Agile Web Development and Risk Management

Jason Owens, jason@jasonowens.com

Last Updated: 1/15/07

Contributors: Owens, Jason, jason@jasonowens.com

The formatting and minor edits of this document have been updated since its original creation.

Table of Contents

Abstract

This paper provides an overview of what is meant by Agile development methods, light processes, and provides examples. It also provides information on more traditional "waterfall" approaches to development, heavy processes, and comparisons of heavy and light methods. Current opinions and available data are provided. Risk management issues are presented relative to Agile methods. Benefits, risks, and challenges of Agile methods are also discussed, as well as what to evaluate when considering using Agile Methods and how to potentially integrate Agile methods into your organization.

Introduction

Purpose of this Paper

The purpose of this paper is to provide an overview of what is meant by Agile development methods, light processes, and provide examples. It is intended to provide some basic information to developers and managers that may have little experience or knowledge of what Agile Methods are.

This paper also provides information on more traditional "waterfall" approaches to development, heavy processes, and comparisons of heavy and light methods. Research on various opinions and perspective of different methodologies are presented in an objective manner, and potential benefits, risks, and challenges of Agile methods are discussed. Risk management issues are presented relative to Agile methods and reflect on how Agile methods may help with managing risks.

Finally, information regarding what to consider when evaluating Agile Methods and how to potentially integrate this new process into your organization and customer (if appropriate) is also presented. For the purpose of this paper "development practices" and "project management methodologies or processes" will be used interchangeably.

Questions to Answer

This paper will attempt to answer the following:

- What is meant by Agile software development?
- What are its strengths? Its weaknesses?
- In what situations is it beneficial to use Agile?
- How does Agile impact Risk Management?
- How can Agile be introduced into an organization?
- How can Agile be introduced to a customer?

Challenges with Empirical Data

Agile methods are relatively new in their current form compared to other software development and project management methodologies. As such, very little in the way of empirical research has been done, and research to support claimed benefits are somewhat lacking (Erickson et al., 2005.) For that matter, there has been a lack of research in general with regards to Agile Methods and Information Systems (Aydin et al., 2005.)

Agile methods do however have roots in *Kaizen* or continuous improvement efforts, as well as other disciplines found in other project management methodologies such as quality management. For the purpose of this paper we can compare Agile to other practices that have been in use for longer periods of time.

Methodologies Overview

Agile Development

What is Agile development

What is Agile development? Simply put, Agile is a general term applied to a collection of similar development practices. At a basic level these practices share common principles and goals: projects broken down into smaller sub-projects, smaller teams of skilled, empowered, tight-knit developers, regular "iterations" of software releases that occur very frequently, accelerated deadlines, high levels of constant customer involvement, integrated testing, flexibility, and very minimal modeling or documentation with regards to project management. Each iteration produces some type of functional code that serves a business purpose and is put into production.

For example, in website development the first iteration might produce a new domain name registration and a splash page for a new company. The second iteration provides the ability to search for some upcoming products. The third iteration may be an update to the website and publishing of content, and a fourth iteration a shopping cart feature.

Agile development is a social, team-based effort where the role of project management is decentralized. Teams are organic, led by a project guiding vision (simple mission statement), and have a high degree of contact with the customer.

Adaptive Methods

Agile can be considered an Adaptive method. Wikipedia.org defines Adaptive methods as the following:

> Adaptive methods focus on adapting quickly to changing realities. When the needs of a project change, an adaptive team changes as well. An adaptive team will have difficulty describing exactly what will happen in the future. The further away a date is, the more vague an adaptive method will be about what will happen on that date. An adaptive team can report exactly what tasks are being done next week, but only which features are planned for next month. When asked about a release six months from now, an adaptive team may only be able to report the mission statement for the release, or a statement of expected value vs. cost.

These practices are sometimes referred to as "light" or "lean" processes. There is some difference between Agile and Lean. The latter can be considered more of a management approach stemming from manufacturing and production systems relating to improving productivity and reducing waste, whereas Agile is considered to be more of a philosophy and is specific to software projects. Many people regard Agile to be the philosophical antithesis of the Project Management Institute's favoritism towards plan-centric management. (Scheer, 2005.) *The Principles of the Agile Manifesto* can be found in the appendix of this paper.

Flavors of Agile

Most all the research documents referenced in this paper mention different types of Agile processes. Extreme Programming (XP) for example is citied frequently as one of the most widely used (Lindstrom & Jeffries, 2004.) The following can be considered Agile development practices (Erickson et al., 2005) Definitions originate from wikipedia.org.

Practice	**Definition**
Extreme Programming (XP)	It prescribes a set of day-to-day practices for developers and managers; the practices are meant to embody and encourage particular values. Proponents believe that the exercise of these practices, which are software engineering best practices taken to "extreme" levels, leads to a development process with the qualities prized by Agile Manifesto signatories. This makes Extreme Programming the most prominent of several agile software development methodologies used to create software. Agile methodologies prioritize adaptability to changing requirements over the project predictability valued by more traditional methodologies.
SCRUM	Has been called a "hyper-productivity tool", and has been documented to dramatically improve productivity in teams previously paralyzed by heavier methodologies…Its intended use is for management of software development projects, and it has been successfully used to "wrap" Extreme Programming and other development methodologies. Scrum is facilitated by a ScrumMaster, whose primary job is to remove impediments to the ability of the team to deliver the sprint goal. The ScrumMaster is not the leader of the team (as they are self-organising) but acts as a productivity buffer between the team and any destabilising influences…A key principle of scrum is its recognition that fundamentally empirical challenges cannot be addressed successfully in a traditional "process control" manner. As such, scrum adopts an empirical approach - accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team's ability to respond in an agile manner to emerging challenges.
Adaptive Software Development (ASD)	A software development process that grew out of rapid application development work...ASD embodies the principle that continuous adaptation of the process to the work at hand is the normal state of affairs. ASD replaces the traditional waterfall cycle with a repeating series of speculate, collaborate, and learn cycles. This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, timeboxed, risk driven, and change tolerant.

Feature-driven Development (FDD)	FDD is a model-driven short-iteration process that consists of five basic activities…An iterative and incremental software development process…FDD blends a number of industry-recognized best practices into a cohesive whole. These practices are all driven from a client-valued functionality (feature) perspective. Its main purpose is to deliver tangible, working software repeatedly in a timely manner.
Dynamic Systems Development Method (DSDM)	A method that provides a framework for Rapid Application Development (RAD), supported by its continuous user involvement in an iterative development and incremental approach which is responsive to changing requirements, in order to develop a system that meets the business needs on time and on budget…DSDM focuses on Information System projects that are characterized by tight timescales and budgets…DSDM includes 3 phases which are pre-project phase, project life-cycle phase, and post project phase. There are 5 stages within its project life-cycle phase. The 5 stages are feasibility study, business study, functional model iteration, design and build iteration, and implementation.
Agile Modeling (AM)	A practice-based methodology for effective modeling and documentation of software-based systems. Simply put, Agile Modeling (AM) is a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and light-weight manner. Agile Modeling is a supplement to other Agile Methodologies such as Extreme Programming (XP) and Scrum.
Agile Unified Process (AUP)	A simplified version of the Rational Unified Process (RUP). It describes a simple, easy to understand approach to developing business application software using agile techniques and concepts yet still remaining true to the RUP. The AUP applies agile techniques include test driven design (TDD), Agile Modeling, agile change management, and Database Refactoring to improve your productivity.

Benefits and Risks of Agile

There are several benefits to using Agile methodologies. The high level of customer involvement and prioritization of deliverables helps to ensure that the right things are being produced and in the right order. Results are seem immediately once an iteration is complete. Project deliverables can change or be added relatively simply. Product or project problems can be discovered early in the process without previous significant investment. The way a team and work is organized allows for flexibility and for junior developers to learn from more experienced ones, as well as removing the dependency on one resource for project management. And the high level and frequency of testing helps to eliminate bugs early and often and simplifies bug fixing at the end of the project.

Agile is not without its drawbacks and risks however. It can be said that in order for an Agile team to be successful it needs to consist of very skilled senior developers (Hudson, 2005.) Team members all need to fulfill various roles interchangeably. The team is most effective when it consists of people who are jacks-of-all-trades and masters of a few and in organizations that are flexible and not hierarchical (Braun, 2005.) These types of people might not always be available or in the numbers needed for an effective team. Given the lack of emphasis on up-front documentation, project knowledge becomes tacit (Nerur et al., 2005) and an organization can become dependent on the developers for knowledge about a project or customer where the information exists only in the minds of the developers (Turk et al., 2005.) Due to short, fixed iterations, project scope can contain much variance, and it can become difficult to determine a schedule and finish date. These issues feed directly into budget complications, where the determination of an initial overall budget and cost can become difficult to predict without knowing what all you will be doing, when you will be doing it, and how long it will take. Agile's emphasis is also on building for current requirements and not necessarily for future ones (Ambler, 2005.)

Agile emphasizes face-to-face frequent communication. This may not be possible in all situations. It also emphasizes collocating a team in a project room or "bullpen." Again given the nature of technology, outsourcing, subcontracting, and telecommuting this might not be possible. Coupled with a lack of centralized documentation or handwritten notes in a project room that not all of the team has access to, and the team will have a difficult time acting as a cohesive whole.

The more parties there are that are involved in communication, the more complex communication becomes. Each person on a project needs to communicate with another in some way. This can be referred to as channels of communication and can be represented by the formula $(n^2 - n)/2$ where n is the number of participants. For example, with two participants (a customer contact and a project manager) there is 1 channel of communication. In an Agile project with a customer team of three, a developer team of four, and decentralized communication there are 21 channels. The addition of another customer and team member to account for a change, and 36 channels are present. The more channels there are, the more complicated communication becomes and the greater potential there is for poor or miscommunication.

There are technical challenges to Agile methods as well. Aside from a central documentation process and repository, typically developers have their own self-contained development environments

(Ambler, 2005.) As a result if environments are not deployed in a consistent manner across environments, there can be integration challenges with a release iteration.

In using Agile methods one is also making several assumptions regarding the nature of their organization or the customer. If these assumptions prove false, Agile methods will likely not be successful. These assumptions are outlined later in this paper.

Waterfall Development

What is Waterfall Development

Waterfall development is a term used to describe more traditional, plan-based project management methodologies. A example of this is the methodologies documented by the Project Management Institute (PMI) in the Project Management Body of Knowledge (PMBOK.) The successful completion of a quality project is managed and driven by a project plan developed in a systematic, documented manner. Core stages are Initiation, Planning, Execution, Control, and Closeout. Each stage has specific inputs and outputs, which feed into subsequent stages. Documentation and research is performed during Initiation and Planning, and the project becomes progressively elaborate as the team moves forward. The methodology assumes that the project can be defined, documented, and planned in a finite manner. The Execution phase should not begin until the project documentation, schedule, budget, and team is in place. Changes to the plan can be made but are required to go through a change-control process where change is reviewed for impact to the project's cost, schedule, and outcome before being made. During the early stages of the project the Project Manager's role is that of a manager and coordinator. As the project progresses into Execution the Project Manager's role shifts more towards communication and monitoring.

Predictive Methods

Waterfall development can be considered a Predictive method. Wikipedia.org defines Predictive methods as the following:

> Predictive methods…focus on planning the future in detail. A predictive team can report exactly what features and tasks are planned for the entire length of the development process. Predictive teams have difficulty changing direction. The plan is typically optimized for the original destination and changing direction can cause completed work to be thrown away and done over differently. Predictive teams will often institute a change control board to ensure that only the most valuable changes are considered.

Benefits and Risks of Waterfall

Waterfall and predictive methods have several strengths. They are fairly established, documented practices with large bodies of knowledge, case studies, and research behind them. There are many tools available to aid in project management. Management and customers can be given a finite schedule, budget, and cost estimate. Project work and cost progress can be monitored on a regular basis and reported on using earned value formulas in order to determine if efforts are on track and on budget. Project resources assigned to tasks need only to know how to accomplish their own work. Communication is usually centralized with the Project Manager and as such is consistent in nature. Documentation and formal processes aid in bringing on new team members or management to the project. Scope can be closely controlled, and project documentation and lessons-learned can be integrated and repurposed for future efforts.

Waterfall and predictive methods have problems as well. The amount of documentation and planning required in predictive methods can be excessive for smaller development projects, such as a small website, and can take a relatively significant amount of time. Predictive methods also assume that a project can be defined and planned in its entirety. For rapidly changing projects or customers with a need to change frequently or react to market demands, waterfall development can inhibit productivity and timely delivery given the process overhead and analysis needed when presented with change. If requirements were not documented effectively a developer might not have all the information they need, and a project manager can become a bottleneck to communication if developers do not have direct access to a customer or if the project manager is not technically proficient in a given issue.

Effective resource management, billing, and progress reporting in predictive methods is dependent on integrated enterprise tools in order to generate results. The output of these tools is dependent on quality, accurate input from project team members. If a project management tool such as MS Project is not integrated with a resource management tool or enabled in a shared manner, developers have to provide estimates to project managers who in turn have to update project documents, them update a resource management tool. Developers then need to take information on the same estimates, enter their time is a billing system, and hope that what the billing system invoices is close to what the project manager has communicated to the customer based on the MS Project file. The input of this information also causes additional project overhead as developers stop working on code in order to enter time, submit a progress report, or try to provide an estimate of percent complete for a given work package or task.

The production and release of functionality in waterfall development takes longer and occurs less frequently. Spending more time at the beginning of the project management process means that code is completed later in the process and in larger chunks. This increases the risk to the project due to less-timely feedback and introduction of results (Ambler, 2005.)

Identifying the Optimal Methodology

Both adaptive and predictive methods have the strengths and weaknesses. In order to identify the optimal methodology to use, there are several factors to consider.

Beware of Dogma

Beware of dogma when doing research. Some information available on Agile or Waterfall methods could be classified as dogma. Positions are established or statements made based on a belief that one methodology is better than another. These statements are not always based on fact or empirical data.

Unfortunately when opponents of Waterfall methods make statements regarding the benefits of Agile versus non-agile methods, the comparison seems to be made against **bad** project management. This is not to say that the stated benefits are untrue, but that the stated benefits might not be inherent to Agile methods only.

The following could be considered examples of methodology dogma:

"At any time you have access to complete accurate information as [to] the status of any feature…the team works in an energized space with constant communication about the project." (Lindstrom & Jeffries, 2004)

True, but can be part of any effectively run project.

"You can pick who will help you on any given task. You are not required to constantly work long hours." (Lindstrom & Jeffries, 2004)

"More than two-thirds of all corporate IT organizations will use some form of "agile" software development process within 18 months, Giga Information Group Inc. predicted this week at its application development conference here." (Lindstrom & Jeffries, 2004)

Statement one isn't necessarily true, and constant long hours *shouldn't* be inherent to waterfall methodologies given proper planning. The second Lindstrom & Jeffries is a quote from a Computerworld.com article from March 2002. However the following paragraph in the article was not included, and states:

"But so far, only a small percentage of corporations have adopted an agile approach, which aims to help companies attack projects that have unclear or rapidly changing requirements, said Liz Barnett, an analyst at Cambridge, Mass.-based Giga. She estimated that just 10% of corporate IT organizations now use the emerging lightweight methodologies that are creating a stir these days in programming circles. However, Barnett said she suspects roughly 25% are exploring some form of agile process." (Computerworld.com, 2002).

"First, every modern software development process…takes an evolutionary approach to development…[second] Agile techniques such as XP, FDD, DSDM, Scrum, and AMDD are growing in popularity." (Ambler, 2005.)

"You just don't see XPers wasting their time updating a Microsoft Project Gantt Chart." (Ambler, 2005.)

The first statement is not based on empirical data, any new approach could be considered evolutionary, and evolutionary or popular does not necessarily equate to optimal. The second statement assumes that time spent updating a Gantt chart is not productive, yet does not state why. In some cases information entered into a project management system is critical for earned value analysis or business evaluations.

"Agile isn't likely to replace the so-called waterfall development methodologies, those proven ivory towers that have been used for the development of everything from missile guidance to widget-tracking ERP systems." (Willoughby, 2005.)

Change is not bad. Evolution is not bad. Just because some a process has been used for a certain period of time does not mean it's necessarily the best tool for the job. Perhaps a methodology that was used to develop a missile guidance system for the Department of Defense might be a little more than what is needed for a 60 hour website project for a small startup company. And there's a difference between "used" and "used successfully."

"Developers like [Agile] processes because they limit or eliminate noncode-oriented documentation and promote simple test driven designs and collective code ownership." (Hudson, 2005.)

"…management will always decree the need for formal documentation and design because they need this information to effectively plan company strategy." (Hudson, 2005.)

"…your project must have fewer than 10 developers (agile processes aren't capable of scaling into larger or more sensitive projects." (Hudson, 2005.)

Statements one and two are extremely generalized and not necessarily true. Statement two also assumes that a company's executive team is going to use project documentation to plan corporate strategy. In statement three, 10 developers seems arbitrary and is not supported in any way. Additionally the author bases the inability on a lack of documentation for new team members and other internal projects. Agile addresses organic teams, and quite often other projects are not an issue.

“And if any group of developers were really that good all the time, why would you need any kind of process?” (Hudson, 2005.)

For the same reason that a group of experienced, master plumbers, electricians, carpenters, and masons can’t just get together and accidentally build a house. They may need less direction, observation, and can produce a superior product, but there still has to be some type of plan and coordination.

Develop Common Denominators

When evaluating different methodologies, it can be helpful to identify a commonality between methods. Focus on what it is you’re trying to accomplish. You’re not trying to use a process, you’re attempting to complete a project successfully.

Extend this logic. The definition of a project not up for debate, nor is the reasons projects might fail. Therefore, think about your project and what has to be accomplished. Think about risk and why projects can fail:

- Poor communication.
- Lack of sponsorship.
- Lack of proper funding or resources.
- Flawed premise or wrong product produced.
- Poor scope, schedule or cost management.
- Poor risk management.
- Poorly documented requirement or understanding of stakeholder needs.
- Inadequate testing or stakeholder feedback.
- Poor leadership or team morale.

Does the process you’re considering address these failure risks adequately? Is the process you’re currently using causing problems or affecting productivity, customer satisfaction, or quality?

Assumptions in Using Agile

An assumption can be defined as something taken for granted or accepted as true without proof (Dictionary.com.) In using Agile methods several assumptions are made regarding the nature of an organization or the customer. If these assumptions prove false for your organization, Agile methods will likely not be successful. The following assumptions are adapted from *Assumptions Underlying Agile Software-Development Processes* (Turk et al., 2005).

Assumption	**Details**	**Challenge**
Visibility	Project visibility can be achieved solely through the delivery of working code.	Applicable to situations where a tangible or visible deliverable is produced (such as a website.) For projects without a human interface component, visibility has to be accomplished by some other means.
Iteration	A project can always be structured into short fixed-time iterations.	This might not always be the case for a given effort or project.
Customer Interaction	Customer teams are available for frequent interaction when needed.	Customers might not have the level of availability required or flexibility in their schedule. They also might not want to have the level of involvement in the project that Agile methods need.
Team Communication	Developers are located in a time and place that allows them to have frequent face-to-face contact.	This may not hold true, especially in organizations that outsource, hire subcontractors, or have staff that normally work remotely of are geographically dispersed.
Face-to-Face	Face-to-face communication is the best method for communicating with staff and customers.	While face-to-face communication usually allows for the most amount of information to be conveyed (body language, eye contact, etc.) it may not always be practical, cost effective, or appropriate. For example, the notification of a risky or complicated addition to a project might be best communicated in writing via email.
Documentation	Developing extensive documentation is counterproductive.	Assumes that any documentation is counterproductive. While it's true that extensive planning up front delays the production of code, a minimal level of documentation can help provide a snapshot of framework to help steer ongoing work. Additionally, there is no reason that documentation cannot be made and the end of a project. This ensures that the documentation is based on the delivered code, and that the code itself does not need to be the source of documentation.
Changing Requirements	Requirements will always change due to technology, customer needs, or business.	No issue.
Cost of Change	The cost of change does not dramatically increase over time.	This assumes that changes or errors that are detected earlier have a lower cost to remedy than those detected later in the project. This assumes however that the sum cost of each issue detected during each iteration is less than a issue detected later in the project on a larger scale. It also

		assumes that change management becomes a minor issue and that changes can just be made as needed.
Team Experience	Developers have the experience required to define and adapt processes as needed.	Not always true. It may be difficult for organizations to find or have enough of the type of developers needed to be successful with Agile methods.
Self Evaluation	Teams are able and willing to evaluate themselves.	Again, not always true. And a team may be willing to evaluate themselves, but the organization in which they work must also be accepting of this process, and not penalize staff when issues are identified.
Self Organization	The best solutions, designs, and requirements come from self-organizing teams.	Assumes that staff on teams are capable of self-organizing in the first place, that this practice is acceptable and does not cause problems in an organization. Also assumes best solutions will emerge. There may be research in the field of behavioral psychology or motivation theory to support this, but it needs to be cited if so.
Quality Assurance	Evaluation of software products and processes can be restricted to frequent informal reviews, interviews, and testing.	The more informal testing procedures used may not be sufficient for all situations. For example, for safety-critical or financial-critical systems more rigorous, formal testing may be required in order to ensure the level of quality and accuracy needed.
Continuous Redesign	Systems can be continuously redesigned while still retaining their structural and conceptual integrity.	Assumes that this statement is true. However, continuous redesign could in fact be causing bigger issues from a systems-approach perspective, but without a framework for earned value analysis or progress reporting this may be hard to identify.
Application-specific Development	Reuse and generality should not be the goals of application-specific software development.	Assumes that this practice is beneficial to the development of the project, prevents wasting resources, keeps a solution simple, and that code or elements are not usually reused anyway.

Common Sense

It can be said that good project management is in large part common sense and good communication. Without this a project will be hard-pressed to be successful. If you're spending more time performing documentation and analysis than you are producing your deliverables, if customer satisfaction is low because there isn't enough (or too much) involvement, or if the organization is not performing well because there is a lack of documentation, then you may need to change your processes. Choose the right process for the right job (Ambler, 2005.)

Risk Management and Agile

There are risks associated with using Agile methods in and of itself. While Agile does address some issues well, it can create other risks for a project based on the nature of an organization or project itself. Several risks associated with Agile methodologies have already been identified in this paper. Risk Management in Agile is outlined further below.

Risk Management Overview

As part of the project management process, the Project Management Institute (PMI) includes a process of risk management in its recommended body of knowledge (PMBOK, 2004). This can be considered the current common risk management methodology for waterfall development. The PMBOK Risk Management process is defined as a process concerned with planning, identification, analysis, response, and monitoring and control of risks during a project. This process can be updated throughout the project.

The objective of risk management is to decrease the likelihood and effect of events that would negatively impact the project, as well as increase the likelihood and impact of positive events. The process includes planning, identification, qualitative (subjective) analysis, quantitative (numerical) analysis, response planning, and monitoring and control.

Helping Projects Succeed

Proper risk management can help a project to succeed. Based on the overview of risk management as well as some of the reasons identified as to why projects fail, the benefits of effective risk management can be illustrated. While each potential reason for failure may have its own unique mitigation strategies, the process is the same:

Reason for Failure	Addressed By
Poor communication	1. Identifying the issue. 2. Understanding the likelihood and affect of the risk on the project. 3. Determining what can be done to mitigate the issue. 4. Being able to communicate what the issue is, how it can be addressed, and what will be done. 5. Implementing the mitigation strategy. 6. Following up on the issue and monitoring. 7. Ongoing communication
Lack of proper funding or resources.	
Flawed premise or wrong product produced.	
Poor scope, schedule or cost management.	
Poorly documented requirement or understanding of stakeholder needs.	
Inadequate testing or stakeholder feedback.	
Poor leadership or team morale.	

Waterfall Risk Management within Agile

The process outlined above becomes challenging within Agile methodologies. The process itself is still valid, and on a per item basis the process is still effective. However, the PMBOK indicates specific inputs and tools for risk management planning. Inputs such as the scope statement, WBS, schedule, budget, and resource planning data feed into risk planning. In waterfall development, as the Initiation and Planning stages progress, the project becomes more elaborated, more documented, and more defined. Data becomes more refined. The end result is an overall picture of the project, based again on predictive development methodologies and the assumption that the future can be planned in detail. With Agile and adaptive development methods this level of documentation does not exist. The scale and amount of inputs are not present, and defined schedules and budgets don't exist in the same manner.

Risks Agile Addresses

Agile does not mean unplanned or unprepared. By definition Agile is flexible. It allows for the team to respond quickly to changes or new risks. Adaptive methods assume that you cannot plan the future in detail, therefore there are limits to how much planning and documentation can be done. To go beyond those limits would be a waste of time and resources. Proponents of Agile cite that it's an ideal methodology for dealing with risk in software development:

> "Since the Agile approach places a higher value on risk management, early user feedback is highly valued. Agile projects should either succeed or fail early on, before too many resources have been committed." (Cohen, 2005)

The accuracy of the above statement is up for debate. There's no research to support that Agile puts a "higher" value on risk management. And a project failing earlier then it would've in a waterfall project isn't necessarily a selling point. It's possible that with effective risk management in a waterfall project, correct risk management planning could have identified and anticipated risks that result in project failure, thus leading to a recommendation not to do the project in the first place.

Agile does however employ certain practices and methods that enhance its ability to identify, analyze, and respond to risk. For example, in reviewing the table identifying the reasons some projects fail, and taking into account risk management planning, inputs, and tools, it can be shown that Agile methods can enhance the process:

Reason for Failure	**Advantage Inherent with Agile**
Poor communication	Agile relies on a high level of communication between tight-knit teams and close frequent contact with customers. Increased communication does not necessarily equate to effective communication, however if there are communication issues they will be recognized very early in the project, and given the nature and skill-set of most Agile development teams it’s likely the team would have effective communication skills.
Lack of proper funding or resources.	Agile only requires funding for the iteration being worked on. If the project can be structured in such a way that iterations can produce usable code, and the system as a whole is functional, then code can be completed as funding or resources are available.
Flawed premise or wrong product produced. Poor scope, schedule or cost management. Poorly documented requirement or understanding of stakeholder needs.	There is no big planning effort in the beginning of an Agile project. The customer is constantly involved, and what is being produced is defined as the project progresses. Given that there is no rigid framework, and “lost work” is minimized with short-duration iterations, it’s easy for the team to change focus or direction as needed.
Inadequate testing or stakeholder feedback.	Part of iterative delivery is quality assurance and testing. As each iteration is released, it’s tested as it’s developed and tested again by the customer from an acceptance standpoint. Test-driven delivery establishes what a piece of code should do and actually goes through that test process to prove it works. As code is released, the system as a whole is also tested. This results in testing being done throughout the life of the project and by the customer. Problems are addressed immediately rather than at the end of the project where they may be more difficult to resolve for various reasons.
Poor leadership or team morale.	Agile teams are usually considered different from waterfall teams. They are self-organized/structured, share roles and responsibilities, and are usually made up of more experienced developers. This forces the group to work as a tight matrix and share in the workload and in project leadership. Increased communication and face-to-face discussions can enhance working relationships and identify any leadership or morale issues early in the project.

The above advantages are not inherent to Agile only, but they do lend themselves easily. In addition to addressing risks that might cause failure, effective risk management also includes making the most of opportunities in order to help your projects succeed. The website gantthead.com identified what it called *Rules of Project Success,* the majority of which are listed below and can be checked against Agile to see how they may fit:

Rules of Project Success	**Advantage Inherent with Agile**
Gain consensus on project outcomes.	High level of communication and customer interaction with regards to prioritization of deliverables and the testing/release of iterations. Item #1 from the Agile Manifesto, “Our highest priority is to satisfy the customer through early and continuous delivery.”
Build the best project team you can.	Usually a given with Agile methods and needed in order to effective work with Agile methods.
Develop a comprehensive, viable plan and keep it up-to-date.	The plan is always up to date as it is always changing and evolving.
Determine how much stuff you really need to get things done.	This is a tenet of Agile. Item #10 from the Agile Manifesto, “Simplicity--the art of maximizing the amount of work not done--is essential.”
Have a realistic schedule.	This is a tenet of Agile. Item #3 from the Agile Manifesto, “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.”
Don’t try to do more than what can be done.	This is a tenet of Agile. Item #10 from the Agile Manifesto, “Simplicity--the art of maximizing the amount of work not done--is essential.” Item #8 from the Agile Manifesto, “Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.”
Remember that people matter.	Core to Agile. “Individuals and interactions over processes and tools” and “Customer collaboration over contract negotiation.”
Gain the formal and ongoing support of both management and stakeholders.	This is a tenet of Agile. Item #12 from the Agile Manifesto, “At regular intervals, the team reflects on how to become more effective then tunes and adjusts its behavior accordingly.”
Be willing to change.	Core to Agile. “Responding to change over following a plan.” Item #2 from the Agile Manifesto, “Welcome changing requirements, even late in development. Agile processes

	harness change for the customer's competitive advantage.”
Keep others informed of what you are doing.	High level of communication and customer interaction with regards to prioritization of deliverables and the testing/release of iterations. Item #1 from the Agile Manifesto, “Our highest priority is to satisfy the customer through early and continuous delivery.”
Be willing to try new things.	Core to Agile. Item #2 from the Agile Manifesto, “Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.”
Become a leader.	Part of an Agile team. Everyone is expected to be a leader and share in the responsibility.
The success of a project always depends more on people than on process.	Core to Agile. “Individuals and interactions over processes and tools” and “Customer collaboration over contract negotiation.”

Risks Agile Creates

There are tradeoffs however to using Agile. In some cases the methodology can create risks while addressing others. Several risks associated with Agile methodologies have already been identified in this paper.

Less Documentation and Tools

There is a minimal amount of project documentation with Agile. Combined with a lack of formal or commercially available tools for Agile methods, there becomes a greater risk of confusion and ability to manage what documentation does exist.

Susceptibility to Scope Creep

While there isn’t a scope statement per se, there is a defined understood general scope to an Agile project and wheat’s being produced. As interactions progress, if customers request to add or change functionality they can do so, and the team should be able to manage it. However, each change to the scope increases the project duration, cost, and delays the date when project resources may become free. Because the same types of reporting and documentation used in waterfall are not present it may be more difficult to understand and report on the effect to the project as a whole.

Knowing When To Be Done

Similar to the risk of scope creep is being able to know when a project is complete or agreeing as to when a project is complete. Each iteration should deliver working code. Customers are not discouraged from making changes, adding functionality, or reacting to market needs.

Team Colocation and Proximity

The nature of Agile teams and how they interact necessitates having a project room, colocating teams together, and frequent face-to-face contact with the team and customers. This may not always be possible, or desirable from a customer perspective. It limits options and makes solutions such as outsourcing difficult.

Skills Required

Agile teams are expected to produce high quality work, be able to work directly with customers, take on leadership roles, and be generalists in many areas. As a result, more seasoned experienced developers are normally required in order to operate at optimum levels. These types of people may be difficult to find, unavailable in sufficient quantities, and be expensive to hire and retain.

Budgeting and Scheduling

There is a minimal amount of project documentation with Agile. Scope, deliverables, and iterations are also likely to change. This means less inputs for the planning process, resulting in difficulty in determining a budget and a schedule.

Agile Assumptions

In using Agile methods several assumptions are made regarding the nature of an organization or the customer (outlined previously in this document.) If these assumptions prove false for an organization, Agile methods will likely not be successful and cause problems for the project and team.

Something New

For many organizations and customers Agile methods may be something they have little to no experience in. This can pose some significant risks in general. This is further explored in the Organizational Challenges section of this document.

Addressing Agile Risks

The risks that Agile creates can be mitigated in some respect. As stated previously, good project management is in large part common sense and good communication. Some modifications can be made when using Agile methods if needed in order to address risk.

The amount of documentation produced can be increased enough to provide information in order to reduce confusion. If commercial tools are not available, simple custom tools can be developed to help facilitate Agile methods, maintain documentation, and assist with communication. A slight increase with documentation and planning might also provide better inputs into other parts of the process.

Risks related to scope creep can be mitigated not by rejecting change but by taking a quick look at what the effect is on the project overall, and again communicating to the customer that the net result will be a change in budget and delay in completion. The best method for this would likely be verbally during a meeting followed with an email confirmation. If ongoing changes or requests for more work occur, this may be an acceptable risk. As long as the development work is producing revenue and the customer is paying, this may be a positive. There may be upcoming work to consider however where there may be a greater loss if resources are not able to commit to a new project where the revenue would exceed the remainder of potential ongoing work of a project that should be completed.

Inabilities to meet the team colocation requirements may also be an acceptable risk. While having a team in the same place together all the time may be ideal, it may not be realistic. It's possible that a team may still be able to be effective if they are able to colocate only some of the time, possibly taking advantage of technology such as video conferencing as needed. The skills needed to fully staff an Agile team also pose a challenge. This risk could be mitigated by having more senior developers mentor other junior developers (addressed later in this paper.)

Finally, as mentioned in the section *Identifying the Optimal Methodology,* use common sense. Test the Agile assumptions to make sure that Agile is a good fit for your organization. Review the identified risks above along with the recommended mitigation strategies and decide if they're a viable solution for your project. And be open to the possibility that Agile methods may not be a good fit for your project management and risk management needs.

Communicating Agile

If after evaluating various methodologies Agile does seem to be a good fit for your organization, there are still some challenges and issues to consider. There's nothing directly actionable however in methodologies. For an organization to adopt Agile methods, more than principles are needed – you need practices. (Scheer, 2005.)

Organizational Challenges

For an organization that is rooted in more predictive methods, changing to Agile can be a culture shock. This can be more challenging if only parts of the organization are using Agile methods while others are used Waterfall methods (especially so with joint projects.) Because Agile is developer-centric (Scheer, 2005) relating the benefits to non-development staff, customers, and management can be difficult. It is sometimes easier to relate the benefits of Agile by stating what wouldn't be happening as compared to another methodology, or what the savings to cost and time are.

Agile methodologies require a shift from command-and-control management to leadership-and-collaboration (Nerur et al., 2005). In a strong matrix or project-based organization the Project Manager role changes, and there can be a challenge in getting them to relinquish the amount of authority and control they previously had (Nerur et al., 2005.) An effective guiding vision for a project needs to be provided, and free and open access for team members to needed information has to be a given.

In order for an Agile team to be effective, management also needs to take a different approach. In *Steering from the Edges* (Augustine et al., 2005) several requirements to management style are identified, including:

- Light touch management style: less micromanagement, decreased control.
- Adaptive leadership: teams balance on the edge of chaos where too much structure is too rigid while not enough can push the team into chaos.
- Ability to adapt to change.
- View of the organization as fluid and composed of intelligent people.
- Recognition of the limitations of establishing order with control.
- Overall humanistic approach to problem solving.

Management needs to meet these challenges in order to be successful. If not properly managed, an elitist or egalitarian atmosphere can sometimes arise, especially when not all departments of an organization use the same methodologies. This can cause additional tension and management-related issues.

There are also technical issues to address. Again, typically developers have their own self-contained development environments (Ambler, 2005.) As a result if environments are not deployed in a consistent manner across environments, there can be integration challenges with a release iteration.

Organizations that value learning need to be able to encourage the documentation or externalization of tacit knowledge (Turk et al., 2005) and this knowledge needs a place to live and a means in which to be accesses by staff. Legal issues may dictate also dictate a greater level of documentation, such as Sarbanes Oxley or contract management.

Maximizing the Benefits and Find the Right Fit

Agile approaches are not silver bullets (Turk et al., 2005) and long-standing methodologies are allowed to evolve and change. Again, use common sense. It's possible to adopt a hybrid of practices where applicable and make the most of various methodologies.

Certain projects may be a better fit for Agile methods, such as Internet applications development, where there is a lot of change, time-to-market pressures, and upgrade costs to future releases are minimal (Turk et al., 2005.)

For example, if a team has to identify what work will be done for iterations, there's no reason a developer can't estimate these tasks. This can be used to provide a baseline schedule and budget to a customer or management for a statement of work, provided everyone understands the nature of Agile. If developers are opposed to this process, it could be because they don't want to (dogma) or because they fear their estimate will not be accurate but will be held to it anyway, a result of poor project management and leadership.

Brief, simple, templated initiation and planning documentation can be done, which can then be used as the basis for iterative planning or deliverables prioritization. Organic Agile teams can be created, a light management style used, and perhaps a lead engineer identified in order to facilitate meetings and customer communications. Once a project is complete, team members can produce documentation and store it in a centralized repository for later use and learning.

Introducing Agile

Evolve Your Methods

Introducing Agile methods to an organization or customer that has never had any experience with it can be difficult. It is sometimes easier to relate the benefits of Agile by stating what wouldn't be happening as compared to another methodology, or what the savings to cost and time are. To communicate the benefits of Agile, look at how a faster time to delivery can be accomplished (Braun, 2005):

- Extra features are not added, and tasks are deliverables are prioritized by the customer.
- Decisions can be made faster because in theory everyone needed is already present.
- Iterations focus on delivering a narrow set of features with best business alignment.

It also may be beneficial to attempt to introduce various aspects of the methodology piecemeal. For example, it may be possible to modify the work packages of a waterfall-based project into iterative release. As the customer or company becomes more comfortable with the iterative nature of releases, progressively perform less planning up front and move iterations up. And as the comfort level increases, then the nature of project teams may change and customer involvement may increase as well.

If a development team is working within an Agile "pocket" within an organization, appointing someone from the team or department that's knowledgeable in other methodologies to act as a Project Manager may help facilitate project progress. This person can act as an Agile "shim" or "translator" to groups outside of an Agile team, and can help facilitate communication, understanding, process, and progress for Agile projects.

Provide Objective Education

If you attempt to move to Agile methods, don't assume that others in your department, organization, or customers understand what it is you're doing and why you're doing it. Provide both simple and detailed information that's easily accessible. Know your audience. Develop the content for different learning styles, and offer workshops, sessions, and "gurus" to help others learn and better understand. Avoid dogma, and produce objective information that states why you feel Agile methods will help you to be effective, successful, and how risks can be addresses.

Be Agile

Agile methods embrace change, flexibility, and place people over process. As such, maintain focus on what your customer needs are. If Agile methods are counter to what your customer wants and what fits with your organization, to use it regardless wouldn't be in the spirit of Agile methods and the Agile Manifesto.

Conclusion

This paper covered the following:

- Issues relating to a lack of empirical data for Agile methods.
- A definition of Agile and adaptive methods.
- Examples and definitions of various practices considered to be agile.
- A definition of Waterfall and predictive methods.
- Benefits and risk of both Agile and Waterfall methodologies.
- Issues to be cognizant of when identifying an optimal methodology.
- Assumptions about Agile methods to be aware of and evaluate.
- Risk management basics.
- Risks of project failure.
- Risks and opportunities Agile addresses.
- Risks Agile creates and potential ways to address them.
- Challenges that may be faced when communicating what Agile methods are and in using them.
- Tips for maximizing benefits of various methodologies.
- Tips for introducing Agile methodologies into your organization and to your customers.

References

1. Agile project management; steering from the edges
Sanjiv Augustine, Bob Payne, Fred Sencindiver, Susan Woodcock. **Association for Computing Machinery. Communications of the ACM.** New York: Dec 2005. Vol. 48, Iss. 12; p. 85

2. Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research
John Erickson, Kalle Lyytinen, Keng Siau. **Journal of Database Management.** Hershey: Oct-Dec 2005. Vol. 16, Iss. 4; p. 88 (13 pages)

3. Assumptions Underlying Agile Software-Development Processes
Daniel Turk, Robert France, Bernhard Rumpe. **Journal of Database Management.** Hershey: Oct-Dec 2005. Vol. 16, Iss. 4; p. 62 (26 pages)

4. On the Adaptation of an Agile Information Systems Development Method
Mehmet N Aydin, Frank Harmsen, Kees van Slooten, Robert A Stegwee. **Journal of Database Management.** Hershey: Oct-Dec 2005. Vol. 16, Iss. 4; p. 24 (17 pages)

5. Agile Programmers Turning to New Tools
Heather Havenstein. **Computerworld.** Framingham: Sep 5, 2005. Vol. 39, Iss. 36; p. 47 (1 page)

6. Lean/Agile Methods for Web Site Development
Ellen Braun. **Online.** Medford: Sep/Oct 2005. Vol. 29, Iss. 5; p. 58 (3 pages)

7. Agile Programming Not For The Risk Averse
Beth Cohen. **InformationWeek.** Manhasset: Aug 15-Aug 22, 2005. p. 74 (1 page)

8. Making the Agile Move
David Rubinstein. **Software Development Times.** Oyster Bay: Aug 1, 2005. p. 38 (1 page)

9. Coder Be Agile, Coder Be Quick
Mark Willoughby. **Computerworld.** Framingham: Jul 25, 2005. Vol. 39, Iss. 30; p. 36 (1 page)

10. Challenges of migrating to agile methodologies
Sridhar Nerur, RadhaKanta Mahapatra, George Mangalaraj. **Association for Computing Machinery. Communications of the ACM.** New York: May 2005. Vol. 48, Iss. 5; p. 72

11. The Case for Agile Database Development
Scott W Ambler. **Software Development Times.** Oyster Bay: May 1, 2005. p. 34 (2 pages)

12. Preparation is the key to successful projects
Julia Vowler. **Computer Weekly.** Sutton: Nov 23, 2004. p. 34 (1 page)

13. Extreme Programming and Agile Software Development Methodologies
Lowell Lindstrom, Ron Jeffries. **Information Systems Management.** Boston: Summer 2004. Vol. 21, Iss. 3; p. 41 (20 pages)

14. Egg saves time with agile development model
Nick Huber. **Computer Weekly.** Sutton: Jun 15, 2004. p. 4 (1 page)

15. Welcome to the 'Zen and the Art of Agile Computing' workshop
Scott Ambler. **Computing Canada.** Willowdale: Apr 9, 2004. Vol. 30, Iss. 5; p. 21 (1 page)

16. Agile development best dealt with in small groups
Scott Ambler. **Computing Canada.** Willowdale: Apr 26, 2002. Vol. 28, Iss. 9; p. 9 (1 page)

17. Users warm up to agile programming
Carol Sliwa. **Computerworld.** Framingham: Mar 18, 2002. Vol. 36, Iss. 12; p. 8 (1 page)

18. **Extremely agile programming**
Alan Radding. **Computerworld.** Framingham: Feb 4, 2002. Vol. 36, Iss. 6; p. 42 (2 pages)

19. **Agile acquisition**
Bill Weaver. **Scientific Computing & Instrumentation.** Feb 2003. Vol. 20, Iss. 3; p. 13 (2 pages)

20. **Popularity contest**
Michael J Hudson. **Intelligent Enterprise.** San Mateo: Jun 28, 2002. Vol. 5, Iss. 11; p. 52 (2 pages)

Appendix A – The Agile Manifesto

From the Agile Manifesto (http://www.agilemanifesto.org/principles.html)

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.