2006:PM08

SOFTWARE DEVELOPMENT THROUGH AGILE WAY

PRASANTH DUMPALA

EXAMENSARBETE
Magister Med Inriktning Mot Programvaruteknik Institutionen För Teknik, Matematik Och Datavetenskap

EXAMENSARBETE

SOFTWARE DEVELOPMENT THROUGH AGILE WAY

Sammanfattning

Ingen svensk sammanfattning finns då denna uppsats skrivits av en engelskspråkig student. Se 'ABSTRACT' for mer detaljer(på engelska).

Författare: PRASANTH DUMPALA
Examinator: Dr. Samantha Jenkins
Handledare: Dr. Steven R. Kirk

Program: Masters in Focus with Software Engineering, 2006
Ämne: software engineering **Examensnivå**: MASTER
Datum: 2006-08-18 **Rapportnummer**: 2006: PM08
Nyckelord: Agile Methods, XP, SCRUM, PSP, TSP, CMM, Traditional Methods, Reuse and Agile Documentation

Utgivare: Högskolan Väst, Institutionen för teknik, matematik och datavetenskap,
461 86 Trollhättan
Tel: 0520-22 30 00 Fax: 0520-22 32 99 Web: www.hv.se

MASTER'S THESIS

SOFTWARE DEVELOPMENT THROUGH AGILE WAY

Summary

This research paper discusses the software development through agile approaches. In which we see that most of the time, agile approaches cannot use for all type of projects. But this paper is exploring the new ideas that how we can use the agile methodologies with the help of traditional approach and PSP/TSP/Software CMMI. So, the over all paper is giving the whole idea, that we can develop any type of software projects through the new agile methodology guidelines which we are describing here. As well as for solving this paper we adopted different ways in which we focused on literature study & also published research paper on agile methodology. In addition, we did this work on the base of hybrid ideology where we combined the best ways of different approaches and achieved our goals.. Where we proved that by the combination of these approaches, we can manage our projects as well as team personals in the organization successfully. And this concept for team management comes from PSP/TSP and Software CMMI. The benefit for management of this is also to get our projects with agility and personnel's training for project management.

Keywords: XP and Scrum, PSP/TSP and CMMI, Traditional Approach/Waterfall Model, Current Status of Agile Software Development, The Source Code with Models, and Documents, The Effective Documentation Handoffs, The Agile Work Environment.

Författare: PRASANTH DUMPALA
Examinator: Dr. Samantha Jenkins
Handledare: Dr. Steven R. Kirk

Program: Masters in Focus with Software Engineering, 2006
Ämne: software engineering **Examensnivå**: MASTER
Datum: 2006-08-18 **Rapportnummer**: 2006: PM08
Nyckelord: Agile Methods, XP, SCRUM, PSP, TSP, CMM, Traditional Methods, Reuse and Agile Documentation

Utgivare: Högskolan Väst, Institutionen för teknik, matematik och datavetenskap,
461 86 Trollhättan
Tel: 0520-22 30 00 Fax: 0520-22 32 99 Web: www.hv.se

Acknowledgments

I choose this research paper as part of my Masters of Software Engineering program. The successful completion of this paper involves dedication, intense strictly and research work.

I also owe a debt of gratitude to the University staff. This paper stands completed virtually as a result of teamwork and all the attributes already spelled out above. For us it has been a painful, yet satisfying experience, and the output of our dedicated efforts leaves me sufficiently motivated to involve me more in this research paper and complete it with a touch of excellence.

I placed our record & appreciation to my university West Sweden University and for those personals who helped me a lot in which Dr. Steven Kirk (Advisor), Email: steven.kirk@hv.se and Dr. Samantha Jenkins (Examiner & Assoc. Prof., Department of Technology, Mathematics & CS), Email: Samantha.Jenkins@hv.se.

Innehållsförteckning

1 Introduction

Agile methods are particularly for teams in organization facing unpredictable or rapidly changing requirements. These Agile methods are known to have many problems like culture, people and communication [1]. Some may be at method level (life-cycle of agile methods) and others at the higher level like communication between people, controlling the project and quality of the project. The contradiction of any such nature breeds the idea of making the ideas of software unified process difficult to implement at project.

Software projects escalate in size out of proportion due to this circumstantial dilemma. This becomes much more troublesome when the software processes are not in order. Any method can go out of shape. Awkwardness of method carries all problems as a project of its own and the entire headache is real in the software. Many risks are there to culminate everything which is already there. All of the processes are not badly done but the whole project is there and suffering is too bad and this becomes a very dilemmatic and very dangerous for the whole lifespan of project. Some of the well known good companies are using the agile development methods are given below:

1) Loral Vought Systems [http://www.vought.com/heritage/years/html/93-03.html]
2) Marlow [http://www.marlow.com]
3) Northrop Grumman Commercial Aircraft division [http://www.dsd.es.northropgrumman.com/index.html]
4) Texas Instrument [http://education.ti.com/educationportal/sites/US/homePage/index.html]
5) Tracor, Inc [http://www.tracor.com]
6) Center for Collaborative Manufacturing [http://www.techcollaborative.org]
7) E-systems [http://www.esystemsinc.com]

A web site Escrow.com analyzed the market which is using the agile development and they documented the results of their success with agile development methods which are given below in the table 1:

	Before Adopting Agile	**After Adopting Agile**	**% Change**
Total Code Size	45,773	15,048	-67%
Average Methods Per Class	630	10.95	+73%
Average Lines Per Method	11.36	5.86	-48%
Average Cyclometric Complexity	3.44	1.56	-54%

Table 1: Showing the success in percentage with agile development methods [2].

Table 1: shows the software development percentages, before and after adopting agility. In which some of the companies adopted the agility and they got improvement in software development.

This paper is about a new idea in the software development by the combination or to suggest a 'hybrid' of existing agile methods which will be helpful for the every kind of software development through agile methodology. So, this paper is also discussing and focusing on the different problems of the agile methodologies and suggesting a new method.

We divided this paper into different sections in which the section 2 which is background and talking about the agile software development methods with management, XP/Scrum (Ref 2.1.1 and 2.1.2) and traditional approach(Ref 2.2) with their working as well as advantages/disadvantages. This section is also discussing on the methods from U.S software Engineering Institute where we explained about Team Software Process (TSP) / Personal Software Process (PSP) and Capability Maturity Model (CMMI) which they are offering in section 2.3.

Section 3 is about methodologies where we followed that which way we adopted for solving this paper.

Section 4 is about Implementation which divided into different subsections. Overall in this section we talked about how to make documentation with agility and what are the effective documentation hands off? Also there are some guidelines for effective documentation as well as about the management customers and the entire whole situation here. That how the different people are working in the agile environment? Also here is some concept of reusability in agile software development like components reusability for getting work as soon as possible. The strengths of agility shows the whole processes which we used and we got our results. In the end we derived some results about the XP/ Scrum/Traditional approach with PSP/TSP/Software CMMI as well as agile development characteristics.

Section 5 we made results for satisfying our goal. And also we proved that software development for all small and large can be done by using the help of our proposed agile methodology.

Section 6, we conclude hybrid approach in which the concept of combination of different approaches used and also which can be used in all type of projects.

All this work done with the help of our adopted methodologies so if some one wants to do research in this area, the way of searching relevant material will come from the current researches which you can find from authorized organizations like IEEE standards or other European standards.

2 Background

In the February 2001, a group was tired with the existing heavy software methodologies met in Utah to find some common ground in alternate software development. After some conversations they came up on a new methodology and they said that we are uncovering better ways of developing software by doing it and helping others do it. And then they decided to explain the way which is given below [3]:

1) The Individuals and interactions over processes and tools
2) The working software over comprehensive documentation
3) The customer collaboration over contract negotiation
4) The Responding to change over following a plan

Before going in the detail of this research paper we should know about agile. And also about some common used agile methodologies in which most common are the extreme programming (XP) and Scrum as described in section 2.1.1 and 2.1.2. Basically the meaning of agile word is showing the fast means something we have to do but in a fast way. So, we have to plan this that how we can develop the software, speedily/agility. The above said four points are the cornerstone of all the

different Agile Software Development Methods and Management which is discussing in section 2.2 (given below).

2.1 The Agile Software Development Methods with Management

In the agile software development methods with management is also very important nowadays. And it's one of a growing number of alternatives to traditional, process-centric software management methods [9]. This is also focusing on people, results, with minimal methods and maximum collaboration. It is geared to the high speed and high change of today's ebusiness projects. (Highsmith, 2000).

2.1.1 Extreme Programming

eXtreme Programming is an approach of agile methodologies for software management which takes a code-centric view of the activity [4]. XP offers several compelling features in which comprehensive unit tests, short release cycles, adding only what's needed for the current task, collective code ownership, continual improvement and also the adding of features in the order of importance [5]. The development environment in an organization which uses XP is generally characterized by some procedures. Which are given below [5,6]:

1) Software should provide the list of features which was given by the customer.
2) Programmers should break the features into stand-alone tasks and after this estimate the work which is necessary to complete those tasks.
3) It is necessary that customer chooses the most important tasks that can be completed by the next release
4) The programmers can choose their tasks, and work in pairs for getting the required output.
5) Programmers write unit tests
6) Programmers add features to pass unit tests
7) Programmers fix features/tests as necessary, until all tests pass
8) Integrate the code which programmers can do like integration of the tasks and then the programmers can produce a released version
9) Customer runs acceptance tests and after this software version goes into production
10) Programmers update their estimates based on the amount of work they've done in release cycle

On the other hand there are some disadvantages which are given below:

1. Code rather than design-centered development. Design practices might not be serious for small programs, it can be disastrous [7] when programs are larger than a few thousand lines of code or the work involves more than a few people.
2. Lack of design documentation. Limits XP to small programs and makes it difficult to take advantage of reuse opportunities.
3. Lack of a quality plan: Where it has been found with the TSP that quality planning helps properly trained teams produce high-quality products [8], and it reduces test time by as much as 90%. XP does not explicitly plan, measure, or manage program quality.
4. The XP method provides essentially no data-gathering guidance.

2.1.2 Scrum

Another agile methodology is Scrum which is also a most popular method. Scrum is an iterative, incremental process for developing any product or managing any work. Scrum produces a

potentially shippable set of functionality at the end of each iteration. The attributes of the scrum which are giving the idea about scrum are given below [3, 10]:

1) It's an agile process to manage and control development work.
2) It's a wrapper for existing engineering practices.
3) It's a team-based approach to iteratively, incrementally develop systems and products when requirements are rapidly changing
4) It's a process that controls the confusion of conflicting interests and needs.
5) It's a way to improve communications and maximize co-operation.
6) It's a way to detect and cause the removal of anything that gets in the way of developing and delivering products.
7) It's a way to maximize productivity.
8) It's scalable from single projects to entire organizations. Scrum has controlled and organized development and implementation for multiple interrelated products and projects with over a thousand developers and implementers.
9) It's a way for everyone to feel good about their job, their contributions, and that they have done the very best they possibly could.

Naturally the scrum focuses on an entire organization on building successful products. Without major changes, often within thirty days, teams are building useful, demonstrable product functionality [3,10]. Scrum can be implemented at the beginning of a project or in the middle of a project or product development effort that is in trouble [3, 1].

The scrum is the set of different interrelated practices and rules. And that optimizes the development environment, reduce organizational overhead, and closely synchronize market requirements with iterative prototypes. Based in modern process control theory, scrum causes the best possible software to be constructed given the available resources, acceptable quality, and required release dates. Useful product functionality is delivered every thirty days as requirements, architecture, and design emerge, even when using unstable technologies [1, 12].

The Sprint Backlog is the list of tasks that the Scrum Team is committing that they will complete in the current Sprint. Items on the Sprint Backlog are drawn from the Product Backlog by the team based on the priorities set by the Product Owner and the team's perception of the time it will take to complete the various features. It is critical that the team selects the items and size of the Sprint Backlog. Because they are the ones committing to completing the tasks they must be the ones to choose what they are committing to. The Sprint Backlog is very commonly maintained as an Excel spreadsheet which is shown in below figure 1.

Scrum teams do not include any of the traditional software engineering roles such as Programmer, Designer, Tester, or Architect. Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint. Scrum teams develop a deep form of camaraderie and a feeling that "we're all in this together." A typical Scrum team is 6-10 people but Jeff Sutherland has scaled Scrum up to over 800 people. The primary way of scaling Scrum to work with large teams is to coordinate a "Scrum of Scrums" or "Meta-Scrum" [10]. With this approach each Scrum team proceeds as normal but each team also contributes one person who attends Scrum of Scrum meetings to coordinate the work of multiple Scrum teams. These meetings are analogous to the Daily Scrum meeting but for do tend to happen weekly rather than daily.

Over fifty organizations have successfully used Scrum in thousands of projects to manage and control work, always with significant productivity improvements. Scrum wraps an organization's existing engineering practices; they are improved as necessary while product increments are delivered to the user or marketplace. As heard about Scrum, "oh, that's just my idea X by another name". Except Scrum is spelled out as values, practices, and rules in a development framework that can be quickly implemented and repeated. Scrum has been used to produce financial

products, Internet products, and medical products by ADM [1]. In every instance, the organization was log jammed, unable to produce shippable products for such a long period of time that engineers, management, and investors were deeply concerned. Scrum broke the logjam and began incremental product delivery, often with the first shippable product occurring with the same quarter [1].

In detail we already talk about the XP and Scrum in the above with the different market practices but about the implementation of XP, Scrum from the approaches of agile with the help of PSP/TSP with CMMI we can get the better way for finding the good results and make our new methodology.

So for getting the new ideology we should know about the PSP/TSP as compared to CMMI and also it's necessary to know about the traditional methodology which is also called as Water Fall model. But this Waterfall/Traditional approach was our old ideology. And most of the projects were failed in this methodology [5]. The undetected errors may propagate up the hierarchy, affecting an increasing amount of the system state, and error recovery routines may therefore become more complex than the application software [11]. So, resultantly the errors in software methods are due to misconceptions of the designers and these are rectified in due process. If these errors remain in the methods then they are also in the higher project perspective. This phenomenon can be dangerous for the software and its lifetime extension and success alike.

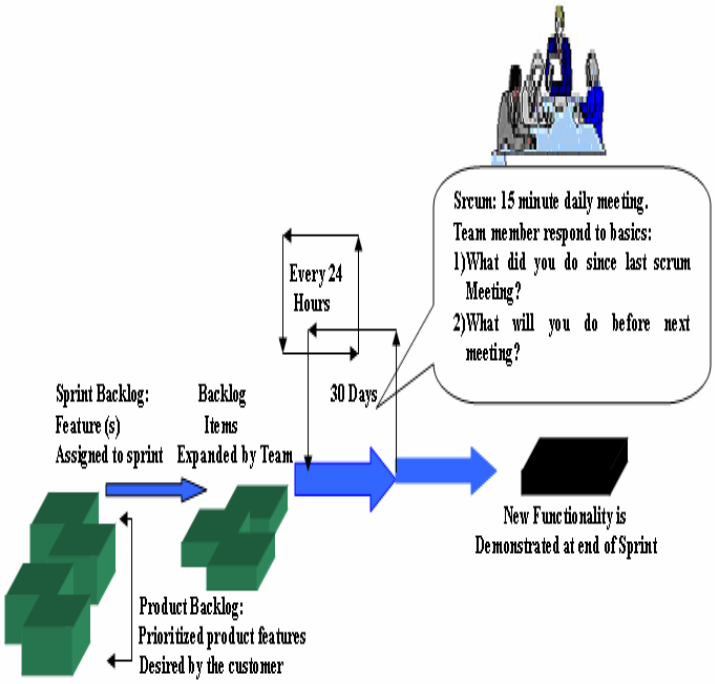

Figure 1, explaining about implementation of scrum [1].

2.2 Traditional (Waterfall) Approach

In History, many organizations used Traditional or Waterfall Approach for the software development. The waterfall model is a software development model [4, 13] (a process for the creation) in which development is seen as flowing steadily downwards just like a waterfall, through the different phases which are given above in the figure 2. This above waterfall model is also having a major benefit that we can fix our program design on the early stages because otherwise its

hard to change after few weeks. Time spent early on the software production can lead to greater economy later on in the software life cycle [8,14,15]. But there is a big problem with this traditional appraoch if we want to make some changes, its mostly very hard. For example if the bug found in the early stages of production we can change in our development. But if we are in verification phase and we got some bug then we will go back in the design phase where first we will change design and then again come in the implementation. So, that's why most of the projects were failed in the history because all the time/money means investment were wasted and those projects were overestimated.

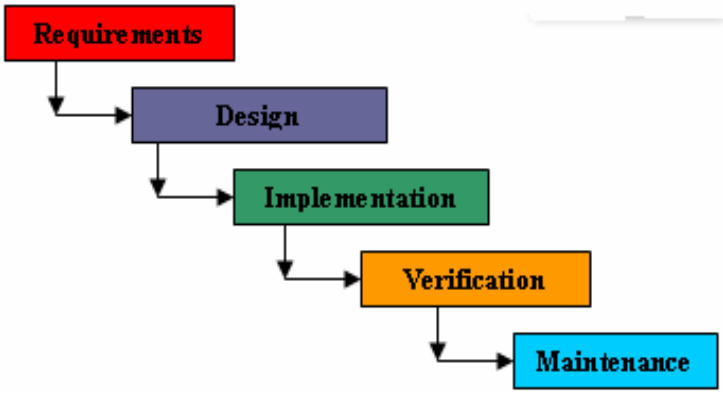

Figure 2, shows the flow of waterfall model on different phase levels [13]

2.3 Methods from US Software Engineering Institute

The Carnegie Mellon Software Engineering Institute (SEI) is saying that they help organizations and individuals to improve their software engineering management practices [7]. The Carnegie Mellon institute is offering different levels of training in which the personal software development is one of the most effective methods to develop.

2.3.1 TSP

Team Software Process (TSP) is proven way to establish quality strategy [16]. This tells us how to measure and establish criteria to manage the quality of products and processes at team level.

Creation of reliable and consistent software needs more than just bunch of developers equipped with PSP. It requires a team effort such that the whole team is more than incoherent individuals trying to work their way. TSP aims at transforming a group into a team, into a unit that believes in common goal, risks and has consensus on all the contingencies, which may subsequently follow [16].

The TSP approach builds teams that work effectively and efficiently together so that they may achieve greater heights by being together [16]. In TSP enabled environment, goals are set and embraced by all team members as their own without feeling oppression.

2.3.2 PSP and its relationship to TSP

PSP helps Software Engineers to reduce errors at their personal level. PSP focuses on polishing personal skills and makes realistic predictions possible [16]. TSP goes one step further based upon the conquests of PSP and binds the whole team together such that everybody knows his role and

goal. The whole team sets goals and then strives for it with utmost commitment and coordination [16].

2.3.3 TSP and Software – Capability Maturity Model (SW-CMM/CMMI)

CMM /CMMI do not provide detail implementation of each process area of a project. It just focuses on the requirements to achieve a specific level. It encompasses some key requirements in the organization and project. Where, TSP provides detailed guideline of the practices that should be followed to satisfy the Key Process Area (KPA) requirements of a CMM/CMMI level [7, 16].

So, the personal software process is helpful for developer in this sense that he can make his accuracy, realistic estimates and then routinely produce on schedule, with reduced development time and significantly reduced numbers of defects in delivered code [16]. Same is the case of Team level and they are saying that the team software process (TSP) methodology builds on the foundations of the CMM and PSP to guide organizations in forming and managing high-performance integrated product teams and also the TSP provides guidance on launching, planning, managing, and reporting a team's work [7, 16].

They are also providing some tools for different kind of purposes but the good thing in the institute is that they are willing to help those institutes who want to help in the field of software engineering.

These lifetimes of errors can be nullified by a thorough study and evaluation as proposed in this paper.

3 Methods / Materials

The paper is based on academic as well as well as industrial aspects in software development through agile methods. So the way which we choose for this topic is based on literature survey as well as on through web browsing/searching materials like research papers, tutorial, market news. And also the market analysis and requirements with the current position of agile methodologies are also important.

The literature study was done from the books, research paper of the people available in the university library and the material available on the websites etc to collect the subject related material. Also in the literature study for this topic, it was necessary to search the material from the conferences for agile methods which were held on different places. So, we followed the chain of references (found articles by following references found in other articles). As well as the discussion of different personnel's and companies which are implementing and criticizing on these issues that what's that best way for software development. The list of the references is given below in the references. The search engines utilized for this topic includes yahoo.com, google.com, answers.com, scirus.com etc.

We planned to take this as a literature study by using run time cases from the organizations. We sent so many emails to different software company's personnel are who are already working in the era or experienced. And we have not got good response and guidance to accomplish our task. So it's not necessary that we will get any good responses from different personnel's because most of the people don't want to response back in detail. Most of the results also came from the discussion of the conferences which were held in different timings in different places under some organizations. Most of my paper results are based on conferences and published papers.

4 Implementation

4.1 Current Status of Agile Development

Mostly, the problem with engineers is that change translates into disorder [17]. Especially when a single error can come with some possibility the entire system will be brought down. But, the change also translates into opportunity. It's as simple if there is time to put a certain amount of functionality into the product easily. Because after that there is time to put in more functionality at the price of a certain amount of disruption and risk. Thus does stupidity steal into our projects but we will have a tendency to take on as much risk as we possibly can [17].

James Bach in "American Programmer" effectively sums up the current Software Management method. There is a constant need for delivering "more" in a given amount of time [17]. The other attributes of the current development environment can be described by the attributes which are given below [17]:

1) The availability of skilled professionals - the newer the technology, tools, methods, and domain, the smaller the pool of skilled professionals [17].
2) The stability of implementation technology - the newer the technology, the lower the stability and the greater the need to balance the technology with other technologies and manual procedures [17].
3) The stability and power of tools - the newer and more powerful the development tool, the smaller the pool of skilled professionals and the more unstable the tool functionality [17].
4) The effectiveness of methods - what modeling, testing, version control, and design methods are going to be used, and how effective, efficient, and proven are they [17].
5) The domain expertise - are skilled professionals available in the various domains, including business and technology [17].

In the case of a normal and heavy weight software Management & development process, we should try to follow the different things in which meet with the customers, model the processes required for the custom coffee maker, get sign-off on the requirements to ensure that the customer does not change their minds, create a detailed project plan of the entire project, and assign resources to tasks, Project progresses like different individuals working on different pieces may contact customer with similar or different questions. So it's difficult to assess overall status as plan is challenged with normal disruptions and surprises. So that's why schedule the problems early are addressed by shortening future tasks, like testing, try to complete project like customer requests many modifications then project becomes a maintenance project rather than a development project [17].

4.2 The Source Code with Models, and Documents

In our literature survey we got some information which is relevant to the documentation within the limits of agility. So, let's start with understanding the relationships between models, documents, source code, and documentation, something depicted in figure 3 [18]. From the AM's point of view a document is any artifact external to source code whose purpose is to convey

information in a persistent manner [18]. This is different from the concept of a model, which is a generalization that describes one or more aspects of a problem or a potential solution addressing a problem. Some models will become documents or be included as a part of them, although many more will simply be discarded once they have fulfilled their purpose. Some models will be used to drive the development of source code, although some models may simply be used to drive the development of other models. Source code is a sequence of instructions, including the comments describing those instructions, for a computer system [18]. Although source code is clearly an abstraction, albeit a detailed one, within the scope of AM it will not be considered a model because I want to distinguish between the two concepts. Furthermore, for the sake of discussion the term documentation includes both documents and comments in source code [18].

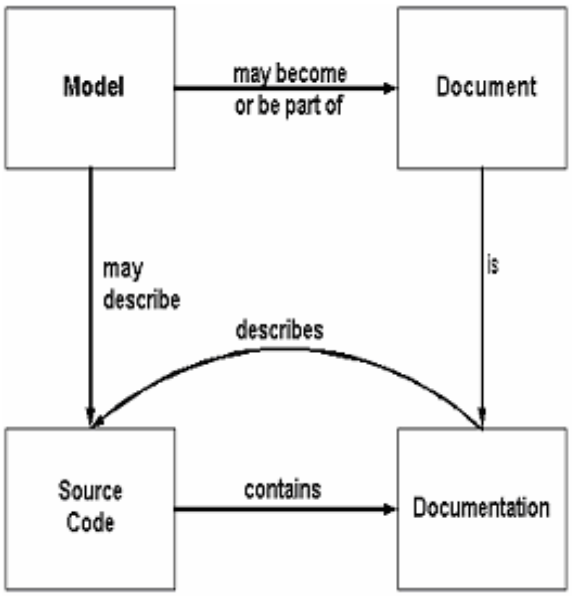

Figure 3, the relationship between models, documents, source code, and documentation [18].

4.3 The Effective Documentation Handoffs

A documentation handoff occurs when one group or person provides documentation to another group or person. Agile developers desperately try to avoid documentation handoffs because they are not a very good way for people to communicate [18]. Unfortunately documentation hand-offs are a reality in some situations – often your development team is so large it cannot be co-located, perhaps a subsystem is being created by another company (implying the need for a contract model), perhaps important project stakeholders are not readily available to your team, perhaps regulations within your industry or organization require the production of certain documents, or perhaps your team is made up of specialists instead of generalizing specialists. The following strategies can help to increase the effectiveness of documentation handoffs [18]:

1. Avoid documentation handoffs, As you migrate to an agile software development process you will constantly run into people who are not as agile, people who see nothing wrong with documentation handoffs. Point out that there are better ways to communicate – face-to-face conversations, video conferencing, telephone conferencing – that you should consider before writing documentation and whenever possible try to find a better way that fulfills your needs [18].

2. Support handoffs with other means of communication, if you can't avoid providing documentation to someone else you should at least strive to support the handoff with face-to-face communication or other approaches. This may enable you to write less documentation, therefore allowing you to focus on other activities, and will help you to avoid some of the common disadvantages of documentation such as misunderstanding the material [18].
3. Avoid documentation handoffs, Chances are good that the people you are interacting with don't like writing and receiving documentation either – at least it doesn't hurt to ask.
4. Write agile documentation, In which we have some comments which are given below [18]:

    1. Focus on the customer and keep it just simple enough, but not too simple.
    2. The customer determines sufficiency
    3. Document with a purpose.
    4. Prefer communication over documentation.
    5. Put the documentation in the most appropriate place.
    6. Wait for some results of the documentation.
    7. Display models publicly.
    8. Start with models you actually keep current.
    9. Require people to justify documentation requests.
    10. Write the fewest documents with least overlap, Get someone with writing experience.

4.4 The Agile Work Environment

Multiple layers of management that separate people from information, customers, and the ability to make knowledgeable decisions won't work in future agile. Neither will people who want to do one job, make limited decisions, take no risks, and pass each challenge to their supervisor. As a manager in the desired environment, every time you make a decision that could be made by the individual who has the knowledge, the proximity to the situation, and the need, you deprive that person of the opportunity to grow. Direction and focus, in this environment, is provided by leaders who drive and communicate the organization's strategic vision throughout the workplace, daily, incessantly, and consistently. People internalize this vision and perform their work to maximize its attainment. Furthermore, if you are still focused on meeting customer needs by providing a quality product, on time, that meets requirements, for a price your customer is willing to pay, you are lagging behind the learning curve. According to Daryl R. Conner, CEO of ODR, Inc., "the defining moment for customer service will be not when established needs are expressed, but will be when the unexpected requirement materializes over night" [1,17]. Conner cites three critical characteristics of the nimble organization. These organizations:

1) Hire only the agile. Conner believes that who is on your team is more important than how the team is structured or its assignment. "When staffing your organization for nimbleness," he says, "80 percent of your resources should be directed toward hiring people already prone towards the desired attributes, and then training and coaching them to expand their capabilities even more. No more than 20 percent of your resources should be allocated to assisting those who say they are willing to work against their own instincts and biases and try to develop completely new propensities" to become nimble and resilient.

2) Understand the interaction of control and resilience. When change is introduced, it is typically better handled by resilient people. It is better integrated by people who are used to constant change and who are not taken by surprise by the announcement or request.

3) Build a core competency around handling ambiguity. People who handle change most effectively recognize that change can be scary, perhaps unpleasant, and that it always requires

something different from them. Despite this, they continue to rise to the occasion and effectively perform their job responsibilities.

Generally, in the practice the ideas are coming from the real time projects and the market situation also. I immensely like the short development cycles prescribed in all of the agile development methods. This is very practical and is true in many organizations. I have also observed in my professional experience in the software industry that having short development cycles of 15 days to one month, and going for a review of the entire project after that really is very pragmatic.

Now for the darker side of agile development methods, Agile development methods place too much premium on people. It expects highly motivated people. It also expects colleagues to interact in a very close manner, everyday, without disagreement, emotions or bias. This will just not happen. It is very tough to work very closely with a group or people and get along with them. People will have disagreements everyday, which will affect the way in which future interactions with clients and fellow developers will be affected. If the heavy emphasis on people is not sorted out, Agile may end up being a very bold experiment.

Agile software development is a conceptual framework for undertaking software engineering projects. There are a number of agile software development methods, such as those espoused by The Agile Alliance, a non-profit organization [19]. This agile development is based upon agile methods which are implemented on process and project level. There are many contradictions to the agile methods in the approach and in execution. Agile methods are having some reputation if error prone due to their ad hoc nature and the approach of the agile methods. The main of agile methods is their potential scope of alleviating component-project instabilities and the ease of higher order transformation. If agile processes fail to meet the raison put off, then there is very little support for them to find and lend in the industry.

This requires a comprehensive approach to the management of variability that can be applied throughout the various lifecycle stages, their artefacts, and their accompanying notations in an universal manner. Moreover, in order to enable a smooth transition to product line development for an organization that so far only performed single system development, it is necessary to keep as many of the existing notations and approaches in place as possible [20].

4.4.1 Reusability in Agile Development

Component Based Development aims at constructing software through the integration of components by using interfaces and contracts among them. However, these components should be bound to a specific application domain in order to be effectively reused [21]

The needs of customers are now highly specific and rapidly changing, although they still want to have high-quality and low-cost products/services. The companies should respond to these new, rapid, continuous and predictable changes in environment and should provide a suitable product/service variety in order to survive and be competitive in the market [22].

Systematic Software Reuse (SSR) [23] is a software development methodology through which organizations identify common functionality among applications within a domain and build prefabricated software assets to plug and play in various custom solutions. Despite the hype and various prescriptive methods that detail the process of implementing SSR, very few organizations are able to materialize the promises set forth by the technology [24].

Agile manufacture based on dynamic alliance is coming into being so that enterprises can remain competitive in a constantly changing business environment and is becoming a main competitive paradigm in the international market. Agility, which has basically two meanings:

flexibility and reconfigurability, as become a very important characteristic of a modern manufacturing enterprise [25].

The main task of agile match design is to select the match elements and determine the position of these elements. The constraints in the agile fixture designing process include the constraints of the function and the constraints of the position of the elements [26]

There are many other ways of developing software. But the agile way is a more reactive than others. It responds to the changes. These are implemented in the method level and the incarnation takes it to another level. The whole project is reflecting the idea and gets disturbed with the developments at micro level and all the project benefits. Such laissez-faire approach helps the project at big level.

The agile software development always focuses on the client. It moves forward based on the client recommendations. All these areas of methods coincide with the culmination of these processes withstanding all the relevant issues of pertinent nature. The emerging of numerous different agile methods has exploded during the last years and is not showing any signs of ceasing. This has resulted in a situation where researchers and practitioners are not aware of the existing approaches or their suitability for varying real-life software development situations. As for researchers and method developers, the lack of unifying research hinders the ability to establish a reliable and cumulative research tradition [27].

Processes employed are in general ad hoc although some organisations are starting to look into the use of agile methods [28].

4.5 Strength of Agility

Agile methods are not very useful if used alone but these can be good if used together with other methods and approaches. It is meant to be used in conjunction with other agile software processes such as Extreme Programming and Scrum Development Method, as well as "near-agile" instantiations of the Unified Process [29].

The goal of agile software development is to increase the ability to react and respond to changing business, customer and technological needs at all organizational levels. Agile software development methods are used in a hope of nearing toward this goal. While agility refers to nimbleness, a method aiding this process needs to be nimble as well. In other words, in order to increase the level of agility, the tools (i.e., agile software development methods) used in the development process need to be agile also, i.e. situation appropriate [30].

The development of an enhancement must be completed within a single release. Thus, enhancement allocation, which assigns enhancement work to a particular team, is a critical task whose success depends on the development teams' expertise and capabilities [31] when the demand shifts, then all the elements of supply change too. This correlation of demand and supply are market forces of economy. The entire economic circle revolves around these things.

To cope with unpredictable demand and a wide variety of products, future production systems require agility. To realize manufacturing agility, the control system has to respond and adapt to variations in real-world, dynamic production environments [32, 33].

4.6 Other Methodologies of Software Development

Evolutionary or iterative development is also an aspect of agile approach. [Agile evolution, Shaw, Brock Source: Computer Bulletin, 2004]. This development comes very handy in numerous software projects of large scale. Another point of commonality for all agile methods is the recognition of software development as an empirical (or nonlinear) process. In engineering, processes are classified as defined or empirical.

A defined process is one that can be started and allowed to run to completion, producing the same results every time [34].

All this pressure involves the agile methods to cover hole. This makes these processes very stable and balanced. Today, competitive pressures require companies to be very fast in introducing new products, to have short production lead times to manufacture and deliver products to customers and to be permanently aligned with the market. Product life-cycles tend to shorten [35].

This situation can only be controlled with agile processes. A surprising number of developers view using agile processes as an attempt to micromanage [36]. Because approaches like Scrum and XP accelerate project cycles, developers typically interact with their managers more frequently but for shorter periods. In a plan-driven process, a manager might go a full week without talking with a particular developer, but daily contact is the norm in most agile processes [36]. All the things that humans make can have errors and software processes are just like that at all. Some of these processes are agile and some of the errors that are there are in the agile processes and methods and these can be corrected. These corrections are in many steps and these steps are intermediate stages of agile software processes. The amount and character of the risk-creating errors point to the importance of interventions that promote learning at the levels of the work team and the organization [37].

On the basis of Journals, discussions in seminars, research papers, market analysis of good companies and literature study we deduced some results which are the following but first we should know about the characteristics of agile development before going in the detail.

4.7 Our Derived Knowledge by XP, Scrum, Traditional Approach with PSP/TSP/CMMI and Agile Development Characteristics

Today's time-sensitive business climate requires that we quickly accommodate requirements changes during development and, after development, be equally adept a delivering the upgrades caused by software's rapid software evolution and the customer's ever-increasing requirements. A dominant idea in agile development is that the team can be more effective in responding to change if it can reduce the cost of moving information between people, and reduce the elapsed time between making a decision to seeing the consequence of that decision, place people physically closer, replace documents with talking in person and at whiteboards, and Improver the team's amicability-its sense of community and morale- so that people are more inclined to relay valuable information quickly, make user experts available to the team or, even better, part of the team and work incrementally.

This all above can be possible if we will use the Personal software process by giving some training in our software houses, for the employees. Because by these kind of training we can get the maximum output from our personnel's. Due to these training, every personnel will be self driven and will be able to work in the Team level. As well as we will have their previous records regarding the progress that what kind of tasks they already done and how much time they consumed already in their previous tasks. So, we can easily estimate them that how much they will take time for next tasks and also PSP type of skills develop the reusability of components for their next task. By using the help of documentation from the traditional approach we can do our work very speedily and can get minimum errors with maximum surety. About the documentation within the limits of agility, we already described in the above sections **4.2** and **4.3.** Where we learnt about that how we can adopt the agility in documentation.

5 Results

The agile methodologies offer down-to-earth approaches to software development that focus on simplifying and improving the software development process, making the customers, the developers and the final product-the most important [38].

While creating new software, engineers require source code of previous projects. In keeping with agile principles, we are only concerned with source code reuse in our present work [39]. Therefore code reuse focus helps our software reuse aim. Aforementioned PSP/TSP technique is a software way of continuous improvement. The best architectures, requirements, and designs emerge from self-organizing teams [40].

Agile Software developers are only human; humans make mistakes that result in defects that must be corrected [41]. As these error base on individuals so correction should be initiated by individuals. Agile methods have evolved as a bottom-up approach to software development [42]. So they prioritize putting the constituent parts of software right. Clearly, agile concepts will continue to migrate into traditional organizations (and vice versa) through planned [43]. Agile software methods should thus be adjusted according to the current status and adaptability strength of the organization. To fully benefit from agile practices, organizations must better define the interfaces between the agile team and its environment, thus avoiding the double work caused by the conflict between agile practices and traditional ones. [44]

6 Conclusion

Our original goal in this work has therefore been achieved. So, in conclusion the agile methods are light weight software methods. Agile development methods are very pragmatic in understanding the fact that requirement in a business environment changes constantly. Highly creative people who have understood the shortcomings of normal software management processes are using agile development methods in organizations. Many organizations all around the world are trying out the various available agile development methods. The advantages from the agile development we can get like shortened development cycle-time of 75%, higher stability of work-loads, higher utilization of work-load i.e. developing large-scale, software systems with a fixed number of developers, higher flexibility to change of Management & development plans, higher quality by earlier feedback from the customers. All the software methods using agile techniques have been improved and solution their problems diagnosed for improvement. My take on the tasks on hand were to devise these recommendations and solutions after pointing the potential pitfalls of this area. PSP and TSP if used with the traditional agile technology are the best combination in the software engineering. Scrum is also a very viable contender to be involved. We worked on this paper in deep but for future work if we will use the research papers which is available on the different site and especially given on IEEE site, then it can be more helpful for further situations in the software development as well as organizations much material regarding to current. For the better understanding that how it will work, see the figure 4 which is given below:

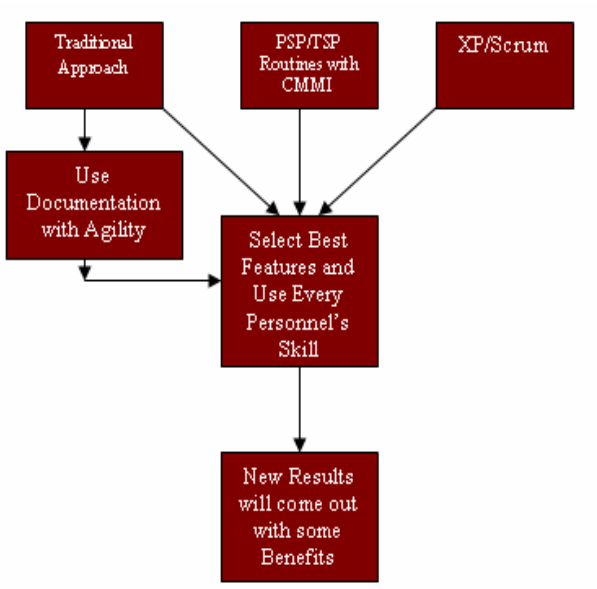

Figure 4, Model for new ideology by using some existing concepts.

Normally we have a lot of ways for software development but in each way we have some benefits as well as some drawbacks. Some times these drawbacks are making the hurdles for achieving our goals. So that's why we researched on the different of software development. Where we studied that if we will combine them and first we will choose their benefits and then combine them. It's the combination of different above said ways but mostly it can help for the best software development through agility because it's not necessary that it will implement on every kind of project. By the combination we can say that we achieved our goal because we can do our any kind of software development through these agile methods which are trying to show in the above said figure 4. But for getting our goals successfully, if we will get the help of PSP/ TSP for training our staff, we can do software development through this method.

We already discussed in our above sections like background, implementation, results and the conclusion where we gave a lot of ideas and working scenarios in software development through agility. But in our proposed recommendations if we will consider large organization with the big teams then the scrum is good with the help of documentation and PSP/TSP trained personnel's otherwise if we want to focus on only code of the software development then we will consider the XP with the help of traditional methods and PSP/TSP trained personals. So, we proposed the hybrid solution where we discussed about PSP/TSP for helping to XP & Scrum. But if we want to do development and major focusing on the coding then we will use XP and we can write the documentation through scrum. But for both we will use the well trained people with the help of PSP/TSP & for organization level software CMMI.

7 Future Work

Agile methods will be better as this work helps new agile methods. All the relevant fields of computers will also find information for the development and research purposes. Errors in the agile methods have been identified and the solutions been proposed. The laid down foundations in the agile approach create new basis of agile methods and diminishes chances of error prone aftermaths.
This paper is focusing on the new plan for agile methodologies to any organization, who want to play a vital role in the field of software development. And better security for the software

engineering field will be implemented and new rules will be introduced and implemented with the help of different software development methods with the help of agile methodology as well as any other organization who wants to make their own standards.

8 References

[1] Ken Schwaber and Mike Beedle, "Agile Software Development with Scrum" Prentice Hall, 2001.

[2] Addicam.V.Sanjay, "Overview of Agile Management & Development Methods", www.projectperfect.com.au, Visited on 09th July, 2006.

[3] Agile Alliance. (2001, February). History: The Agile Manifesto, Visited on Sept 27th June 2006, http://agilemanifesto.org/history.html

[4] Scott W. Ambler, "Agile Modeling - Effective Practices for Extreme Programming and the Unified Process", John Wiley & Sons 2005.

[5] Maurer, Frank. (2002). XP in detail: University of Calgary, Visited on May 22nd 2006, http://sern.ucalgary.ca/courses/SENG/609.24/W2002/slides/XP2.ppt

[6] Nelson, Elden. (2002, January). Extreme Programming vs. Interaction Design, Visited on May 25th 2006, http://www.fawcette.com/interviews/beck-cooper

[7] Dave Scherb Dscherb, SEI Areas of Work: Management, 2nd July 2006, http://www.sei.cmu.edu/managing/managing.html

[8] W.S Humpyrey, "A Displine for Software Engineering First edition, Addison- Wesley company, Reading, 1995.

[9] Agile model driven development is good enough, Ambler, Scott W. IEEE Software, 2003

[10] Control Chaos. (2001). SCRUM software Development process. Visited on 22 April 2006, http://www.controlchaos.com/scrumwp.htm

[11] Error injection aimed at fault removal in fault tolerance mechanisms-criteria for error selection using field data on software faults
Christmansson,J.;Santhanam,P.; Software Reliability Engineering, 1996.

[12] Control Chaos. (2002). What is Scrum? http://www.controlchaos.com/scrumo.htm, Visited on 23rd April 2006,

[13] Capers Jones, "Software Project Management Practices: Failure versus Success", CROSSTALK The Journal of Defense Software Engineering Oct 2004 Issue, http://www.stsc.hill.af.mil/crossTalk/2004/10/0410Jones.html

[14] Sommerville, Ian, Software Engineering, 7th edition, Addison-Wesley, Boston 2004.

[15] S.L Pfleeger, "Software Engineering Theory and Practice, (2ed)", Prentice Hall, Upper saddle River, 2001.

[16] The Software Engineering Institute (SEI) by Carnegie Mellon University., The Team Software Process (TSP) and the Personal Software Process (PSP), http://www.sei.cmu.edu/tsp/, Visited on 30th June, 2006.

[17] Addicam.V.Sanja, info_agile_programming.pdf, "Overview of Agile Management & Development", Visited on 15th June, 2006, www.projectperfect.com.au

[18] John Wiley & Sons , Agile Modeling, "Effective Practices for Extreme Programming and the Unified Process", Scott W. Ambler, http://www.agilemodeling.com/essays/agileDocumentation.htm

[19] http://en.wikipedia.org/wiki/Agile_software _development, Visited on 27th March, 2006.

[20] A customizable approach to full lifecycle variability management, Klaus Schmid, and Isabel John, Science of Computer Programming , 2004.

[21]Odyssey-Search: A multi-agent system for component information search and retrieval , Regina M.M. Bragaa, b, Cláudia M.L. Wernera, Marta Mattoso, Journal of Systems and Software , 2006.

[22] A survey on the methods and tools of concurrent new product development and agile manufacturing, Buyukozkan, Gulcin; Dereli, Turkay; Baykasoglu, Adil, Journal of Intelligent Manufacturing, 2004

[23] M Smolarova, P Navrat - Journal of Computing and Information Technology, 1997 – www.dcs.elf.stuba, visited on 1st July, 2006.

[24] Resources and incentives for the adoption of systematic software reuse, Karma Sherif, Radha Appan and Zhangxi Lin, International Journal of Information Management , 2006

[25] Study on multi-agent-based agile supply chain management, Lou, Ping; Zhou, Zu-De; Chen, You-Ping; Ai, Wu, International Journal of Advanced Manufacturing Technology, 2004

[26] Case-based agile fixture design, Li, Peigen Li, W.; Rong, Y. , Journal of Materials Processing Technology, 2002

[27] New directions on agile methods: a comparative analysis, Abrahamsson, P.; Warsta,J.;Siponen,;Ronkainen,J.; International Conference on Software Engineering, 2003

[28] Investigating Web size metrics for early Web cost estimation, Emilia Mendes, Nile Mosley and Steve Counsell, Journal of Systems and Software , August 2005

[29] Lessons in agility from Internet-based development, Ambler, IEEE Software 2002

[30] New directions on agile methods: a comparative analysis, Abrahamsson, P.; Warsta, J.; Siponen,; Ronkainen, J.;
International Conference on Software Engineering, 2003.

[31] Web-based Agile Software development Aoyama, Mikio, IEEE Software, 1998

[32] Agile control systems, Young, K.W. Muehlhaeusser, R.; Piggin, R.S.H.; Rachitrangsan, .Journal of Automobile Engineering, 2001

[33] Core-bored search-and-rescue applications for an agile limbed robot, Larson, Amy C.; Bae, Jaewook; Lapoint, Monica International Conference on Intelligent Robots and Systems, 2004

[34] Agile software development: It's about feedback and change, Williams, Laurie, Alistair Source: IEEE Computer Society, 2003

[35] On the dynamics of Agile/Virtual Enterprise reconfiguration, Putnik, Goran D., International Journal of Networking and Virtual Organisations, 2005

[36] Introducing an agile process to an organization, Cohn, Mike; Ford, Doris, IEEE Computer Society 2003

[37] Human actions and errors in risk handling—an empirically grounded discussion of cognitive action-regulation levels, Marianne Döös, Tomas Backström, Carin Sundström-Frisk , Safety Science, 2004

[38] Integrating agile software development into stage-gate managed product development, Daniel Karlstrom, Per Runeson, Empirical Software Engineering, 2006, DOI10.1007/s10664-006-6402-8

[39] Frank McCarey, Mel Ó Cinnéide, Nicholas Kushmerick, and Rascal: A Recommender Agent for Agile Reuse, Springer Netherlands, DOI: 10.1007/s10462-005-9012-8

[40] Online document, Available HTTP:http://agilemanifesto.org/principles.html

[41] Richard E. Fairley, Mary Jane Willshire, Iterative Rework: The Good, the Bad, and the Ugly, IEEE Computer Society, 2005

[42] Daniel Karlstrom, Per Runeson, Integrating agile software development into stage-gate managed product development, Empirical Software Engineering, 2006, DOI: 10.1007/s10664-006-6402-8

[43] Barry Boehm, Richard Turner, Management Challenges to Implementing Agile Processes in Traditional Development Organizations, IEEE Computer Society, 2005

[44] Agile Software Development in Large Organizations, Mikael Lindvall, Dirk Muthig, Michael Stupperich, David Kiefer, John May, Tuomo Kähkönen, 2004, IEEE Computer Society, 2004 IEEE