has influenced and revolutionized businesses more promisingly than the other industries in general. Looking back at the evolution of the software development industry, it can be said that the 1960s were boom years for entrepreneurial firms established to sell programming and system design skills under contract in a market where the rapidly expanding use of computers created a high demand for those skills. The first of these companies Computer Usage Corporation—was founded earlier, in 1955,(Kubie, 1994) but by the end of the 1960s, there were thousands of such firms. Most such companies were small, but a handful was large enough to go public, employing hundreds of programmers.(Campbell-Kelly, Winter 1995). In January 1967, International Computer Programs (ICP) in Indianapolis, Indiana, began publishing a quarterly catalog of computer programs available for sale, and the software product industry began to take shape. (Johnson, 2002).A major event that helped set the stage for the dramatic growth of software products in the 1970s and 1980s was IBM's decision to partially unbundle its software products in 1969. Emerson Pugh, a well-known computer historian, describes the "Origins of Software Bundling" at IBM. Then Watts Humphrey, a key software executive at IBM in the 1960s and later, covers his experiences in "Software Unbundling: A Personal Perspective," and Burton Grad provides "A Personal Recollection: IBM's Unbundling of Software and Services." (Grad & Johnson, 2002)

In today's context, according to Software Magazine, the top 500 companies in the computer software and services industry generated $640 billion dollars in revenue in 2012 and employed more than 4.1 million people who design, program, maintain, sell, or support computer software and services.(Desmond, 2012)
One key factor lies for the success of the software industry which is the software development methodology that binds the people, processes, technology and tools. Hence a software development methodology is an integral part of software development. It ensures quality is given prominence and provides a consistent and standard delivery approach for software within any software development team environment. Hence it facilitates multiple stakeholders of a software development project to be interconnected and move forward with accomplishing objectives as the software development activities evolves. Exploring into the Sri Lankan context of software development, the IT industry in Sri Lanka was established about one and a half to two decades ago, with a large number of small and medium enterprises. Current estimates place the total number of entities offering services

in the software development industries in Sri Lanka at over 200.(SLASSCOM, 2012)Current there is no literature that studies about the adaption of software development methodology in the Sri Lankan context. Further similar studies in the Asian region have been less. Hence this papers purpose is to prepare a detailed case study in the context of Sri Lankan Software Development Methodology adoption which is needed to be studied extensively.

Therefore the aim of this study and the driving Research Questions for this study are:

- How did the Software development methodology adoption occur in Sri Lanka
- What is the current methodologies adopted by Sri Lankan Software developers

Since a vast majority of the current studies on the use of software development methods highlight the experiences of US and the European organizations. In contrast, little is reported on how those methods are actually being practiced within a small Asian country, and how successful they have been. (Rahim, Seyal, & Rahman, 1998) Therefore this research is motivated by the knowledge gap in the literature on comprehensive studies that describe the adaption of software development methodologies in a country. The research relevance can be considered high because the main objective of software engineering is the development of high-quality, on-time, and within budget software projects, which can only be delivered with the utilization of a systematic development process, as has been proven in other engineering disciplines. Therefore, this study contributes to organize the diverse and partial views of software development methodology adoption. The remainder of the paper is structured as follows. In Section 2, we discuss the existing literature that is available with similar studies from other parts of the world. Section 3 describes the research methodology approach used in the paper and demonstrates how the research was conducted. Section 4 discusses in detail the results and in discussion mode the outcome of the study which would be beneficial to practioners and researchers. In Section 5 we concluded the paper with limitations and future work.

II. LITERATURE REVIEW

In the literature review, the research will look at some of the key software development methodologies including waterfall, spiral, prototyping, rapid application development, rational unified processes and agile. In here the research will look at their evolution and present some of the benefits and drawbacks each of these methodologies possesses. Hence through this background study, it will be helpful for the researcher to identify some of the most used software development methodologies and understands its evolution which will be helpful to directly map towards the Sri Lankan evolution and the adoption by the software developer.

Waterfall Methodology

The evolution of software development life cycles (SDLC) starts in the mid twentieth century. According to Boehm (1988) prior to the year 1956, the software was developed by Code & Fix technique which included two steps: step 1, write some code, and step 2, fix the problems of this code. However in 1956, the experience recognizes the problems with more large software development, and then, a model stagewise was originated. This model stipulates that the software should be developed in successive stages: operational plan, operational specifications, code specification, code parameters test, integration test, shakedown (test and implementation), and system evolution. (Boehm, 1988) Royce (1970) proposed the Waterfall methodology in order to avoid the difficult nature of "code and fix" approach. He proposed the construction of a prototype, and involvement of the users in several phases. The methodology was also proposed initially to deal with the increasing complexity of aerospace software, which included development of software packages for spacecraft mission planning, commanding and post-flight analysis. In various researches the Waterfall Model is classified as a traditional methodology (Dyck & Majchrzak, 2012).The central concept of the waterfall methodology is the integrated verification and validation of the results by the customer in order to complete a certain phase (Royce, 1970).Taking on a process-centered perspective, the supportive disciplines of the waterfall methodology are mainly quality management and documentation. (Dyck & Majchrzak, 2012) An underlying implication of the waterfall methodology is that both problem and solution can be fully described upfront (Agresti, 1986), further introduces the requirements, analysis and design phases before coding(Rodríguez-Martínez, Mora, & Alvarez, 2009). This tends the software development team to focus on clear and concise requirements before moving through to development and hence deal with unwanted rework. Discussing on the negative side of the waterfall methodology, researchers negatively emphasize on the waterfall methodology on extensive documentation (Royce, 1970). In this context, some sources criticize that testing activities also start too late and without any customer involvement (Hindel, H¨ormann, M¨uller, & Schmied, 2006).

Considering other process oriented characteristics, the waterfall methodology describes a sequential proceeding strategy also covering all phases from requirements to operations, while not explicitly covering maintenance and disassembly (Royce, 1970).Even though there are many criticisms for the Waterfall model, it has been the base for several subsequent models and the first to suggest the iteration and feedback issues; (Rodríguez-Martínez et al., 2009)

Spiral Methodology

Spiral methodology adds the risk analysis phase as a critical part of its definition, as well as the notion of an incremental development. (Rodríguez-Martínez et al., 2009). Much has been written about the value of risk reduction, especially as it is applied to software project management. A progressive approach to reduce risks for software development was defined by Barry Boehm. The approach involves working in incremental steps or phases to reduce software development risks. (Weckman, Colvin, Gaskins, & Mackulak, 1999).
In each phase of this spiral approach, the objectives and risks associated with those objectives are defined. Then the necessary sub-tasks or prototypes needed to resolve those risks are developed. Finally simulations and models are executed to verify that the objectives can be achieved. (Weckman et al., 1999)One difficulty was determining where the elaborated objectives, constraints, and alternatives come from. Another difficulty in applying the spiral model across an organization's various projects was that the organization has no common reference points for organizing its management procedures, cost and schedule estimates, and so on. This is because the cycles are risk driven, and each project has different risks. (Boehm, 1988)Quality is built into spiral model by means of activities involved at each phase, like risk analysis, prototype development, development plan, validation and verification, integration and acceptance testing. Each phase ensures that development is not moved to the next phase unless the previous phase is satisfied in terms of its activities. There is thorough analysis of requirements and risks even before the development starts, which guarantees that system contains only those requirements that are feasible and possible to implement. Furthermore, development phase of spiral performs step by step analysis of the product which ensures that no faults are escaped.(Boehm, 1988)

Prototyping Methodology

Prototyping suggest the development of an initial first operational version of the system in a quickly way (prototype), before to the full development. (Rodríguez-Martínez et al., 2009). The software industry has adopted this industrial technique to construct prototypes as models, simulations, or as partial implementations of systems and to use them for a variety of different purposes, e.g., to test the feasibility of certain technical aspects of a system, or as specification tools to determine user requirements.(Carr & Verner, 1997) Prototyping, on the other hand, can be viewed a 'process' (Floyd, 1984) which is either a well-defined phase within the software development life cycle, or is an 'approach' that influences the whole of it (Budde et al., 1992). The prototyping process can encourage the efficient development of applications by breaking a complex and often ill-defined problem into several comprehensive yet smaller and simpler parts (Kaushaar and Shirland, 1985). A prototyping development approach can help build, and subsequently refine, a product to meet end-user or market expectations.(Gomaa H., 1983).Researchers have also noted that prototyping enables us to partition the development process into smaller, easier to handle steps(Kaushaar and Shirland, 1985), is cost-effective (Boehm et al., 1984, Gordon and Bieman, 1994, Palvia and Nosek, 1990), improves communication between development personnel (Alavi, 1985) helps determine technical feasibility (Floyd,1984), is a good risk management technique (Tate and Verner, 1990) and results in greater user involvement and participation in the development process (Naumannand Jenkins, 1982).It would appear that the prototype process would not be appropriate when user needs are static or well-defined, or when development experience with similar applications has been extensive (Kraushaar & Shirland, 1985). Evolutionary prototyping, however, can lead to problems when performance is not adequately measured and either inefficient code is retained in the final product or the prototype demonstrates functionality that is unrealizable under normal usage loads. (Gordon & Bieman, 1995)

Rapid Application Development (RAD)

RAD proposes scenario-based analysis, the use of CASE tools and the component specification for maximum reuse (in the modern RAD concept). (Rodríguez-Martínez et al., 2009). James Martin coined the term RAD in the early 1990s to distinguish the methodology from the traditional waterfall model for systems development. "RAD refers to a development life cycle designed to give much faster development and higher quality results than the traditional life cycle. It is designed to take maximum advantage of powerful development software that has evolved recently." (Martin, 1991)While no universal definition of RAD exists, it can be characterized in two ways: as a methodology prescribing certain phases in software development (similar in principle to the spiral, iterative models of software construction), and as a class of tools that allow for speedy object development, graphical user interfaces, and reusable code for client/server applications. (Agarwal, Prasad, Tanniru, & Lynch, 2000) The essential characteristics of RAD tools include the capability for planning, data and process modeling, code generation, and testing and debugging. RAD methodologies encompass three-stage and four-stage cycles. The four-stage cycle consists of requirements planning, user design, construction, and cutover, while in the three-stage cycle, requirements planning and user design are consolidated into one iterative activity. (Agarwal et al., 2000). In a typical RAD life cycle, the requirements specification and design phases consume approximately 30% of the total effort (Martin, 1991)RAD lacks suitable representations for managing the co-operative systems development process currently. (Beynon-Davies & Holmes, 2002). RAD is accused of being anti-quality. It is commonly believed speed and quality are incompatible in software development. You can have one or the other but not both at the same time. On the other hand, supporters of RAD argue that modern quality principles are inherent in RAD. Fitness for purpose, avoiding waste, getting things right the first time, individual responsibility for quality, and meeting customer requirements is as engraved in RAD as in quality. What RAD tries to avoid is the bureaucratic nature of current quality control and assurance practice. (Howard, 2002). Other major concerns with RAD include the costs of maintaining clean rooms and of funding the extensive user involvement required for prototyping. Many organizations have found the culture changes required for RAD impossible to achieve either within the business as a whole or within project teams. (Howard, 2002)

Rational Unified Process (RUP)

RUP combines a two-dimensional (phases, workflows/activities) model with an iterative/incremental approach. (Rodríguez-Martínez et al., 2009). The Rational Unified Process is a Software Engineering Process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, (Kruchten, 1999) within a predictable schedule and budget.(Jacobson, Booch, & Rumbaugh, 1999)

The software lifecycle is broken into cycles, each cycle working on a new generation of the product. The Rational Unified Process divides one development cycle in four consecutive phases (Kruchten, 1996)

- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

Each phase is concluded with a well-defined milestone, a point in time at which certain critical decisions must be made, and therefore key goals must have been achieved (Boehm, 1996). The advantages of RUP have been the aesthetic clarity which makes the management point of RUP very easy, whereas the lacking details required supporting Software Engineers lacking is the key issue for its negative feedback over the years. (Hull, Taylor, Hanna, & Millar, 2002)

Agile Methodology

Agile methodologies are a new host of methodologies that claim to overcome the limitations of traditional plan-driven SDMs. The “Agile Manifesto” published by a group of software practitioners outlines the principles of agile systems development(Chan & Thong, 2009). In short, these principles emphasize the importance of individuals and their interactions, customer collaboration, early and continuous delivery of software, and the capability to respond to volatile requirements. Examples of agile methodologies that align with the Agile Manifesto include Extreme Programming (XP), Crystal methods, Lean Development, Scrum, and Adaptive Software Development(Highsmith, 2002) Highsmith (2002)suggested that the differences between traditional SDMs and agile methodologies rest on two assumptions about customers. First, traditional SDMs assume that customers do not know their requirements but developers do, whereas agile methodologies assume that both customers and developers do not have full knowledge of system requirements at the beginning (Highsmith, 2002). Second, traditional SDMs assume customers are short-sighted, and thus developers have to build in extra functionalities to meet the future needs of customers, often leading to over designed systems (Highsmith, 2002). On the other hand, agile methodologies emphasize simplicity—the art of maximizing the work not done(Lindstrom & Jeffries, 2004). Also, the differences in philosophy between traditional SDMs and agile methodologies lead to differences in a number of practices and requirements, such as planning and control, role assignment among developers, customer's role, and technology used (Nerur, Mahapatra, & Mangalaraj, 2005).In 2001, the ‘‘agile manifesto” was written by the practitioners who proposed many of the agile development methods. The manifesto states that agile development should focus on four core values1:(Dyba & Dingsøyr, 2008)

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

Most studies reported that agile development practices are easy to adopt and work well. Benefits were reported in the following areas: customer collaboration, work processes for handling defects, learning in pair programming, thinking ahead for management, focusing on current work for engineers, and estimation. (Dyba & Dingsøyr, 2008) With respect to limitations, the lean development technique did not work well for one of the teams trying it out, pair programming was seen as inefficient, and some claimed that XP works best with experienced development teams. A further limitation that was reported by one of the studies, which has also been repeatedly mentioned in the literature (McBreen, 2002), was the lack of attention to design and architectural issues.(Stephens & Rosenberg, 2003)

III. RESEARCH METHODOLOGY

This research is an exploratory study. The research adopted a hybrid approach mainly based on quantitative research methodologies which slightly combined with qualitative research methodologies. As this is a relatively new research domain with insufficient academic research contribution, we had to consult the online articles, success stories, case studies and personal experiences shared by practitioners in the industry in Sri Lanka. To serve the need of quantitative research, a web based survey comprising of questionnaire was designed to approach practitioners who have experience working in software development methodologies in Sri Lanka.
The questionnaire was sent through emails, community groups, and discussion boards to a large pool of practitioners. The data was collected over a period of four months from June, 2013 to October, 2013. The overall relevant response rate was 41% due to specific and technical nature of the survey. The questionnaire was open for all to access. We scanned data of 51 respondents, which were found more relevant and complete for evaluation of the results. Microsoft excel and SPSS were mostly used to evaluate the respondents' results. The questionnaire was divided into four parts. It contains total 28 close and open ended questions to capture the views of IT professionals. In the first section, we asked the demographic information of respondents, while the second section captured the particulars of the respondent's company. The third section was related to the organizations and individuals current & past experience of software development methodologies usage and some of the rankings related to effectiveness of using specific methodologies listed. Section 4 was designed to capture the perception of the future usage of software development methodology in the organization and at individual level. The responses were quite diverse according to the professional experience of the respondents. In order to become more focused, we identified the respondents which were specific and practically working in the particular roles (Software Engineers, Software Architects, QA Engineers and Project Managers). Informal interviews were also conducted to refine the respondent findings. The results presented here are formulated by compiling the literature reviews, the survey and informal interviews conducted. Descriptive statistics (mean and standard deviation) were used to analyze scores and to derive gaps of the current effective of software development methodology usage. Further similar analysis was used to derive results of future preferred methodology adoption. Percentage and graph based analysis was used to summarize and present the other findings of the research.

IV. RESULTS AND DISCUSSION

This section summarizes research findings and results.

A. Particulars of Respondents

First section of research questionnaire was designed to obtain the respondents' demography. It included very general questions such as respondents' professional experience and current position etc. The respondent sample was quite diverse, that's why we got variation in their opinions and observations. Over here we have presented the averaged results of our findings. Altogether we got 51 responses. These were covering around 30 unique software development R & D companies spread across Sri Lanka. All most every company was attached to a global software development. From the respondents, 8% respondents had professional experience over 9 years, 32% respondents had professional experience between 4 - 9 years, 52% respondents had professional experience between 1 - 4 years and 8% respondents had less than 1 year of experience altogether. The diversity in professional expertise enabled us to get a better understanding of the situation. Participation of respondents on the basis their current position is shown in the table 1.

Table 1 Respondent's Professional Division

Designation	**Percentage %**
Software Engineers	60
Software Quality Assurance Engineers	8
Software Architects	16
Project Managers	12
Other	4

B. Particulars of Respondents' Companies

Second section collects the information related to respondents' companies in which they were working at the time of survey conducted. Questions related to respondents' companies e.g. no of employees and types of solutions developed were gathered. This section analyzes the information related to respondents' IT companies' software development practices. Respondents worked with companies of different size which varies from Less than 5 employees to more than 150 employees working companies. 44% of the respondents worked in companies comprising more than 150 employees. 40% respondents worked in companies having employees between 75 to 150. 8% of the respondents worked in companies having employees between 25 - 75. The remaining 8% respondents fell into the category of having employees below 25 in their companies. These companies fall into the falling types of solutions that were developed in-house. E-commerce/cloud-based/web applications (e.g. Web portals), Mobile Application Development(e.g. Android Apps), Real time applications (e.g. process control, manufacturing), Management information systems (e.g. decision support), Transaction processing systems (e.g. payroll, POS, accounting, inventory), Embedded systems (e.g. software running in consumer devices or vehicles), Systems software (e.g. telecommunications software), Interactive Multimedia Applications (e.g. Games), Legacy Transformation Systems (e.g. .Net porting). Figure 1 depicts the distribution of solutions developed by these companies.

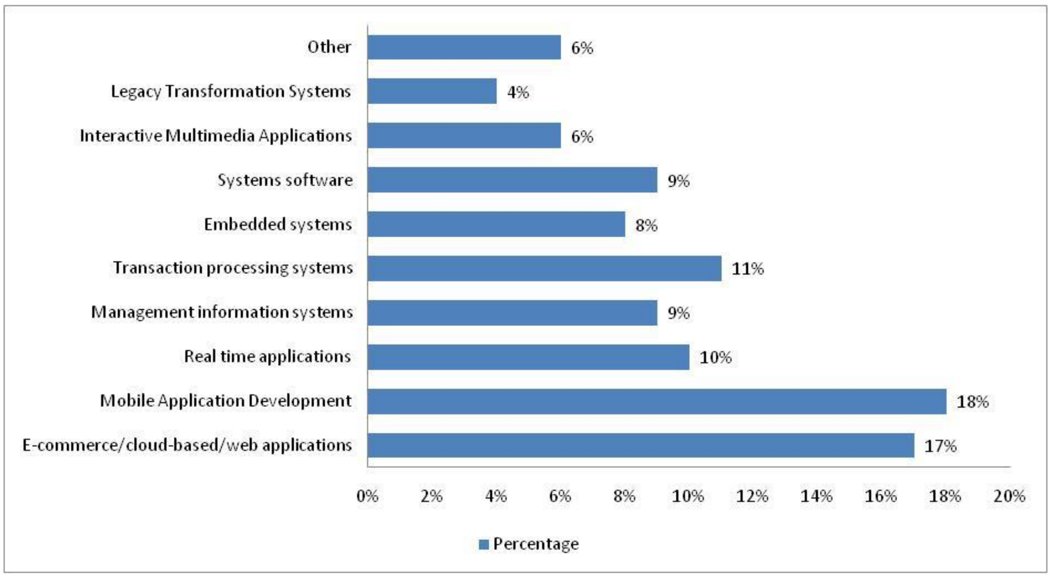


Figure 1 : Solutions Developed

C. Current and past analysis of Software Development Methodology usage

Third section of the questionnaire collected the current and past usage analysis of software development methodologies and rating of effectiveness of each methodology used currently were gathered.
Out of the 51 respondents, 100% of the respondents confirmed that their organizations use a software development methodology. When questioned on the current methodologies used in the companies projects as an overall indicator, 52.17% of the participants stated using agile, while the rest of the distribution was spread among RAD, prototyping, RUP, spiral and waterfall methodologies. Statistics are illustrated in figure 2.

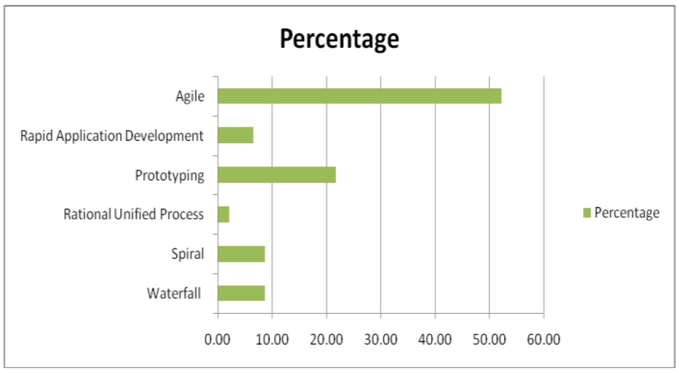


Figure 2 : Current usage of software development methodologies

The survey also gathered the mostly used software development methodology in the company. According to the results 68% of the participants indicated it was Agile and the next mostly used methodology was named as waterfall which was 21% of the total percentage. When inquired on the opinion on why it is used mostly in your organizations projects? 43% of the respondents stated that is most practical and effective, 29% stated its client specific, 18% According to company policy, 7% percent stated its less defects and re-work and 4% stated its suitable for outsourcing projects.

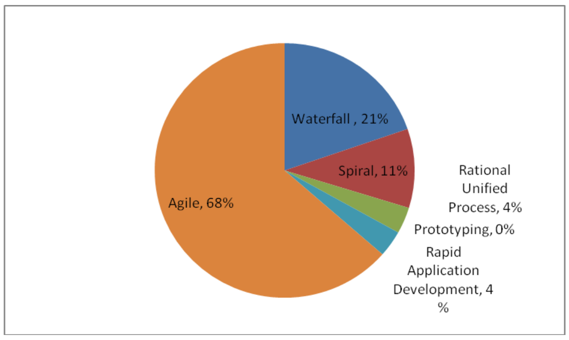


Figure 3 : Mostly used software development methodology

When we asked the question of what is the software development methodology used in your first project, as depicted in figure 4 above, 52% of the respondents stated it was waterfall, 24% stated it was prototyping, and the rest stated they used Spiral, RAD and Agile in a distribution of 8 percentage each respectively, which working on their first project. While we inquired on the reason, most of the respondents mentioned that it was due to organizations policy and client demand at that time. A considerable percentage mentioned it is due to only available option and the most effective at that time. Another percentage mentioned knew nothing about the other methods.

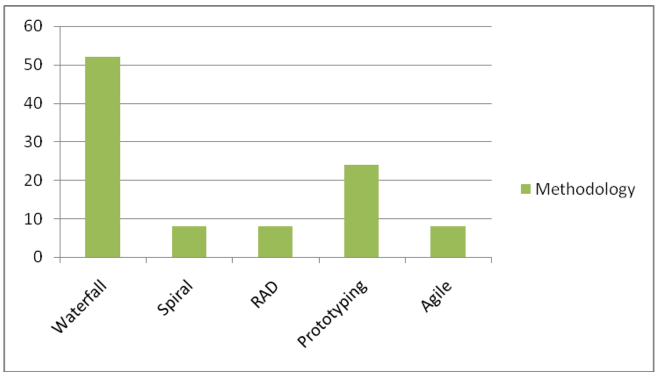
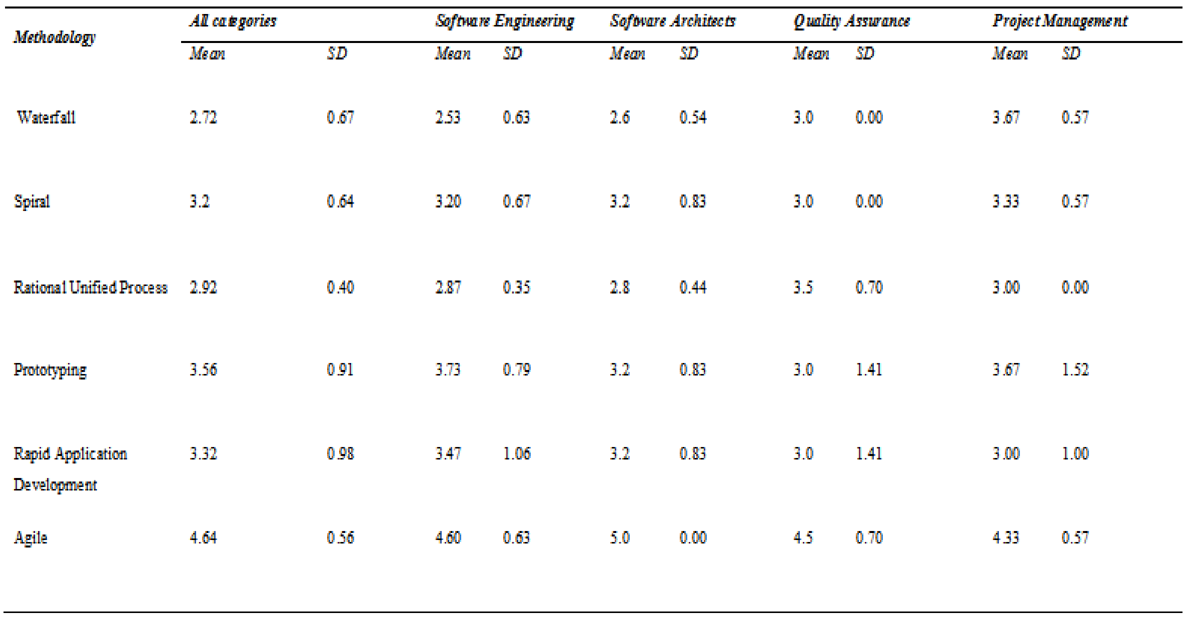


Figure 4 : Methodology used in first project?

To analyze the effectives of these methodologies being used, a five-point Likert scale was used to establish the level of effectiveness of current methodologies. Out of the six general methodologies used which were Waterfall, Spiral, RUP, Prototyping, RAD and Agile, the results are summarized in Table 2 below.
Out of the analysis below, it clearly draws out the Agile methodology as the most effective methodology as per current status among the participants. While we interviewed few individuals for their opinion on this results, they agreed on the view of the results expressed, however considering the compared methodologies in place. However they also mentioned there are limitations that were existing even though the majority of practioners have listed this one out. Further we clearly see that there is a high standard deviation of data analysis for Prototyping and RAD methodologies. In all categories, this reflects, hence we feel that the current usage of these methodologies may differ from each respondents experiences. Also we feel due to this high variance, these two methodologies have not been consistently used within the development scope of projects in Sri Lanka.
Considering the lowest effective results that have been listed out, we feel that waterfall methodology has been the least effective with the rates identified below. However with a slight difference where the Project Managers has slightly preferred it. This may be due to the nature of project management compared with Waterfall.
Looking into the Software Engineers result outcomes, it clearly indicates that the current developers prefer the agile context. While interviewing few of the Software Engineers, we gathered that the adoption has occurred from traditional based methods such as waterfall to prototype and then from it to agile practice. Most of these developers has indicated that they have used waterfall method at the initial stages of their careers. Looking at their experience profiles, during the past decade of working in the industry, most of them have used waterfall as the main methodology. Therefore we can clearly identify that Sri Lankan software R & D companies have approached this method at a early stage of development.

One clear reason for moving from these standard methodologies to more iterative based once has been the fact that the traditional once have always caused re-work and client dissatisfaction over quality on the longer run. Further we gathered that most of the projects had not been completed on time using the traditional methods. Always there had been project estimation slippages due to client not been part of the requirements engineering initially. As per the literature review on the methodologies presented, the drawbacks listed out in section 2 has a very direct relation to most of the comments shared by the interviewed software engineers. Mainly with comparison of the results depicted in table 2, the preference for Prototype has also been remarkably higher from the Software Engineers aspects. This is also a indication that iterative work is much preferred by developers in the current context rather than finishing large phases of work which would be ineffective for better productive results.

Table 2. Effectiveness of methodologies used



Methodology	*All categories*		*Software Engineering*		*Software Architects*		*Quality Assurance*		*Project Management*	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*
Waterfall	2.72	0.67	2.53	0.63	2.6	0.54	3.0	0.00	3.67	0.57
Spiral	3.2	0.64	3.20	0.67	3.2	0.83	3.0	0.00	3.33	0.57
Rational Unified Process	2.92	0.40	2.87	0.35	2.8	0.44	3.5	0.70	3.00	0.00
Prototyping	3.56	0.91	3.73	0.79	3.2	0.83	3.0	1.41	3.67	1.52
Rapid Application Development	3.32	0.98	3.47	1.06	3.2	0.83	3.0	1.41	3.00	1.00
Agile	4.64	0.56	4.60	0.63	5.0	0.00	4.5	0.70	4.33	0.57

D. Future effectiveness of Software Development Methodology usage

Table 3. Future Effectiveness of SDM

	All categories		*Software Engineering*		*Software Architects*		*Quality Assurance*		*Project Management*	
Methodology	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*	*Mean*	*SD*
Waterfall	2.72	0.84	2.67	0.82	2.0	0.82	3.0	0.00	3.67	0.58
Spiral	3.12	0.83	3.27	0.70	2.50	1.29	3.00	0.00	3.00	1.00
Rational Unified Process	2.92	0.64	3.00	0.53	2.25	0.96	3.50	0.71	3.00	0.00
Rapid Application Development	3.36	0.99	3.6	0.91	2.50	1.29	3.50	0.71	3.00	1.00
Scrum	4.64	0.49	4.67	0.49	4.75	0.50	4.50	0.71	4.33	0.58
Extreme Programming (XP)	4.16	0.75	4.07	0.80	4.25	0.50	4.00	1.41	4.33	0.58
Crystal Methods	3.20	0.58	3.13	0.52	3.00	0.00	4.00	1.41	3.33	0.58
Feature Driven Development (FDD)	3.00	1.47	2.87	1.46	2.50	1.73	4.00	0.00	4.33	0.58
Dynamic Systems Development Methodology (DSDM)	3.36	0.64	3.33	0.62	3.00	0.00	3.50	0.71	4.00	1.00
Adaptive Software Development (ASD)	3.36	0.64	3.40	0.74	3.00	0.58	3.50	0.71	3.67	0.58

Another analysis that was carried out to determine the future effectiveness of using specific software development methodologies. The participants were asked to rate the effectiveness using a five-point Likert scale. We added few of the agile methodologies and we dropped out the prototyping methodology as we thought it wouldn't be effectively used in the future with the diverse use of agile methods. We added Scrum, Extreme Programming (XP), Crystal Methods, Feature Driven Development (FDD), Dynamic Systems Development Methodology (DSDM), Adaptive Software Development (ASD) since the literature on SDM usage indicated to us that the future would be more relevant for these methodologies. Apart from that we also validated Waterfall, Spiral, RUP, and RAD, which the detailed results are summarized in Table 3 above.

The analysis gave use clear insights that the Agile methods are the most effective in terms of usage for future software development projects. Out of our sample participants, 4.64 and 4.16 mean values were retrieved for Scrum and XP agile methodologies respectively for the five-point Likert scale questions we raised on effectiveness. However Scrum methodology had the least standard deviation of 0.49 as a overall level, and further the same level of mean and standard deviations were shown comparing results from Software Engineers, Architects, QA's and Project Managers. Thus it is evident that Scrum is a popular agile methodology which is emerging within the Sri Lankan software development context and the likelihood of companies and developers using and preferring it as a methodology is viable.

However the effectiveness of XP was also similar, but it changed slightly while analysing the results with non-software engineering practioners. Through the literature and knowledge gained, XP is far more superior for Software Engineers and its core principles lie on coding. hence that could have been the reason for Software Engineers to prefer it for future usage and other not having the same opinion.

Out of the ten methodologies analyzed the least favorite for the future is the Waterfall methodology. Almost all the categories of roles has evaluated it as a least effective, other than a slight preference shown by the Project Managers. We feel that some of the Project Managers would have seen it effective as their discipline would have been skillfully mastered while using the methodology.

We also like to emphasize over here that Feature Driven Development (FDD) showed promise, but it has a very high standard deviation among all categories. Thus indicating to us that it is a methodology that some of the developers are aware and maybe some developers have not used its benefits in real practice.

Thus again with the analyze it denotes that the Sri Lankan software developer and related discipline community feels that the adoption of the software development methodology in the Sri Lankan perspective goes mainly to the agile methodologies which are in use. Hence in comparison to the changes of usage in the world Sri Lanka is also changing similarly as per our analysis.

V. CONCLUSION

The research was a preliminary study of a larger research that is been carried out currently. Hence the main goal of the research was to understand the current usage of software development mythologies used by the software developers in Sri Lanka. Also to understand how the methodology usage has evolved during the past decade. We used both quantitative and qualitative methods to carry out our research and we had selected participants from Sri Lankan software industry for this research. The findings of the research was that many organizations and developers in the current context use Agile methodologies for the software development. Currently they are satisfied with its attributes and further many of them recommend its effectiveness for future use as well. Therefore we can clearly say that the Sri Lankan developer community in parallel with what is currently happening the words are more aligned. Further the findings of how the development methodology evolved during past years was gathered. Most of the past decade development activities has triggered with the waterfall and traditional methodologies and now it has slowly evolved into agile methodologies. However the surprising and promising factor is that most of the developers and organizations use at least one of the identified methodologies to perform software development. Discussing on the limitations of this study, it could be said that if the sample of this study was increased it would have given a more better outcome. Hence to provide further research into this area, we could propose that methodology specific study comparing effectiveness to the Sri Lankan context be studies in order to support future practitioners in determining which methodology they could use in which circumstances.

REFERENCES

[1] Agarwal, R., Prasad, J., Tanniru, M., & Lynch, J. (2000). Risks of rapid application development. Commun. ACM, 43(11es), 1. doi: 10.1145/352515.352516

[2] Agresti, W. W. (1986). The conventional software life-cycle model: Its evolution and assumptions. New Paradigms for Software Development. IEEE CS, 2-5.

[3] Alavi, M. (1985) An Assessment of the Prototyping Approach to Systems, Communications of the ACM, 27, 6, 556-564.

[4] Beynon-Davies, P., & Holmes, S. (2002). Design breakdowns, scenarios and rapid application development. Information and Software Technology, 44(10), 579-592. doi: http://dx.doi.org/10.1016/S0950-5849(02)00078-2

[5] Boehm, B.W., Gray, T.E. and Seewaldt, T. (1984) Prototyping Versus Specifying: A Multiproject Experiment, IEEE Transactions on Software Engineering, SE-10, 4, 290 -402.

[6] Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. IEEE Computer.

[7] Boehm, B. (1996). Anchoring the Software Process. IEEE Softw., 13(4), 73-82. doi: 10.1109/52.526834

[8] Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., & Madachy, R. (1998). Using the WinWin Spiral Model: A Case Study. Computer, 31(7), 33-44. doi: 10.1109/2.689675

[9] Budde, R., Kautz, K., Kuhlenkamp K., Zullighoven, H. (1992) What is Prototyping?" Information Technology & People, 6, 2-4, 89-95.

[10] Campbell-Kelly, M. (Winter 1995). Development and Structure of the International Software Industry, 1950–1990. Business and Economic History, 24(2), 84-85.

[11] Carr, M., & Verner, D. J. (1997). Prototyping and Software Development Approaches.

[12] Chan, F. K. Y., & Thong, J. Y. L. (2009). Acceptance of agile methodologies: A critical review and conceptual framework. Decis. Support Syst., 46(4), 803-814. doi: 10.1016/j.dss.2008.11.009

[13] Desmond, J. P. (2012). Software 500: Revenue Up 17 Percent. Software Magazine. Retrieved from http://softwaremag.com/content/ContentCT.asp?P=3374 website:

[14] Dyba, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Inf. Softw. Technol., 50(9-10), 833-859. doi: 10.1016/j.infsof.2008.01.006

[15] Dyck, S., & Majchrzak, T. A. (2012). Identifying Common Characteristics in Fundamental, Integrated, and Agile Software Development Methodologies. Paper presented at the Proceedings of the 2012 45th Hawaii International Conference on System Sciences.

[16] Floyd, C. (1984) A Systematic Look at Prototyping, in: Budde, R.,Kuhlenkamp, K., Mathiassen, L. and Zullighoven, H. (Eds.) Approaches to Prototyping, Springer-Verlag: Heidelberg, 1-17.

[17] Gomaa, H. (1983) The Impact of Rapid Prototyping on Specifying User Requirements, ACM SIGSOFT Software Engineering Notes 8, 2, 17-28.

[18] Gordon, V.S. and Bieman, J.M. (1994) Rapid Prototyping: Lessons Learned, IEEE Software, 12, 1, 85-95

[19] Grad, B., & Johnson, L. (2002). The Start of the Software Products Industry. IEEE Ann. Hist. Comput., 24(1), 3-4. doi: 10.1109/mahc.2002.988573

[20] Highsmith, J. (2002). Agile software development ecosystems: Addison-Wesley Longman Publishing Co., Inc.

[21] Hindel, B., H¨ormann, K., M¨uller, M., & Schmied, J. (2006). Basiswissen Software-Projektmanagement - Aus- und Weiterbildung zum Certified Professional for Project Management nach iSQI. Paper presented at the Aus- und Weiterbildung zum Certified Professional for Project Management nach iSQI-Standard.

[22] Howard, A. (2002). Rapid Application Development: Rough and Dirty or Value-for-Money Engineering? COMMUNICATIONS OF THE ACM, 45.

[23] Hull, M. E. C., Taylor, P. S., Hanna, J. R. P., & Millar, R. J. (2002). Software development processes — an assessment. Information and Software Technology, 44(1), 1-12. doi: http://dx.doi.org/10.1016/S0950-5849(01)00158-6

[24] Jacobson, I., Booch, G., & Rumbaugh, J. (1999). The unified software development process: Addison-Wesley Longman Publishing Co., Inc.

[25] Johnson, L. (2002). Creating the Software Industry: Recollections of Software Company Founders of the 1960s. IEEE Ann. Hist. Comput., 24(1), 14-42. doi: 10.1109/85.988576

[26] Kraushaar, J. K., & Shirland, L. E. (1985). A prototyping method for applications development by end users and information systems specialists. MIS Q., 9(3), 189-197. doi: 10.2307/248948

[27] Kruchten, P. (1996). A Rational Development Process, CrossTalk. Hill AFB, UT: STSC.

[28] Kruchten, P. (1999). The Rational Unified Process: an introduction: Addison-Wesley Longman Publishing Co., Inc.

[29] Kubie, E. C. (1994). Recollections of the first software company. IEEE Ann. Hist. Comput., 16(2), 65-71. doi: 10.1109/85.279238

[30] Lindstrom, L., & Jeffries, R. (2004). Extreme programming and agile software development methodologies. Information Systems Management 21(3), 41–60.

[31] Martin, J. (1991). Rapid Application Development. New York: Macmillan.

[32] McBreen, P. (2002). Questioning Extreme Programming: Addison-Wesley Longman Publishing Co., Inc.

[33] Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. Commun. ACM, 48(5), 72-78. doi: 10.1145/1060710.1060712

[34] Nauman, J.D. and Jenkins, M. (1982) Prototyping: The New Paradigm forSystems Development, MIS Quarterly, 6, 3, 29-44.

[35] Palvia, P. and Nosek, J.T. (1990) An Empirical Evaluation of System Development Methodologies, Information Resources Management Journal, 3,23-32.

[36] Rahim, M. M., Seyal, A. H., & Rahman, M. N. A. (1998). Use of software systems development methods An empirical study in Brunei Darussalam. Information and Software Technology, 39(14–15), 949-963. doi: http://dx.doi.org/10.1016/S0950-5849(97)00052-9

[37] Rodríguez-Martínez, L. C., Mora, M., & Alvarez, F. J. (2009). A Descriptive/Comparative Study of the Evolution of Process Models of Software Development Life Cycles (PM-SDLCs). 298-303. doi: 10.1109/enc.2009.45

[38] Royce, W. W. (1970). Managing the Development of Large Software Systems. Proceedings of the IEEE WESCON, 1-9.

[39] SLASSCOM. (2012). Sri Lanka Advantage - The Sri Lankan IT-BPO Industry Retrieved 02/21/2013, 2013

[40] Stephens, M., & Rosenberg, D. (2003). Extreme Programming Refactored: The Case Against XP: APress L. P.

[41] Verner, J., Tate, G. and Cerpa, N. (1995) Prototyping: Some New Results,Information and Software Technology, 38, 12, 743-755.

[42] Weckman, J., Colvin, T., Gaskins, R. J., & Mackulak, G. T. (1999). Application of simulation and the Boehm spiral model to 300-mm logistics system risk reduction. Paper presented at the Proceedings of the 31st conference on Winter simulation: Simulation---a bridge to the future - Volume 1, Phoenix, Arizona, USA.