Loyola University of Chicago

Research Project

Agile Software Development

Summer Session A

Student Name	Student ID
Syed Saad Ahmed	Sahmed10@luc.edu

Submitted To: Prof Robert Yacobellis

Submission Date: July 2nd, 2010

Topic Selected: Review and investigate the use and impact of Agile methods in industry

TOTAL PAGES: 48

TOTAL WORDS: 14,829

Acknowledgments

I would like to thank Professor Robert Yacobellis for sponsoring this research idea and introducing me to the wonderful world of Agile Development.

Abstract

This is a research report for the Summer 10 session course at loyola University Chicago and deals with the research of software development methodology of Agile Development. It's basic purpose is to investigate the use and impact of Agile methods in industry and tries to answer questions such as Do Agile projects have to stand alone, or can they be combined with more "traditional" software life cycles and methodologies like waterfall, incremental, Spiral, or JAD and RAD? What are the weaknesses of the previous methodologies mentioned above, and how did Agile evolve from them? Also a general research overview on mainstream Agile models such as XP and its practices SCRUM, Crystal family, DSDM, and lean development methodology.

The references section provides a list of references, material, journals, interviews conducted, and articles reviewed for the completion of this report.

Table of Contents

USE AND IMPACT OF AGILE METHODS IN INDUSTRY

Introduction to Software Development Life Cycle

Software life cycle or software process usually describes a set of activities and tasks that take place in some particular order during the process. The objective of the execution of a life cycle is creating a software product. The primary activities involved in a life cycle are as follows

- Requirements: Responsible for gathering requirements, needed information, function, behavior and performance
- Design: Designs the overall blue print of the system
- Implementation: Generate actual source code, database code, documentation etc
- Verification: Verify if the implemented product solves the actual problem
- Maintenance: Maintenance of the created software.

Each of this activity is a phase of the life cycle and is responsible for the progress of the project to its completion. The above phases are streamlined in different models also called methodologies of software development. Each model has its own approach toward following the phases and thus has its own merit and demerits. The international standard for describing the method of selecting, implementing and monitoring the life cycle for software is ISO 12207. *(Wikipedia, 2010).*

Traditional Software Development Approaches

There exists multiple models and methods to assist and carry out the software development process. The most traditional software development approach could be said to be flow charting

which although not a whole methodology but gave rise to the notion of methodologies. The most historic framework which still has its significance today could be described as the linear framework approach. *(Agile Software Development, Cockburn, Highsmith)*

a) Linear Waterfall Model

The linear waterfall model or commonly known just as the waterfall model is a SDLC process where the developers are supposed to follow the above mentioned phases in a particular order, which is that one phase is dependent on the previous phase and so on. Also each phase generates some output which becomes the input of the next phase in life cycle. Each phase is finished completely before proceeding to the next phase. Also each phase has a set of deliverables which is a result of the successful completion of that particular phase.
The benefits of this model is that its easy to understand and easy to use. Each phase has a set of mile stones. The water fall life cycle, emphasizes big up front specifications, estimates, and speculative plans and instill on that for the remainder of the life cycle. *(Agile Software Development: Best Practices for Large Software Development Projects)*

b) Flaw in the plan
However as mentioned above the waterfall considers software production as a predictable process and assumes requirements would all be well known upfront and considers the outcome of each phases to be absolute and frozen, the result of which is inflexibility. The waterfall model is more suitable for predictable manufacturing where change is not the norm and predictability rates are extremely high. Thus waterfall life cycle is the wrong paradigm for software which is extremely unpredictable in nature during the development process, and practices and values rooted in it are not helpful. *(Agile Software Development: Best Practices for Large Software Development Projects)*

c) Iterative and Evolutionary Model
Iterative and evolutionary development model has been around since 1960s but only in past few decades has been understood to be a better response for the weakness of waterfall model.
It should be mentioned that Agile models is a subset of the iterative and evolutionary model and there are also some other iterative and evolutionary models which are not agile i.e doesn't put more emphasizes on the hallmark practices of Agile or are too heavy weight. In the layman terms iterative model. So in layman terms iterative development involves several iterations in sequence over the lifecycle and each iteration is responsible for creating a portion of the system or a refinement over the existing system. Each iteration outcome acts as a feedback or adaption mechanism for the next iteration cycle and the advantage of what was being learned during the development of earlier, incremental, deliverable versions of the system. By Evolutionary ID we mean that the software is created evolutionary by refinement over the course of iterations rather than frozen in a major up front specification. Iterative and Evolutionary development may also be called iterative and incremental development where the partial system grows incrementally with new features, iteration by iteration. The results of each iteration is integrated into a final public release which is released to the customer, however the result of each iteration is also delivered and showed to the customer as an internal release for feedback for next iterations.

This is the primary difference between waterfall and IID models, that in waterfall business values is released all at once and only at the end of the project. *(Craig Larman, 2003)*

d) Modified Waterfall

Modified waterfall is a slightly different version of waterfall where the phases are still followed in order from one to one and process flows from top to bottom however few more iterations are carried out after the final phase of waterfall life cycle to compensate for the downfalls of a single iteration. Also to mention that the initial waterfall model was designed by the creator to carry out another iteration but the idea didn't get registered to people and was misinterpreted. The author actually endorsed iterative development on the phases of waterfall. *(Wikipedia Waterfall, 2010)*
The modified waterfall allows backtracking to the previous phases if an incompetency arises, For e.g instead of just freezing the output of one phase MWF would allow the user to switch back to the previous phase to take care of the problem and then moving back to forward phase. *(Art of Agile Development, James Shore)*

e) Spiral Model

The spiral model is a combination of waterfall mode, iterative development and prototyping. The spiral model is especially suitable in large projects which has constantly shifting goals. The spiral model place emphasizes on iteratively following the phases of waterfall. This generates a prototype for the system from the premilinary design and then the second prototype is evolved by following the phases of Plan, Risk Analysis, Engineering, and Customer Evaluation again. The user sees the system early because of rapid prototyping and iterations. The Spiral Model is a risk-driven, as opposed to code-driven, approach that uses process modeling rather than methodology phases. *(Wikipedia, 2010)*

f) RAD(Rapid Application Development)

This is another methodology based on iterative incremental approach and prototyping. The main definition of RAD could be a system could be built in rapid time. The whole idea of RAD is based on the new CASE tools which has arrived to facilitate and quicken the time for software development. The RAD encapsulates all the phases of waterfall and keep special interest on requirement analysis and business requirements however the subsequent phases of implementing and programming are done in teams of who use iterative and evolutionary methods to create prototype as quickly as possible. The idea behind rad is to develop a sytem that passes the functionality list of atleast the minimum set of requirements rather than all the requirements. *(Wikipedia, 2010)*

g) Formal Methods

Format methods is another approach towards SDLC by designing, specifying, verifying, and analyzing the software in terms of mathematical notions and formula. The set of mathematical axioms and inference rules can be fed to the computer which describe all or part of a system. Thus we can derive a theorem prover to which we can input the specifications or requirements and the output would be a report where the inconsistencies are. *(Wikipedia, 2010)*

Flaws and failures of Traditional Approaches

In the previous section we outlined some of the traditional approaches to software development methodologies. Now we will see the problem with some of those approaches. We must point out that the target of our flaws is mostly the traditional water fall model. Mainly because waterfall is the most inconsistent one with the idea of Software Development paradigm, and some of the other models mentioned are logical evolution of waterfall model and tries to compensate the drawback and weakness of waterfall. However all the other models also suffer from some other problems and are heavy weight in nature, as compared to the new paradigm of Agile Development. *(Agile Software Development: Best Practices for Large Software Development Projects)*

a) Software as a new product development

The problem with the traditional waterfall approach is that it assumes software to be similar to a predictable manufacturing process where near the start one can reliably estimate effort , cost and requirements. This is because manufacturing process has a high rate of repeated identical or near identical creation. Thus where the manufacturing precitable process can afford to first complete the entire specifications and business requirements and then build without regard to change. Also it is possible to identify, define and schedule all the other activities from the beginning of the process. This is in contrast with software development where there is high degree of change, creativity and no previous identical cases to derive estimates or schedules from. Software development is not a defined process which can be designed same way and run repeatedly with predictive results. The main reason is that the input of the software process development is people who devised the flow of software development. The waterfall model was appropriate for a defined industrial process and was based on physical engineering process which is predictive. (*Craig Larman, 2003)*

Some of the factors which prevent reliable up- front specifications include

- The clients or users are not sure what they want
- They have difficulty stating all the want and know
- Many details of what they want will only be revealed during development
- The details are overwhelmingly complex for people
- As they see the product develop, they change their minds,
- External forces lead to changes or enhancements in requests.

So we can conclude that

> *Most software is not a predictable or mass manufacturing problem Software development is a new product development.*

b) Requirements, resources and deadlines

The requirements phase in the traditional waterfall is frozen and demands the customer to bring all the requirements upfront before the actual implementation of the code can began. The

drawback of it was that most of the time user because of burden was made to come up with specifications in a hurry and then sign off which eventually wasn't required in the end and never used. The deadlines in waterfall are not rigid as well, the main deadline of the last phase was usually define which happened to be after 6 months. This might give the developers a false sense of security initially that everything is on time and eventually in the end find out that they were on the wrong direction altogether. The psychological effect of having a deadline far ahead in time is laziness and unproductivity until the deadline approaches near. The assumption that the project can be managed with predictability leads to treating individuals as resources. *(Agile Software Development, Cockburn, Highsmith)*

c) Uncertainty and Change

Uncertainty and change are inherent characteristics of Software Development. The bigger the size of the project and the more complicated it is the higher the chances of change rate. The main factor of change in software development is that the requirements of a user are uncertain and immune to change over time. Changing requirements of the user aren't welcome in the traditional waterfall method, which freezes the upfront defined specifications and forgets about it, thus in the end many features of the software which were specified early wouldn't be used. People sometimes aren't sure what they want. They think they know, but when they actually see it running, they realize that they wanted something else. Thus the business requirements aren't stable so its better to meet them in an iterative fashion. Its desirable that the developers embrace uncertainty as a part of process, the result of this would be a freedom to be innovative and creative instead of falling a defined set of rigor practices and inflexibility. *(Agile Software Development: Best Practices for Large Software Development Projects)*

d) Collaboration with customers

The input of a software development process is the client, who drives the development process. But in the waterfall method the only interaction with client happens in the beginning when he brings the business requirements to the development team. The next interaction is probably at the last phase of life cycle where the system is integrated and completely developed and is presented to the client for acceptance testing. This is contradictory to the idea of having a client driven development where feedback generated from the user steers the project further. *(Agile Software Development, Cockburn, Highsmith)*

Going Agile

In this section we will see how agile evolved out of previous methodologies of IID. We must keep in mind that Agile methodologies are a subset of Iterative and Incremental Development Although iterative and evolutionary method have been around since 1960s, Agile is a new hype which offers a fresh new insights to IID and recycles old ideas to a new flavor.

a) Evolution of Agile Development

As mentioned Agile evolved out of IID methods and after the relative failure rate of waterfall in large projects. In 1970s Waterfall was taught as the ideal approach towards Software Development due to the less experience of new programmers and also because Software development was a relatively new field and was considered equivalent to predictive manufacturing. The criteria for success of a software is on time completion, within budget, feature complete and also in complete working state. By utilizing the waterfall model these requirements weren't met. Infact only 10% of the software succeeded. *("Software Project Management Practices: Failure Versus Success," Capers Jones).*

Thus with time and failure of projects the main key points for the failure were realized as follows *("Agile Software Development," Alister Cockburn)*_

1) Customers mind changes with time
2) Lack of Communication and Feedback
3) No working version of the software until the last phase.
4) Inflexibility to change
5) Large duration of projects lifecycles.

Thus considering the above points a light centric approach which helps us adapt and change was evolved from insights and IID.

b) Agile as a lightweight framework

Although IID models have been influential in solving the problems mentioned above, they still relied on heavy weight processes and complex practices. Agile was a stepping stone in refining the framework of IID in favor of light weight process and sufficient practices, which means only to do what is needed and in a simple manner. Agile has the same iterative and incremental idea but encourages simple rules and a light touch and introduces some new practices which specifically emphasizes reducing the above mentioned problems.*(Agile. Methodologies, Venkat Subramaniam)*

c) Agile Manifesto and Agile Alliance

In 2001 a group of experienced software developers met to discuss the new successful strategies from lots of several methodologies and to find common groung among therm. They

each had a different combination of old and new ideas which they thought were successful for a project, but eventually realized that all those ideas emphasizes some common principles. Out of which came the agile manifesto. *(Agile Alliance , 2010)*

The agile manifesto had the following principles quoted from *(Agile Manifesto, 2010)*

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

d) Agile principles

Thus to practice the above mentioned principles agile development applies time boxed iterative and evolutionary development, adaptive planning, incremental delivery. All Agile models basically involve iteration and continuous feed back which lies at the heart of agile methods to successfully develop and deliver software. *(Craig Larman, 2010)*

All agile models also promote teamwork, collaboration, minimal upfront planning and value working software over heavy documentation. The main project success indicator is working software which comes in small increments which meets the need of stake holders in an efficient manner. Thus a project focuses on product features as the main unit of work planning, working and delivery. Any activity which does not directly contribute to this goal should be disregarded. Usually a project is divided into several releases. A release will have a list of features, defects, enhancement request and other work items organized. Each release is then divided into several iterations where each iteration would be assigned the items from the release list of requirements in order of priority. *(Agile 101, 2010)*

Thus we can conclude that Agile framework is built on the idea of continuous evolution i.e continuous planning, continuous testing, continuous integration, and other forms of continuous evolution of both the project and the software.

Some other key points in Agile are as follows

- **Iterative and Incremental development**: All agile models break tasks into small increments and employs short time boxed iterative development which requires a project to be built in several iterations where each iteration is a stable tested partially complete system. Agile emphasizes the iteration date to be short and fixed, and not to be changed even if the scope for the iteration is not complete at the end. Meeting the iteration deadline by reducing scope is valued more rather than extending the deadline to complete the scope. *(Craig Larman , 2010)*
- **Client Driven Planning:** Unlike waterfall which had minimal client involvement, Agile basically takes client as the inputs in order to plan and proceed the project further. Which means that after each iteration from the feeback of the user the next set of features would be prioritized and assigned to the iteration. Thus the user is continuously involved throughout the life cycle of the project and to guide the direction of the project.*(Craig Larman, 2010)*
- **Evolutionary Requirements and planning:** Instead of defining all the requirement upfront before starting development, agile only describes feature which are necessary for a particular iteration. Thus the requirements and additional features are evolved and refine when necessary with time. Emphasizes on just enough planning needed for the particular release is made instead of detailing everything at once.*(Agile 101, 2010)*
- **Embrace Communication and collaboration:** Agile has a main idea of increasing communication among stakeholders, developers, testers, managers and among clients to promote agility. Team size is usually kept small for easy communication. By daily communication, and meetings with stakeholders overall risk is minimized and adapt to changes can be made quickly if the customer disapproves of something. . *(Wikipedia, Agile Development, 2010)*
- **Embracing Change:** In agile change is embrace instead of being discouraged. Change occurs everywhere. Reqreuiments may change as technology or time change occurs. Agile successfully embraces changing requirements of clients by keeping the scope of each iteration short and by adaptive , continuous planning rather than detail predictive planning. Thus frequent inspection and adaption are the key factors for embraching change. (*Mishkin Berteig, 2005)*

e) Agile Planning and project management

The above principles of manifesto outlined the guiding rules for the success of software projects according to Agile development methodology. The common themes often presented with agile management is for the manager to promote the vision, more communication, no command control and focus on delivery activities. *(Craig Larman, 2003)*

The main plan of an agile project management routine is to submit deliverable in stages which delivery short delivery time. This can be regarded as the prime difference between Agile and IID methodologies where the delivery time in IID is substantially longer than Agile.(Wikipedia, Agile Management, 2010)

The agile planning is usually done only as needed and doesn't rely on heavy documentations like waterfall. Overall it encourages face to face communication instead of lots of documentation as a primary method of communication. *(Alister Cockburn, 2007)*

Even though the agile is a project management model, there is no strict requirement for a convention project manager who assigns tasks, because each team member signs up for his own task. The main idea behind agile teams is self organization and continuous self improvement.

f) Estimation in Agile Projects

In agile, estimation and feedback goes opposite to each other. If you cant estimate increase the feedback. Don't spend too much time on estimating. The goal of the estimates is to gain a general understanding of the cost of the system or product. This is used to determine whether it is economic to develop it. *(Estimation in Agile Projects, Christoph Steindl)*

Also relative estimation is encouraged instead of estimating features across a spectrum of unit lengths. Features are usually estimated in these categories such as days, story points, or hours. Relative estimation values working estimation ideas instead of precisely to the point estimates. Estimations are refined successively. If a feature exceeds an agreed maximum estimate, then it should be broken down further into multiple features. *(Agile 101, 2010)*

g) Flavours of Agile

Following we will summarize some of the key models of Agile. While each of the agile methods is unique in its specific approach, they all share a common vision and core values which was outlined in the manifesto and principles. They are all light weight process which values simplicity and adaptability.

All the various flavors of Agile mostly employ realistic practices and are simplistic in nature and have been modeled to be just barely good enough.

Some of the popular models along with their creators are

- Extreme Programming (Kent Beck)
- Scrum (Ken Schwaber, Jeff Sutherland, Mike Beedle)
- Crystal (Alistair Cockburn)

- DSDM (Arie van Bennekum)
- Feature-Driven Development (Jeff De Luca)
- Lean Development (Bob Charette)
- Adaptive Software Development (Jim Highsmith)

We will discuss two models in detail in the next section.

Specific Agile Models

Like mentioned above the various flavors of Agile differ in their approach but basically follows the same vision and purpose. All these different flavors has developed its own communities which are distinct groups with different practices but to be regarded as being agile they should follow the same broad philosophy of Agile. Over all we had been just describing the philosophy of Agile, but now we will try to describe some of the specific approaches with their own set of different practices.

SCRUM Methodology

Scrum is basically a rugby term which means restarting the game. Scrum methodology can be described as a process skeleton which has a general set of management guidelines and practices and some predefined roles. The scrum model is a management oriented model, which instead of placing more emphasizes on engineering practices it places more emphasizes on management practices and many people combine it with other models engineering practices or use it as a "wrapper" with other agile models such as XP.*(Martin Fowler, Flavors of Agile)*

There are lots of projects where scrum management model is integrated with XP , UP, and EVO practices where Scrum guides the overall process of development but by following practices specific to XP etc. One of Scrum's biggest advantages is that it is very easy to learn and requires little effort to start using

There are several implementations of systems in many different fields for managing the Scrum process, which range from cars to software packages with outstanding results. Its also heavily employed in the software industry in such big companies as Yahoo and Motrolla(Naftanaila Ionel, CAOTSPMM)

a) Life Cycle

The set of management practices of Scrum is very different from those of Waterfall such as requirements, implementations etc. The scrum life cycle is composed of four phases which are planning, staging, development and release. *(Craig Larman, 2003)* The Planning and staging can also be combined to call Pre-Game. The development may also be called mid game while release as Post Game. In the planning phase the project is planned as needed and high level design decisions are made. In scrum the iteration is usually called a Sprint which is fixed length of 30 days. Several sprints are required for release of a product. *(What is Scrum?, Marc Clifton, J. Dunlap)*

The product life cycle can be outlined as

Pre-Game: A product back log is established in the planning phase of the project which creates the vision, expectations and requirements. It also prioritizes all the features which are required by the customer as a general outline of the software. *(Craig Larman, 2003)* Note: This doesn't require to define the complete set of specifications upfront but just a blue print or vision of what is eventually required. Just enough requirements are prioritized in the subsequent staging phase to make it ready for first iteration which is also called a sprint backlog. Design and protypes are also considered. . *(Naftanaila Ionel, CAOTSPMM)*

Mid-Game: This can be actually regarded as the development phase of the lifecycle where actual sprints are held. Features from the sprint backlog are implemented over an iteration span of 30 days. The daily scrum meetings are a crucial part of this phase. A short 15 minute meeting is held each day by the scrum master , who asks questions such as What has been accomplished since last meeting? The obstacles in achieving the goal? What will be accomplished in the next meeting. *(What is Scrum?, Marc Clifton, J. Dunlap)*Sprint reviews are also done at the end of each iteration. At the end of iteration new functionality are added to the working increment of the software. However more sprints would be required for the completion of the project. Thus a sprint backlog is updated after each iteration by prioritizing items from the product backlog. Scrum meetings are usually short and encouraged to be stand up. *(Naftanaila Ionel, CAOTSPMM)*

End-Game: This is the closure of the project where project is brought to a close and a formal release of the software is made. Documentation and training on the software is also done during this phase. The system is deployed on the user end. *(What is Scrum?, Marc Clifton, J. Dunlap)*

b) Work products, roles and practices

The work products of scrum can be defined as follows *(Scrum Work Products, 2008)*

- Potentially Shippable Product Incremement
- Product Backlog
- Release Burndown Chart
- Sprint Backlog
- Sprint Burndown Chart
- Task Board

Each of this work product is used to manage the process of SCRUM. The main players in a scrum model are the scrum master who maintains the process, the product owner and the scrum team which is usually a very small cross functional team. The scrum master usually guides the team and is a leader but is also involved in the working team, i.e doing the work. The team doesn't have more than 6 people. *(Scrum Development, Wikipedia, 2010)*

The scrum model also has a different perspective towards seeing the people involved in the project which is called "Pigs and Chicken". Pigs are actual people involved in the project while chickens are usually stake holders for whom the software is being built *(What is SCRUM, 2010)*

Like mentioned in the beginning most of the practices in Scrum are borrowed from other agile more engineering centric models such as XP. However a specific scrum practice could be described as unit testing.

c) Values

The common values in scrum are usually inherited from the parent Agile philosophy where Interactions, working software, customer callobration, and change management are valued however in Scrum own perspective this values are defined as Commitment, Focus, Openness, Respect and Courage. The above managerial framework of Scrum is basically built on to provide and harness these values. *(XP and SCRUM values, 2010)*

d) Strength and Weaknesses

The strengths of SCRUM comes from the above mentioned practices and values. The simple managerial and evolutionary framework is its biggest strength also the ability of the team individuals to be self-managed and independent is a strong factor for success. The flexibility of Scrum life cycle allows the manager to manage the project variable as it grows. The sprints allows the project to change its deliverable anytime delivering the most appropriate and desired functionalities in a release. Also Its easy to implement, understand and encourages teamwork and transparency. However the scrum model is also be immune to some weakness such as *(Scrum pros and cons, 2010) and (Naftanaila Ionel, CAOTSPMM)*

- Scrum relies on minimal guidance within disciplines other than project management and

depends on team working with different roles each time. For eg a developer working as a QA and vice versa.*(CraigLarman , 2003)*

- Too much emphasizes on managerial aspect and not enough emphasizes on development aspects. *(Scrum pros and cons, 2010)*
- Direct emphasizes on user requirements because of which nonfunctional requirements of the system such as security, performance, clean documentation can get ignored.

Extreme Programming

Extreme Programming more commonly known as XP is one of several popular Agile Processes. Similar to Scrum XP emphasizes team work, collaboration and iterative development. However XP is popoular because of its engineering practices and emphasizes of responding to change in customer requirement even in the last stages of the life cycle. The five core values of XP just like Scrum are communication, simplicity, feedback, respect and courage. XP is well known for its Test Driven development practices where you are getting to test from day one and driving the development from testing and getting feedback from the tests. The customers are delivered the systems as soon as possible and from their suggestions further changes are made. XP employs certain set of productive practices who only put emphasizes on producing software, unproductive activities have been trimmed and thrown away to reduce costs and frustration of everyone involved. *(Extreme Programming, A gentle Introduction)*

The most astonishing thing about XP is we can eliminate requirements analysis, design, and testing along with all other phases as well forma documents. Xp emphasizes on face to face collaboration daily, which eliminates communication delays and misunderstandings that you no longer need separate phases. Thus all the activities of the phases are perform every single day, i.e analysis, design, code and test. Most of the time the phases are done simultaneously where different members are working on different activities associated with separate phases. Deployable software is produced every week to the customer which generates rapid feedback. In an iteration each phase acts as a feedback loop to refine plans quickly, which means that suppose if a defect is found in coding or testing of one iteration phase, we can use that knowledge to analyze requirements regarding that in subsequent iterations. *(The Art of Agile Development, James Shore, Shane Warden)*

a) Life Cycle

The XP life cycle can be defined as consisting of five distinct phases which are Exploration, Planning, Iterations to First Release, productionizing, and maintainence. *(Craig Larman,*

*2003)*The first phase exploration usually entails discovering features, estimates, feasibility etc. The business value story cards (features) are generated which are sufficient for the first released. The next phase of planning follows the exploration phase where the writing of story cards and estimates are taken further and prioritized. This process is also called planning game. The planning effort is most intense during first few weeks to getr the project into a stable phase, than for the remainder of the project customer constantly review to improve the vision and release plan to account for new events. First release stories and date of inceptions are agreed upon b/w customers and developers. For the next phase the user would pick stories to implement. This is also called *Release Planning* is a practice where the Customer presents the desired features to the programmers, and the programmers estimate their difficulty. With the cost estimates in hand, and with knowledge of the importance of the features. The developers would thus break the stories down into subsequent short tasks. Following the agile philosophy XP doesn't allow the timeboxed iteration to be extended, if the user stories are found to be too overworking for the current release, they are shorten instead of extending the time of the iteration. During the development continuous collaboration with customers on requirement details is emphasized. In the subsequent phases deployment and activities concerned with marketing and productionzing the product is carried out. *.(Art of Agile Development, James Shore, Shane Warden)*

b) Work products, roles and practices

The following phases which are Thinking, Collaborating, Releasing, Planning, and Developing each has XP practices which correspond to traditional phases . Keep in mind that xp uses iteration rather than phases. Teams perform everyone of these activity each week. Most are performed every day.*(Art of Agile Development, James Shore, Shane Warden)*

Some of the practices related to the these phases are

c) **Thinking**: Pair programming, energized work , informative workspace, root cause analysis, retrospectives
d) **Collaborating**: Trust, Sit together, Real Customer Invovement, Ubiqutious Language, Stand up meetings, Coding standards, Iteration Demo, Reporting
e) **Releasing**: Done Done, No bugs, Version Control, Ten Min Build, Continious Integration, Collective Code Ownership, Documentation
f) **Planning**: Vision, Rlease Planning, The planning Game, Risk management, Iteration Planning, Stack, Stories, Estimating.
g) **Developing**: Incremental Requirements, Customer Tests, TDD, Refactoring, Simple Design, Incremental Design and Architecture, Spike Solutions, Performance Optimization, Exploratory Testing.

The above mentioned practices in XP are so well recognized that they have also been employed in other models. Thus they can also be labeled as Agile practices rather than being only XP centric practices. A separate section for these practices has been written in this paper.

The main work product of XP are small releases which focuses on business value, the team produces the software in a series of small fully-integrated releases. The team may include testers, who help the Customer define the customer acceptance tests, so a release is a fully robust working product that pass all the tests the Customer has defined. *(What is XP?, XProgramming).* Some of the other work producdts could be defined to be the story cards which the user generates or writes, which can be anything from features, fixes, or non function requirements the user wants. An important point is XP stories are not uses cases. *(Craig Larman, 2003)*

c) Values

The founder of XP Kent Beck says in Extreme Programming Explained:

"This is one of the premises of XP. It is the technical premise of XP. If the cost of change rose slowly over time, you would act completely differently from how you do under the assumption that costs rise exponentially."

He defined the following values

1) Communication
2) Simplicity
3) Feedback
4) Courage

Values are the backbone of XP. It is the understanding of values that makes the practices reasonable. It is possible to implement the practices without caring about the values, however, you can take most of XP only if your organization values match the values of XP.*(Agile Software Development XP Values, 2010)*

d) Strength and Weaknesses

Most of the strengths of XP can be attributed to the general strength which comes from following the Agile model philosophy. However XP practices has also been heavily debated and criticized. Especially some XP practices like Pair Programming and continuous design has been a target of extreme criticism. *(Extreme Programming Refactored, Doug Rosenberg.*). Also XP has been noted for several potential drawback such as including problems with unstable requirements, no documented compromises of user conflicts, and lack of an overall design specification or document. Some of the other weaker points could be defined as*(Extreme Programming, Wikipedia, 2010)*

- Requirements are expressed as automated acceptance tests rather than specification documents.
- Requirements are defined incrementally, rather than trying to get them all in advance.
- Software developers are usually required to work in pairs.
- A customer representative is attached to the project. This role can become a single-point-of-failure for the project, and some people have found it to be a source of stress
- Requires too much of a cultural change to adopt the XP practices.
- User stories are hard to define for non-functional quality attributes, because agile is feature driven.

Agile Programming Practicies

Agile teams have found in practice that agile code is extensible, low-defect code with the simplest robust design that will work for the features currently implemented. It is well-factored and well-protected by unit tests. Among the agile methodologies, Extreme Programming (XP) goes into the most depth concerning how programmers can keep themselves and their code agile. It is increasingly common to find agile teams that have adopted non-XP customer practices but have adopted at least some of the XP programmer practices: *(Agile 101, 2010)*

a) Test-Driven Development

Thorough sets of automated units tests serve as a kind of net for detecting bugs. They nail down, precisely and deterministically, the current behavior of the system. Good Test-First teams find that they get subtantially fewer defects throughout the system lifecycle, and spend much less time debugging. Well-written unit tests also serve as excellent design documentation that is always, by definition, in synch with the production code. A somewhat unexpected benefit: many programmers report that "the little green bar" that shows that all tests are running clean becomes addictive. Once you are accustomed to these small, frequent little hits of positive feedback about your code's health, it's really hard to give that up.

b) Regular Refactoring

Refactoring is the process of clarifying and simplifying the design of existing code, without changing its behavior. Agile teams are maintaining and extending their code a lot from iteration to iteration, and without continuous refactoring, this is hard to do. This is because un-refactored code tends to rot. Rot takes several forms: unhealthy dependencies between classes or packages, bad allocation of class responsibilities, way too many responsibilities per method or class, duplicate code, and many other varieties of confusion and clutter.

c) Continous Integration

Continuous Integration (CI) involves producing a clean build of the system several times per day, usually with a tool, which uses Ant and various source-control systems. Agile teams typically configure CI to include automated compilation, unit test execution, and source control integration. Sometimes CI also includes automatically running automated acceptance tests . In effect, CI means that the build is nearly always clean.

d) Simple Design

In any agile context, simple design means, to paraphrase the poet Wallace Stevens, "the art of what suffices." It means coding for today's specified requirements, and no more. It means doing more with less. But this is not necessarily a natural inclination for us programmers.

e) Pair Programming

Pairing involves having two programmers working at a single workstation. One programmer "drives," operating the keyboard, while the other "navigates," watching, learning, asking, talking, and making suggestions. In theory, the driver focuses on the code at hand: the syntax, semantics, and algorithm. The navigator focuses less on that, and more on a level of abstraction higher: the test they are trying to get to pass, the technical task to be delivered next, the time elapsed since all the tests were run, the time elapsed since the last repository commit, and the quality of the overall design. The theory is that pairing results in better designs, fewer bugs, and much better spread of knowledge across a development team, and therefore more functionality per unit time, measured over the long term.

f) Sharing the codebase

By sharing the code base, as with pairing, reduces management's overall vulnerability to staff turnover, and makes it easier to deploy staff. If everybody more or less knows what is going on in all of the codebase, then you have more flexibility when figuring out who will do what in the next iteration. And if someone leaves the team, it is much less of a crisis; other team members can fill the gap pretty straightforwardly.

g) Coding standard

If programmers all adhere to a single coding standard (including everything from tabs vs. spaces and curly bracket placement to naming conventions for things like classes, methods, and interfaces), everything just works better. It's easier to maintain and extend code, to refactor it, and to reconcile integration conflicts, if a common standard is applied consistently throughout. The standard itself matters much less than adherence to it. Fortunately, modern IDEs make it trivial to apply many kinds of formatting, even after the fact.

h) Open Work Area

In Extreme Programming, programmers are expected to all work within earshot of each other in a single room full of workstations, and at least one customer representative is also encouraged to work with them in the room. And while "facilities issues" are notoriously problematic politically and emotionally, lots of teams have made open areas work. There are ways to encourage people to work in such open areas, without asking them to give up all of their private space or "space status."

Companies Employing Agile

In today's business world, with its ever-changing marketplace, many companies recognize the need to adapt and change in order to remain competitive. The old school methodologies are hardly of any use in the today's era of modern new product development such as Software Development. Thus many big companies have employed Agile since their inceptions and many other inexperienced small companies are migrating towards Agile, after their failures and issues with the waterfall model. We will see four cases of companies who are employing Agile, two can be regarded as huge multimillionaire dollar companies and others as small companies.

Big Companies Employing Agile

In this section we will see two such big companies who have employed Agile Development in most of their product management and development and are big pioneers of this model. They knew from beginning that success often depends on how you identify new opportunities and continue to change your company, that's why change is a driving force in their success and they embrace it instead of neglecting it.

Agile At Yahoo

Yahoo has employed Agile since a very long time and consider it as its most important assest. Infact Yahoo practices proves that Agile is a success in today's world. Yahoo! is a large enterprise with a $32 billion market cap and has one of the largest Agile implementations in the world. Throughout its history, the spirit of Agile has survived, often in surprising and unexpected ways. Whether being mandated from the executive-level or arising from self-motivated small teams, one theme is constant - Agile has embedded itself into the DNA of Yahoo!. And to the present day, Agile continues to emerge and re-emerge in many forms from the dedicated individuals who use the tools of Agile to create some of the most innovative user experiences on the Internet. *(Agile at Yahoo! From the Trenches)*

The primary Agile model employed at Yahoo is the scrum methodology. Although yahoo was a pioneer of Agile practices since beginning such as IID and incremental development, they adopted a specific model in around 2001 which was Scrum. A general survey was taken at Yahoo from 600 of the employs and 85% statistics say that they would continue using Scrum if the decision was left to them. Scrum has also been proven to reduce the development time in

Yahoo's product lifecycle. *(Excellent Data From Yahoo, 2006)*

a) *Adoption of Scrum*

Yahoo!'s first effort to meld its culture with a managed software development process began in 2002 with the release of a globally mandated waterfall process called the "Product Development Process" (PDP). Unfortunately, many teams simply ignored the process or, where they couldn't ignore it, paid lip service and made it look like they had adhered to the steps retroactively. The teams that did follow the PDP found that it was heavy, slowed them down, and added little real value. *(Rolling out Agile in a Large Enterprise, Gabrielle Benefield). At that time* There were many small teams with power to do pretty much anything with rather little discipline present. They needed more structure and consultants gave them 300 pages long heavyweight process. Everybody hated the previous model and it did not really work. Yahoo found scrum as a possibility to go back to their roots, while still having some structure in place. Nowadays a lot of new product development is done with Scrum at Yahoo which now encompasses more than 200 teams projects and over 1,500 employees in the US, Europe, and India *(Agile at Yahoo! From the Trenches).*

Gabrielle Benefields a senior Director of Agile Development at Yahoo , co-leading the company's large-scale corporate adoption of Scrum, captured key notes statstics during the company wide adoption of Agile. He documented that average velocity increase due to the Scrum adoption across the company was estimated conservatively at 35% per year (measured from team impressions since before Scrum velocity wasn't measured). In reality in many cases it was more like 300%-400%. Even under conservative estimate that made net development cost reduction of over $1 million a year and Return of Investment in transition and expensive trainers was about 100%.. *(Yahoo Scrum Adoption Lessons, 2008, Artem Marchenko)*

b) Result Statistics

Yahoo! went from being a small startup to a large enterprise company quickly. Initially in Yahoo the major problem in product development was project and team layer issues of planning, project management, release management, and team interactions. After the adoption of Scrum their initial commitment was to receive comprehensive Scrum training and to adopt all the standard practices described in Ken Schawabar "Agile Project Management with Scrum" and to complete atleast one scrum sprint initially just to see its result. However the results were positive and the two month result statistics were recorded. *(Rolling out Agile in a Large Enterprise, Gabrielle Benefield)*

5. Rate Scrum in 30 days relative to how the team was building products previously.

Result : 74 percent of respondents said Scrum improved thirty-day productivity, while 2 % said it had been worse and the remaining saying about the same

- Clarity of goals of what the team was suppose to deliver.

  Result: 80% Respondents said Scrum helped clarify team gols, while 6% saying it was worse than before

- Collaboration and cooperation with the team?
  Result: 89% said it was better, 0% said it was worse while remaining saying it was the same

- Business value of what the team produced in 30 days?

  Result: 64% of respondents felt Scrum improved the business value of their product at the thirty day mark. While 2% didn't agree

- Amount of time wasted/ work thrown out /cycles misused.

  Result: 68 percent of those surveyed said Scrum helped reduce the amount of time wasted and 19% said the opposite

- Overall Quality and rightness of what the team produced

  Result: 54% were positive , with 5% negative and the remaining people being neutral.

- How would you rate your overall feelings about using Scrum , taking into account the benefits, drawback, and work involved?

  Result: 77% percent of those surveyed had positive feelings about SCRUM.

Some of the other current statistics in Agile at Yahoo

c) Yahoo Agile practices, values and principles

The general practices, values and principles of Agile at Yahoo align with those with the philosophy of Agile Development and also with the management practices of Scrum. However over the years they have develop their own particular insights on believes and values which they consider are helping them achieve their goal with great success. The Agile team and

strategy continues to evolve. Their only constant factor is change. The Agile team has more than doubled in size and continues to grow. *(Rolling out Agile in a Large Enterprise, Gabrielle Benefield)*

Some of their principles summarized are as follows

- Coach. Don't Dictate.
- Privacy Is Important
- Design: Find Common Ground
- Align with Management
- Coach Deep, Not Broad
- Don't expect everyone to like it
- Fund the Coaching Team Adequately
- Implement Solid Engineering Practices
- The Organization Must Also Adapt

For a thorough understanding of these principles please see the research paper Rolling out Agile in a Large Enterprise by Gabrielle Benefield.

d) Conclusion

In an interview conducted by Kurt Mackie an online news editor with Gabrielle Benefield a pioneer of Agile at Yahoo, when asked whats the future of Agile in Yahoo, he enthusiastically replied.*(Interview with Gabrielle Benefield*, *2008)*

"At Yahoo, we would never want to mandate this process. I think the process is one that stands on its own and people choose to use it. Over time, it's become very dominant. One of the cofounders at Yahoo said that Agile has been one of the most positive things to happen to the company. Over time, to truly compete, I believe that Agile is the best thing for the company."

- Gabrielle Benefield

Agile At IBM

Agile has been a significant part of IBM software development since Agile came into the buzzword. IBM Company is probably one of the largest practitioners of the development practice, executive stresses , with an official saying the company itself probably has one of the world's largest adoptions of agile. IBM's Per Kroll, chief architect for IBM Rational Expertise Development & Innovation, cited the company's proliferation of agile practices among some of

its approximately 35,000 developers. *(IBM promotes agile development, Paul Krill)*

When Agile techniques were mainstream they held such promise for IBM that beginning in mid-2006 an explicit program was put in place to adopt these processes on a wide-scale basis throughout IBM Software Group, an organization with over 25,000 developers. *(Scaling Agile an Executive Guide, Scott W Wambler)*

The IBM experience at Agile was that changing the organization culture was their primary challenge when adopting Agile techniques at a large scale. They also believe this is the primary challenge in most organizations. The main reason is that the organizational culture reflects the people, your organizational goals, the way that people are organized, and the ways that they prefer to work – all of these issues are near and dear to the hearts of the people involved.*(Scaling Agile: An Executive Guide, Scott W Wambler)*

a) Agile tools at IBM

IBM believes that Agile software development can be significantly different from one organization to another. Agile is not a “one size fits all” proposition. Also other than employing Agile in their own practices IBm also provide agile application lifecycle management platform especially designed to support distributed development. These framework of tools to promote and expand the use of modern software engineering practices especially emphasizing iterative development are called IBM Rational Software and the IBM Global services have been working to help many customers apply these techniques within their environments . *(IBM Rational Software, 2010).* Many other tools to help developers embrace an Agile approach are available from HP and IBM, among others. IBM is also famous for creating the Rational Unified process methodology(which was initially developed by Rational Software) which is an iterative software development process.

One such software from IBM Rational Software product line is Rational Team Concert where Agile process awareness and methods are built right into Rational Team Concert software, so a team can immediately start using agile methods such as Scrum, the Open Unified Process (OpenUP) etc. A team can then customize the methods to suit its particular way of working, even while a project is still in progress. *(IBM Rational Team Concert, Bernie Coyne,)*

A problem with Agile is it most suitable with small and medium sized teams. The question is, how do you scale it?" . IBM's Jazz project, a collaborative development platform, is attempting to tackle that question. *(IBM employs Agile Methodology, 2007)*

IBM believes that using their commercial tools instead of point specific tools will affect your Agile requirement strategy better in the following ways. *(Agility@Scale: Strategies for Scaling Agile Software Development, Scott Ambler)*

- Geographical Distribution
- Team Size
- Compliance Requirement
- Domain Complexity
- Organization distribution
- Technical Complexity
- Organizational Complexity
- Enterprise Discipline

Also you can tailor your agile practices according to the situation you find yourself in through the use of their tools. For a detail explanation please see the reference mentioned above.

b) Result Statistics

The following statistics were measured inside IBM internal projects*(Agile and Secure Software Development, Andras R. Szakal)*

Environment: Thousands of Customers, 70 team members, IBM Rational Quality Management Products, IBM Rational method composer and Rational Unified Process

Agile Techniques Used: Use case based development approach, Frequent iterations, Continous Testing, Ongoing Customer involvement, RUP for process improvement

Results: 30% increase in developer productivity, design level defects cut in half, QA productivity increased by 20%, Test coverage increased by 30% , 200% ROI in first release

Interestingly, the Agility at Scale 2009 survey found that it was quite common for agile teams to be geographically distributed in some manner: The bottom line is that some organizations, including IBM, have been very successful applying agile techniques on geographically distributed teams. In fact, agile GDD is far more common than mainstream agile discussion seem to let on. *(Agility@Scale: Strategies for Scaling Agile Software Development, Scott Ambler)*

Also IBM noted that by making use of their rational software product line to help other companies they noted the following realistic improvements in the companies.*(Scaling Agile an Executive Guide, Scott W Wambler)*

Success Criteria	Year 1	Year 2	Year 3
Quality	3-5% fewer defects	3-5% fewer defects	3-5% fewer defects
Labor costs	2-3% improvement	4-5% improvement	4-5% improvement
Time to Value	5% faster	10% faster	5% faster
Delivered Fiunctionality	5% improved accuracy	5% improved accuracy	5% improved accuracy

c) IBM Agile practices, principles and Values

IBM gives their personal secret success plan for Agile in the following four valuable values.

- Collaborative
- Automated
- Empowered
- Flexible
- Adaptive

And they say *then and only then you will have fine wine (Agile and Secure Software Development, Andras R. Szakal)*

And their IBM Agile principles could be summarized as

1) Leverages the inevitability of change and generates new occasions for learning throughout the project.

2).Provides leadership to create an atmosphere in which the team determines its capacity and commits to shared goals.

3).Uses frequent interaction to move the whole product team toward its goals.

4).Engages with customers and other stakeholders throughout the project to generate continuous feedback.

5).Measures success in terms of delivering a flow of functional, proven stakeholder-valued capabilities.

6).Employs test-driven development and Agile Model Driven Development and is intolerant of defects.

7).Strives for relentless improvementof the product and the process.

The business case at Agile could be understood by the following internal view from IBM*(Agile and Secure Software Development, Andras R. Szakal)*

"Use continuous stakeholder feedback to deliver high quality consumable code through use cases and a series of short, stable, time-boxed iterations."

IBM success story with agile could also be realized by the following Tivoli workload scheduler(Agile and Secure Software Development, Andras R. Szakal)

"By developing our product in iterations and setting checkpoints at least monthly for all the development activities, we were able to make necessary changes to the architecture and avoid the pain of discovering such problems in the later phases of the development cycle."

Small Companies Employing Agile

Agile methods adoption and use is not straight forward as people think, it means that potential practicioners need to carefully assess whether they are exposed to the risk that can make agile model adoption problematics. This is the case with small software companies who are less able to absorb the impact of failed experimentation. In this section we will see two especially small companies who successfully employed Agile and faced success. One company is Little & Co on which I found several material and is document below, the other company is called Cprime and didn't have much material documents. Interested reader please see reference.

Little and Co Story

Litle & Co. is a leading payment management and processing platform for businesses that sell goods and services directly to consumers. The company's new technology and fresh approach to the development of payment systems have resulted in a next-generation payment processing platform that provides more services to merchants while simultaneously reducing their maintenance costs and giving them more flexibility and options*. (Litle & Co, 2010)* Founded in 2001, Litle & Co. has won a number of prestigious awards, including the No. 1 ranking on Inc. magazine's 2006 list of the 500 fastest-growing private companies in the United States. *(Accru Rev, Litle Success Story, 2008)*

a) Problems and Solutions

Since starting they have wanted to use the Agile Development especially the XP Programming model. Litle uses many of the XP practices including pair programming. They started with weekly iterations then moved to 2 weeks then 3 and are now monthly. Initially as the team was small consisting of 6 developers but over time they have grown to a development team of 35 including QA and a codebase that contains 50,000 files, the practices of Agile they were employing started having problems. (*Do It yourself Agile, An Agile Case Study, 2008)*

The initial problem they had was when the team they employed grew it became increasingly challenging to manage its software development Agile process with CVS. This was a major problem because they make use of world's major credit card and alternative payment networks, such as Visa, MasterCard, and PayPal, its mission-critical applications must be available 24/7, with no downtime. *(Accru Rev, Litle Success Story, 2008)*
The overlapping iterations were one of the major reasons that CVS wasn't scaling. In early 2007 Litle And Co deployed Accu Rev Software , David Tarbox, Director of software Development says. (*Litle & Co. Accelerates Agile XP Software Development Process with AccuRev, 2010)*

"Obviously, coming from CVS, we looked at Subversion, but it didn't solve the underlying architectural branch-based shortcomings. The AccuRev stream-based inheritance model allows us to do lightweight, incremental merges and track their history throughout the software development life cycle—both up and down the structure—automatically. We would have to do that manually in another tool. The biggest surprise was that AccuRev has allowed us to do things we couldn't conceive of with CVS—that has resulted in a greater benefit to the company than all the efficiency gains over CVS."

AccuRev is basically a process-centric software change and configuration management (SCCM) solution designed for today's Agile, complex parallel, distributed and offshore application development environments. AccuRev's stream-based architecture has been designed to ensure optimal team collaboration, improve software asset reuse, and accelerate the software development process by 30 percent.(*AccuRev - SCM, Agile Development and Release Management Solutions)*

By deploying AccruRev software they faced the following benefits*(Accru Rev, Litle Success Story, 2008)*

- AccuRev enabled Litle to easily scale its Agile/XP environment
- Multi-Stage Frequent Integration reduced build failures by 50 percent
- AccuRev eliminated the need to use labels, saving an hour off each build
- AccuRev eliminated two days of downtime each month by eliminating the single huge merge at the end
- AccuRev simplified the process of passing PCI and SAS70 Type 2 audits

Another problem Litle development teams ran into as they grew was integration of newly developed code into the existing codebase. Agile stresses a constant "build and integration" type of process. By using CVS they had to hold off the code base for integration until the very end, this one branch method, merging was costing the company much in time and money. To address this, in 2007, the company added the practice of Multi-Stage Frequent Integration to its implementation of XP. Prior to implementing Multi-Stage Frequent Integration, they would have to manually pore over all build and test failures to determine which change caused which failures. (*Do It yourself Agile, An Agile Case Study, 2008)*

b) Practices

Following are some of the key practices of Agile employed by Litle and Co. (*Litle & Co. Accelerates Agile XP Software Development Process with AccuRev, 2010)*

- More than 30 developers and QA people working at the same location
- Development Teams entirely co-located
- Each project, the company assigns a team consisting of multiple pair programming team
- Agile/XP software development since 2001
- A codebase that contains 50,000 files, including test cases
- Monthly iteration cycle
- Twelve overlapping iterations per year
- Litle's software is hosted at multiple off-site data centers, and the company upgrades its software every month like clockwork by using approximately six-week iterations.

c) Conclusion

Tim Litle, the founder of this company believes in four things for success first , we have to be good systems guys . Second, we have to know our market. Third, we have to know Visa and MasterCard regulations. And fourth, we have to know how to manage risk. (*How I did it : Tim Litle Chairman of Litle and Co, 2006*) Litle also sees its long track record of successful use of Agile as a competitive advantage: because they are on monthly releases, salespeople are able

to work with the product people to adjust the company's priorities to accommodate what makes the most sense for the business. (*Do It yourself Agile, An Agile Case Study, 2008)*

Agile adoption Survey and Statistics

In this section we will take a broad survey of the adoption of Agile in recent times and a statistical approach on research, historical results and evidence related to Agile. We will also see result statistics data on how the previous model failed in some large projects and some of the business cases in recent times promoting Agile and IID.

a) Waterfall Failure research statistics

Waterfall model was a response to code and fix development in the 1960's. The original author of waterfall model paper Winston Royce actually recommended an approach different than what evolved into the popular notion of waterfall . He recommends to do it twice. The project failure rate was very high for waterfall projects and all of the variations over the years.. The primary reason has been attributed to incomplete or changing user requirements.*(Waterfall Versus Agile Software Development, 2010)*

The Standish Group's Chaos Report is a landmark study of IT project failure. The Standish Group research shows a staggering 31.1% of projects will be cancelled before completion. Further results indicate that 52.7% of projects will cost over 189% of their original estimates.

According to the Standish Group In 1994, They estimated U.S. IT projects wasted $140 billion—$80 billion of that from failed projects—out of a total of $250 billion in project spending. This could be attributed for projects not being iterative and having "challenged" requirements with large deadlines. *(Standish: Project Success Rates Improved Over 10 Years, 2004*)

In a study of failure factors on 1027 IT projects in the UK only 13% didn't fail. Waterfall practices and project management with the waterfall model was the single largest factor for failure being cited in 82% of the projects, with an overall weighted failure influence of 25%.

In another study of 6700 projects it was found that four out of five key factors to project failure were associated with waterfall model, including inability to deal with changing requirements, problems with late integration and detailed upfront requirements. *(Craig Larman , 2003)*

Also it was found that in a study of projects features which were brought up in upfront specifications by hasty decisions, their usage were as follows *(XP, 2010)*

1) 7% Always

2) 45% Never
3) 13% Often
4) 16% Sometimes
5) 19% Rarely

The big step towards popularity for the waterfall model came with DoD STD 2167. The Department of Defense being the most frequent user of waterfall in the past. By Dod Std 2167 all the DoD projects were required to follow a waterfall lifecycle. A report on failure rates in a sample of earlier DoD projects concluded that 75% of the projects failed. *(Craig Larman, 2003)*

In April 1997, another survey was carried out by KPMG Canada focusing on IT project management issues to Canada's leading 1,450 public and private sector organizations. The main purpose was to outline the reasons behind the failure of Information Technology projects. Out of 1,450 questionnaires sent, 176 were analyzed. Of these, 61 % reported details on a failed IT project. Conclusion was all the factors for failure were related to waterfall model practices. Considering that an estimated $25 billion is spent on IT application development in Canada annually, the survey data indicated that unbudgeted IT project expenditures must run into the billions of dollars. (*Analysis of Software Development Processes, Dandriani)*

The end result was that eventually all managers realized that the prime reason for their failure rate was using a heavy weight model, which is they try to plan out a large part of a software project in great detail over a long span of time because they want to predict every conceivable project milestone which is not possible in Software Product Development.

b) Agile Migration Statistics

Many people have been practicing Agile methods and ideas even before when in 2001 they gave name to a collection of ideas and models as Agile. These models has been growing in popularity ever since and are common in usage now. Agile software development techniques have taken the industry by storm, with 76% of organizations reporting in 2009 that they had one or more agile projects underway. *(The Agile Scaling Model (ASM): Adapting Agile Methods for Complex, 2009*)

The importance of the agile approach is surely growing in the software development community. In a survey done in 2008 by Dr Dobb it was found that the agile adoption rate was 69% for that year, again 69% for the previous year 2007 and 65% in 2006. However Dr Dobb pointed out that he suspected stealth adoption where the team was doing Agile without the knowledge of senior management, so the adoption survey might be a little inaccurate. However he also said "*When I analyzed the adoption rates by role, I found that only 61.4 percent of developers thought they were doing Agile, whereas 78.2 percent of IT management thought so, the exact opposite of what I would've expected to see if stealth adoption was occurring. Based on these*

numbers, I suspect that developers and management have different criteria for what it means to be Agile, and that developers have set a higher bar for themselves". (Has Agile peaked, Dr Dobb, 2008)

Another research by Dr Dobb was done in 2007. (Survey Says... Agile Has Crossed the Chasm ,Dr. Dobb's Journal).

Some findings include: *(Agile Adoption Rate Survey Results: March 2007)*

1. 69% of respondents indicated that their organizations are doing one or more agile projects. Of those that hadn't yet started, 24% believed their organizations would do so within the next year.
2. 44% indicated a 90%+ success rate at agile projects, 33% indicated between 75 and 90%. It appears that agile seems to be working out.
3. Co-located agile projects are more successful on average than non-co-located, which in turn are more successful than projects involving offshoring.
4. 98.6% of agile teams worked adopted iterations, and 83% had iteration lengths between 1 and 4 weeks.
5. Smaller teams had higher success rates than larger teams.
6. 85% of organizations doing agile had more than one project completed, so it's gone beyond the pilot project stage in most organizations.

Another such approach was done by M&T in 2008 which summarized that only 13% of the companies weren't aware of Agile while 17% of the companies were using agile in all new projects. Some other interesting statistics are as(*Adoption of Agile Methods, 2010)*

	2008	2005
Not aware	13%	26%
Not using	13%	16%
Investigating	14%	14%
Analysed and rejected	4%	3%
Pilot projects	8%	4%
Partial implementation (adoption of some agile practices)	17%	17%
Partial deployment (some projects are using this approach)	14%	12%
Deployed (all new projects are using this approach)	17%	8%

Participants: 512 (232 in 2005)

This report also pointed out just like Dr Dobbs that Agile adoption is actually higher than reported in this study. Because Agile processes are often adopted at the grassroots level, they frequently fly below the radar. This makes it unlikely that a single IT decision-maker knows what methodology every development team is using down the link.

c) Success Rate of Agile Development

Data shows that IID is correlated with lower risk, higher productivity and lower defect rates than waterfall projects. Also one of the major life critical system was developed iteratively rather than using waterfall. Examples include the SUA space shuttle flight control software which was developed in 17 iterations, and the new Canadian air trafiical control system. IID is a central component of Agile model. Also a large study of failure and success factors in over 1000 UK IT projects found that 90% of the successful projects were less than 12 months duration, indeed that 47% were less than 6 months. Breaking the large project into small components and milestones is another hallmark of Agile. Another study by Boehm and Paccio showed that a typical software project experienced a 25% change in requirements. IID and evolutionary requirements study of 107 projects showed that only 18%of the projects tried to complete the requirements in a single early step, 32% used two cycles of requirements refinement, and in 50% of projects that requirements analysis was completed over three or more iterations. Embracing Change is the third most important point of Agile development. This data basically implies that IID and evolutionary practices that emphasize adaptiblity, embrace to change, and steps to provoke constant collaboration are consistent with success. *(Craig Larman, 2003)*

Also for some concrete statistics results for the effectiveness of Agile Software Development compared with other traditional models were summarized by Dr Dobb in the artlcle *Has Agile Peaked in June 2008*. Some of the results are.*(Agile Adoption Rate Survey Results: February 2008)*

Productivity: Much higher(22%), Somewhat higher (60%), No Change(13%), Somewhat lower (13%), Much Lower (1%).

Quality: Much higher(29%), Somewhat higher(48%), No Change (14%), Somewhat Lower(6%), Much Lower(3%)

Business Stake Holder Satisfaction: Much Higher(31%), Somewhat Higher(47%), No Change(15%), Somewhat Lower(4%), Much Lower(3%).

Cost Of System Development: No Change (40%), Somewhat Lower(32%), Much Lower(5%), Somewhat Higher(18%), Much Higher(5%).

Another of interesting Dr Dobb Article Defining Success which dealt with success with companies using Agile was published in Dec 2007.

Some of the interesting findings are: *(IT Project Success Rates Survey Results: August 2007)*

1. Respondents reported that Agile software development projects have a 71.5% success rate, traditional projects a 62.8% success rate, and offshored software development projects a 42.7% success rate.
2. Respondents with only Agile experience, 15 out of 586, indicated a success rate of 84.3%, respondents with traditional-only experience (164 of 586) reported a success rate of 66.5%. Respondents with both Agile and traditional experience (336 of 586) indicated success rates of 70.9% for Agile and 61.1% for traditional.
3. Time: 61.3% of respondents believe that delivering when the system is ready to be shipped is more important than delivering on schedule.
4. Money: 79.6% of respondents believe that providing the best ROI is more important than delivering under budget.
5. Scope: 87.3% of respondents believe that meeting actual needs of stakeholders is more important than building the system to specification.
6. Quality: 87.3% of respondents believe that delivering high quality is more important than delivering on time and on budget.
7. Staff: 75.8% of respondents believe that having a healthy workplace is more important than delivering on time and on budget.
8. When asked to prioritize the five criteria listed above, quality is the most important factor followed by scope, time, staff, and money respectively.
9. 40.9% of respondents considered cancelling a troubled project to be a success.
10. 68.6% of respondents had been on a project which they knew was going to fail right from the very start

Another research done by Scott on Dr Dobb results, a proponent of Agile at IBM shows two important factors related to team location and success percentage. (*Results of Agile Adoption Survey 2008)*

Team location Success percentage: Co-located Team 83% Distributed teams but physically reachable 72% Distributed across geographies 60%

Responses from the survey suggested that adopting Agile is a low risk decision. The following statistics validates the effectiveness of Agile Software Development over Traditional approaches. (*Results of Agile Adoption Survey 2008)*

Factor	**Improved**	**No Change**	**Worsened**
Productivity	82%	13%	5%
Quality	77%	14%	9%
Stakeholder Satisfaction	78%	15%	7%
Cost	37%	40%	23%

Also according to Standish Group who are well known for publishing statistics on project failure (CHAOS reports), it summarizes that project success rates has improved over 10 years. It says *(Standish: Project Success Rates Improved Over 10 Years, 2004)*

"The 10th edition of the annual CHAOS report from The Standish Group, which researches the reasons for IT project failure in the United States, indicates that project success rates have increased to 34 percent of all projects. That's more than a 100-percent improvement from the success rate found in the first study in 1994.

Asked for the chief reasons project success rates have improved, Standish Chairman Jim Johnson says, "The primary reason is the projects have gotten a lot smaller. Doing projects with iterative processing as opposed to the waterfall method, which called for all project requirements to be defined up front, is a major step forward.""

d) Evidence that Agile Software Scales

This section basically highlights how agile software development scales with many factors not just team size. There are people that believe Agile works better for smaller projects and Waterfall works better for more complex projects. In the early days of Software development, the applications were smaller in scope and relatively straightforward. Today, the picture has

changed significantly and organizations want to apply agile development to a broader set of projects. Some of the factors of scaling could be*(The Agile Scaling Model (ASM): Adapting Agile Methods for Complex, 2009*)

- Team Size
- Georgraphical Distribution
- Enterprise discipline
- Organizational complexity.
- Technical complexity
- Organization distribution
- Domain complexity

In 2009 Agile practice surveys collected some interesting facts regarding Agile and Scalability statistics. But sometimes these scaling factors also introduce complexities which the team must overcome*. (Evidence that Agile Software Development Scales, 2008)*

Agile and Team Size: How many IT people were on your Agile development Team?

Results: 34% in 1-5 teams, 34% in 1-6 teams, 11-20 in 15% teams, 21-50 in 10%, 51-100 in 4%, 101-200 in 1%, 201-500 in 1%, 501+ in 1%.

Team size increases communications and organizational risk which is an indication of both technical complexity and organizational complexity.

Agile and Geographical Team: How distributed were the IT people on your agile development team?

Results: Same Building (17%), Colocated(42%), Some very distant(29%), Within Driving Distance(13%), Don't know(1%).

Success rates by paradigm and distribution level: Project success rate by paradigm and Distribution Level.

Results for Agile development: Average(70%), Co-located(79%), Near Located(73%), Far Located (55%).

Agile and Regulatory Compliance: Does your agile team have to comply with industry regulations?

Results: Yes(33%), No(60%), Don't Know(7%).

Agile and Technical Complexity: Agile and Legacy assets

Results: Working with legacy in some way(78%), Integrating with legacy systemsI(57%), Evolving Legacy Systems(51%), Working with Legacy Data(45%).

Agile and Organizational Complexity: Does your agile development team follow a CMMI compliant process?

Results: No(78%), Yes(9%), Don't Know(13%).

IBM identifies two steps for agile to scale which are *(The Agile Scaling Model (ASM): Adapting Agile Methods for Complex, 2009*). These two steps are advocated by IBM Rational.

- The first step in scaling agile approaches is to move from partial methods to a full-fledged, disciplined agile delivery process.
- The second step to scaling agile is to recognize your degree of complexity

Agile Project Management practices

The selection, procurement, and deployment of a project management system is fraught with risk in exchange for significant business and financial rewards. Agile project management deems to reduce the risks through employing successful methods and practices. Managing aan Agile project is not the same as managing any other large project. A manager has to understand certain principles and practices for successfully gaining value from an Agile project. Some basic principles which should be the core emphasize of a project manager as defined previously are *(Agile Project Management Methods for ERP, Glen B. Alleman)*

- Increased participation by the stakeholders.
- Incremental and iterative delivery of business value.
- Maximum return on assets using a real options decision process.

Now we will see some other core practices of an Agile project manager.

a) Vision and velocity of a project

Velocity of a project is defined as a measure for delivering business value. It's a simple concept which means the amount of work done by a team per iteration of a project. A project manager should estimate velocity and track it for proper project management because it has proven to provide tremendous insight/visibility into project progress and status. Its beneficial to use a velocity chart through the lifetime of the project so a vision and plan for further iterations could be made. Initially the velocity might fluctuate, but after some iteration it is going to be stabilized and be constant through out the release. *(Agile 101, 2010)*

b) Feature Planning and FBS

A feature is defined as a unit of work in Agile. A feature planning should take care of following points when defining a feature

- It should deliver business value
- It should be small enough to fit inside iteration.
- It should be estimable
- It should be testable

Each feature should be planned, estimated and prioritized for a release. Feature breakdown structure(FBS) is similar to Work break down structure(WBS) of waterfall. Its main function is to start out with features that are large and then break them out into smaller features over time.

c) Risk Management Planning

Risk Management processes may have the air of a traditional, process-driven project management activity. However, agile methods are great risk reduction vehicles, and are actually very well aligned for rapid risk identification and reduction. The iterative nature of agile projects allow us to tackle high risk areas of the project sooner rather than later. This gets problems out in the open while there is still plenty of time and budget to work on them and reduces the effort invested in work that needs scrapping. The risk management process outlined in PMBOK centers more on waterfall model, but the basic mechanics are the same, however the process of actually following the practices and how they integrate into the lifecycle are different. Some of the strategies for handling risk in Agile are *(Agile Risk Management, 2010)*

- Actively attack the risks before they can impact the project.
- Maintain a risk burn down graph
- Listen for Potential Risks at the Stand-Up

d) Iterations and Release planning

Release planning is briefly lay out the whole project for that particular release. A release basically delivers a subset of the working software. Release typically range between 2 and 12 months. For longer releases, it may make sense to break it down into several sub-releases. A release consists of many iterations. Each iteration is usually 1 to 3 weeks long. In a release plan usually four things are accounted for ; scope, resources, time and quality. A release meeting is usually held to discuss these variables and formulate a release plan. Release planning meetings typically last between 4 and 8 hours. The plan is constructed from previous team velocity per iteration. *(Release Planning, Extreme Programming , 2010)*

e) Performance Planning

The purpose of performance planning is to ensure the goal or target is identified and that all stakeholders see the feasibility of the project. A study showed that 80% of performance problems reside in the environment, such as processes and systems, so ensure the problem is really a learning/training problem, not some other performance problem. *(Planning in Agile Learning Design, 2009).* In an agile process performance planning is usually adapted and answered by asking following questions "How are we doing?" and "Are we going to make it?" The nature of iterative development requires these questions to be asked at two levels: for each iteration and for each release. We should make use of several progress charts such as burn down chart, scope creep chart, release burn down chart to measure performance and then adapt to it. (*Agile 101, 2010)*

Agile Development Suites

There is no single recipe for practicing agile development. Every team or company might have their own flavor of following the philosophy of Agile. However all the companies do agree on a common set of practices of philosophy. Implementing Agile practices can be automated by various tools and development suites. Development suites and tools can ease and hasten the process of software development in a team. Software to help with Stand-ups, continuous integration, pair programming, code review and continuous improvement can be underpinned in a single framework of suite. There exists many suites which promote Agile; we will example some of them in this section.

a) Java Agile SDLC support

The software development cycle or SDLC is basically the process you follow to produce working software. This naturally involves coding, but there is more to building an application than just

writing code. Usually all software developers use a build environment to build and compile their code, however now a days build environment do a lot more than just compile and build the project, They provide facilities to include a version control system, have an issue management system, a continuous integration server to integrate frequently, testing(unit testing, integration testing) , deployment server, refactoring services, and lots of quality metric tools. As can be seen that most of this facilities corresponds directly to Agile practices. These third party tools can be integrated into the build tool and invoked when desired or can be automated. In this section we will see some of the java open source tools which facilities the above described process. *(Java Power Tools, John Ferguson Smart)*
Two common build environments which java provides are Maven, and Ant. Maven is a declarative project management framework. However ant is more of action oriented task based build tool. Both have their merits and demerits and can be used to practice Agile. Some of the open source tools related to practicing Agile are. *(Java Power Tools, John Ferguson Smart)*

Build Tools: Ant, Maven
Version Control Systems: CVS, SVN
Continous Integration: Continuum, Hudson, Cruise Control, LuntBuild, Hudson
Unit Testing: Junit, TestNg
Integration, function, load and performance testing: JMeter, SoapUI, DBUnit, StrutsTestCase
Qaulity Metric Tools: Check Style, PMD, Find Bugs, Mylyn, Jupiter
Issue Management Tools: Bugzilla, Trac

b) AccruRev Agile Cycle Suite

Accrurev is a company which offers comprehensive solutions for successfully scaling Agile processes across the enterprise. Their most successful tool is AgileCycle, which is a fully-integrated Agile Application Lifecycle Management (Agile ALM) product suite including award winning Agile tools for Project Management, Configuration and Process Management, Continuous Integration, and Deployment to Production. AgileCycle is designed to improve time-to-market, product quality, and cost effectiveness for large and small software projects. AgileCycle is available exclusively from AccuRev, Inc. For more detail please see the link *(http://www.accurev.com/agilecycle.html)*

c) Jira Studio

Jira studio is a development studio developed by an Australian company Atlassin that are well known for developing software tools and collaboration tools. The Jira studio can be used by project managers, developers, testers, and other members of the team. It provides a one stop solution to all the problems of a development team such as integration, hosted development environment, bug management, issue tracking, revision control, repository reviewer, wiki and code review. Atlassin believes that Jira combine a clean, fast interface for capturing and organizing

issues with customizable workflows, Open Social dashboards and a pluggable integration framework, JIRA is the perfect fit at the center of your development team. For more information see the following link (*http://www.atlassian.com/software/jira/)*

d) Thought Works Mingle

Thoughtworks is a global IT company headquarter in Chicago. They are a pioneer in practicing Agile methods and practices and believes themselves to be the world leaders in Agile Software Development. They have a whole faction called Thoughtworks Studio which is the Agile software tools development and training division of Thoughtworks. Their agile products and services are used by 250+ organizations worldwide to get the most out of Agile and become optimally business-responsive. Their key Agile management tools is Mingle which is an Agile project management software. Mingle helps with Flexibility in adapting agile, Collaboration, Visibility of current situation to adapt a company's process to Agile best practices, and helps your teams stay on top of evolving requirements in collaboration with business user. Another Agile tool they offer is called Agile ALM which covers both management and engineering aspects of Agile Software Development. For more information see the following link *(http://www.thoughtworks-studios.com/)*

e) Confluence for Collaboration

Another key software provided by Atlassian is Confluence, this is not a whole development suite helping in Agile models but is specifically tailored for helping in continuous collaboration among the team and customers. They provide extensive plugins to help people work better together and share information effortlessly. You can choose from a set of 200 plugin to customize your needs for collaboration among team members. Atlassian says that Confluence is trusted by 8000 businesses, *and* is the leading collaboration software and enterprise *wiki* for intranets and knowledge management. For more information see the following link. (*http://www.atlassian.com/software/confluence/)*

References

1) Agile and Iterative Development, A managers approach by Craig Larman
2) "Software Project Management Practices: Failure Versus Success," Capers Jones (http://www.stsc.hill.af.mil/crosstalk/2004/10/0410Jones.html)
3) "Agile Software Development," Alister Cockburn, Addison-Wesley.
4) "Agile Methodologies", Venkat Subramaniam http://www.agiledeveloper.com/presentations/AgileMethodologies.pdf
5) Agile Alliance, 2010 (http://www.agilealliance.com)
6) Agile Manifesto , 2010 (http://www.agilemanifesto.org)
7) Wikipedia, Agile Management , 2010 (http://en.wikipedia.org/wiki/Agile_management)
8) (Agile 101, 2010) http://www.versionone.com/Agile101/Agile_Development.asp
9) Wikipedia, Agile Development, 2010 http://en.wikipedia.org/wiki/Agile_development
10) Estimation in Agile Projects Christoph Steindl http://www.agilealliance.org/system/article/file/1421/file.pdf
11) Change is Natural, Embrace Change Mishkin Berteig http://www.agileadvice.com/archives/2005/05/change_is_const.html
12) Martin Fowler, Flavors of Agile http://martinfowler.com/articles/newMethodology.html#FlavorsOfAgileDevelopment
13) http://en.wikipedia.org/wiki/Scrum_%28development%29
14) *(What is Scrum?, Marc Clifton, J. Dunlap)*
15) http://www.codeproject.com/KB/architecture/scrum.aspx
16) *(Scrum Development Process, Jeff Sutherland,)* http://www.jeffsutherland.com/oopsla/schwapub.pdf
17) XP and Scrum practices http://www.slideshare.net/nashjain/xp-and-scrum-practices
18) Scrum pros and cons
19) http://sumedhsays.wordpress.com/2007/08/01/scrum-pros-and-cons/
20) CRITICAL ANALYSYS OF THE SCRUM PROJECT
21) MANAGEMENT METHODOLOGY, Nftnil Ionel
22) http://steconomice.uoradea.ro/anale/volume/2008/v4-management-marketing/077.pdf
23) Extreme Programming, A Simple Introduction
24) www.extremeprogramming.org
25) *(The Art of Agile Development, James Shore, Shane Warden)*
26) Extreme Programming, What is XP? http://xprogramming.com/book/whatisxp/
27) Agile Software Development, XP values http://agilesoftwaredevelopment.com/xp/values
28) Extreme programming, wikipedia

http://en.wikipedia.org/wiki/Extreme_Programming#Controversial_aspects
29) Agile Advice Most Popular - 2006/12
30) How Two Hours Can Waste Two Weeks
31) Performance Goals - The Wisdom of Teams
32) 8 Team Room Tips
33) The Seven Core Practices of Agile Work
34) The Case for Context Switching
35) Intro to Scrum - and Excellent Data from Yahoo!
http://www.agileadvice.com/archives/2006/09/intro_to_scrum.html
36) *Yahoo Scrum Adoption Lessons, 2008, Artem Marchenko)*
http://agilesoftwaredevelopment.com/blog/artem/lessons-yahoos-scrum-adoption
37) Lessons from a Yahoo Scrum Rollout, Interview with Gabrielle Benefield, 2008
http://campustechnology.com/articles/2008/02/lessons-from-a-yahoo-scrum-rollout.aspx
38) Rolling out Agile in a Large Enterprise by Gabrielle Benefield.,
http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/hicss/2008/3075/00/3075toc.xml&DOI=10.1109/HICSS.2008.382
39) IBM employs 'agile'
methodologyhttp://www.computerworlduk.com/technology/development/software/news/index.cfm?newsId=6286
40) IBM promotes agile development
http://www.infoworld.com/d/developer-world/ibm-promotes-agile-development-953
41) (Agile and Secure Software Development, Andras R. Szakal) https://buildsecurityin.us-cert.gov/swa/presentations_09/Day%203%20-%20SZAKAL%20-%20Agile%20and%20Secure%20SwA%20Forum%20v1.1.pdf
42) IBM Rational Team
Concertftp://public.dhe.ibm.com/common/ssi/pm/rg/n/rao14010usen/RAO14010USEN.PDF
43) (Agility@Scale: Strategies for Scaling Agile Software Development, Scott Ambler)
https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/agile_requirements_at_scale?lang=en
IBM Agile Rational Software, http://www-01.ibm.com/software/rational
44) The Agile Scaling Model (ASM): Adapting Agile Methods for Complex
Environmentsftp://ftp.software.ibm.com/common/ssi/sa/wh/n/raw14204usen/RAW14204USEN.PDF
45) Agile Adoption Rate Survey Results: March
2007http://www.ambysoft.com/surveys/agileMarch2007.html
46) IT Project Success Rates Survey Results: August 2007
http://www.ambysoft.com/surveys/success2007.html
47) Results of Agile Adoption Survey 2008http://www.infoq.com/news/2008/05/agile-adoption-survey-2008
48) (XP, 2010) http://xp2003.org/index.html
49) (Standish: Project Success Rates Improved Over 10 Years, 2004)
http://www.softwaremag.com/L.cfm?Doc=newsletter/2004-01-15/Standish
*50) Waterfall Versus Agile Software Development*http://www.nonstopportals.com/blog/168-

waterfall-versus-agile-software-development-

51) *Analysis of Software Development Processes*http://dandriani.wordpress.com/2010/01/06/41/
52) *AccuRev - SCM, Agile Development and Release Management Solutionshttp://www.accurev.com/accurev.html*
53) *(Litle & Co, 2010) http://www.litle.com/what-we-do*
54) *Litle & Co. Accelerates Agile XP Software Development Process with AccuRev* http://www.accurev.com/accurev-litle-success-story.html
55) *(Accru Rev, Litle Success Story, 2008)* http://www.accurev.com/scm-case-studies/litle-success-story.pdf
56) *Do It yourself Agile, An Agile Case Study* http://damonpoole.blogspot.com/2008/05/agile-case-study-litle-co.html
57) *How I did it : Tim Litle Chairman of Litle and Co, 2006* http://www.inc.com/magazine/20060901/hidi-litle.html
58) *Agile Risk Management, 2010* http://leadinganswers.typepad.com/leading_answers/2007/09/agile-risk-mana.html
59) *(Release Planning, Extreme Programming , 2010)* http://www.extremeprogramming.org/rules/planninggame.html
60) *(Planning in Agile Learning Design, 2009)* http://bdld.blogspot.com/2009/11/planning-in-agile-learning-design.html
61) *(Agility@Scale: Strategies for Scaling Agile Software Development, Scott Ambler)* https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/agile_requirements_at_scale?lang=en
62) Adoption of Agile Methods http://www.methodsandtools.com/dynpoll/oldpoll.php?Agile2
63) Evidence that Agile Software Development Scaleshttp://www.enterpriseunifiedprocess.com/essays/agileScales.html
64) (Agile and Secure Software Development, Andras R. Szakal)

https://buildsecurityin.us-cert.gov/swa/presentations_09/Day%203%20-%20SZAKAL%20-%20Agile%20and%20Secure%20SwA%20Forum%20v1.1.pdf
65) IBM Rational Team Concert ftp://public.dhe.ibm.com/common/ssi/pm/rg/n/rao14010usen/RAO14010USEN.PDF
66) CPRIME Case Study, http://www.wiredrive.com/blog/wp-content/uploads/2010/05/WD_Scrum_Case_Study_cPrime.pdf