

Managing Chaos in an Agile World



Samir Ray & Dipesh Patel

Introduction

In 1994, the Standish Group published what is now a well known study on project management, The CHAOS Report. The study presented an analysis of successful, challenged, and impaired projects and stated that “software development projects are in chaos”¹. The results of this study have fueled much of the emphasis on project management and software development methodologies in the last 14 years as IT professionals sought to improve the success rate of their projects by better managing the chaos. While the results from the latest studies are significantly better, they still leave a great deal of room for additional improvement.

In recent years, Agile development methodologies have also been increasing in popularity. Empirical studies demonstrate their overall success in delivering better quality software and qualitative case studies tout improved results and overall customer satisfaction. While the results are highly desirable, the simple adoption of an Agile methodology is not sufficient to guarantee a higher rate of successful projects. Agile projects face many new challenges that continue to demand the need for expert project management.

Lessons Learned from the CHAOS Reports

The CHAOS reports include a survey of IT executives attempting to identify the most common factors contributing to project failures (seen in Figure 1). Failed projects include those that are over budget, in terms of dollars and time, and not fully delivering expected features (challenged) or cancelled (impaired).

Factor	Challenged Projects	Impaired Projects
Incomplete Requirements & Specifications	12.3%	13.1%
Lack of User Input	12.8%	12.4%
Changing Requirements & Specifications	11.8%	8.7%
Lack of Resources	6.4%	10.6%
Lack of Executive Support	7.5%	9.3%
Unrealistic Expectations	5.9%	9.9%
Technology Incompetence / Illiteracy	7.0%	4.3%
Lack of Planning	0.0%	8.1%
Didn't Need It Any Longer	0.0%	7.5%
Lack of IT Management	0.0%	6.2%
Unclear Objectives	5.3%	0.0%
Unrealistic Time Frames	4.3%	0.0%
New Technology	3.7%	0.0%
Other	23.0%	9.9%

Figure 1¹

The immediate lesson for the software development community was better project management. To date, the focus for fixing projects has been on more formal processes and controls – locking down requirements early and minimizing changes in scope. Good project managers have been trained to fully document all requirements, obtain user sign-off, facilitate monthly steering committee meetings, communicate progress tracked against a plan, and execute a well-defined change control process.

¹ 1994 Standish Group CHAOS Report

An updated study from the Standish Group showed that from 1994 to 2006 we saw a jump in successful projects from 16% to 35%. Much of this improvement can be attributed to a greater focus on project management. However, 35% is still not a level that we should consider adequate or acceptable. Additionally, the answer project management has provided in the past has been to formalize user and executive involvement (requirements sign-off, change control committees, etc.) as well as push back on features to reduce the impact of the top factors in project failures: incomplete requirements, changing requirements, lack of user involvement, and lack of executive support. While budget, timeline, and features are all individual contributing factors in measuring the success or failure of a project, they are not a holistic measurement of customer satisfaction. Further, it is just not possible for customers to fully outline all of the desired features of a complex system upfront. The approach of pushing back on scope and minimizing features to deliver “successful” projects does not necessarily meet customer needs.

The Agile Answer

By using an Agile Software Development methodology, our highest priority is to satisfy the customer. Per the Agile Manifesto², this includes putting procedures in place that support constantly evolving requirements. We focus on delivering working software and involving the business customer on a daily basis to help them better envision the end solution rather than focusing on creating lengthy requirements documents that may or may not be read in detail. This reduces the “bait and switch” effect many customers experience after they receive software based on requirements documents. Agile approaches emphasize communication and detailed monitoring and controlling mechanisms over rigorous project process controls to help manage the chaos in software development projects.



Figure 2²

To date, several methodologies have been developed that provide guidance on applying these Agile principles in project execution. While each methodology offers some unique traits, common themes are:

- Work is done in smaller iterations (usually ranging from 1 - 4 weeks)
- Requirements are documented, estimated, and prioritized in smaller, discrete units that allow flexibility in planning
- New requirements are estimated, prioritized, and incorporated throughout construction
- Estimates often measure the relative complexity of the requirement in terms of value units (i.e. story points, # of features) that are not necessarily a discrete amount of time
- Estimates for each requirement are used to make prioritization decisions and plan iterations, they do not represent a detailed task list and timeline
- Velocity measures the number of units that can be completed by the team within an iteration
- Project teams are comprised of business and technology resources
- Development team members and customers communicate informally on a daily basis to clarify requirements and resolve issues
- Emphasis on review, continuous integration, and testing to ensure high software quality

Good project management also plays a valuable part in the execution of Agile projects. Agile methodologies are most definitely not a replacement for project management methodologies such as

² Manifesto for Agile Software Development (<http://www.agilemanifesto.org/>)

PMBOK³'s Project Management Process Groups. Those groups include Initiating, Planning, Executing, Monitoring and Controlling, and Closing and occur not only at the overall project level, but also in each iteration during the overall project's Execution process.

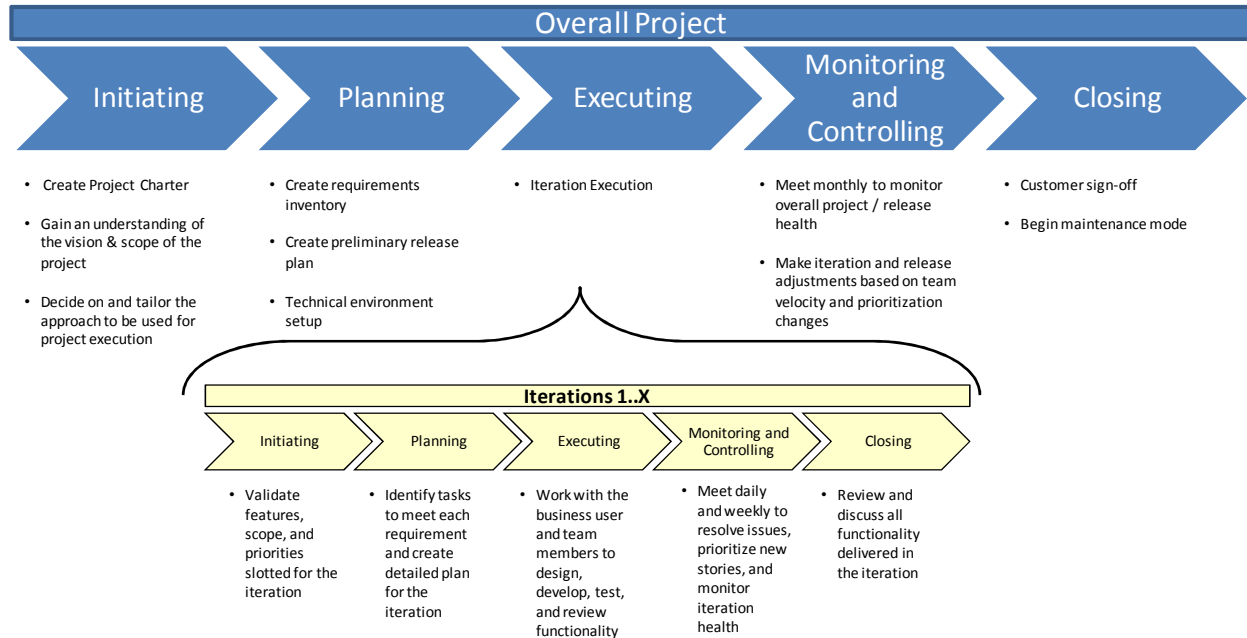


Figure 3

As depicted above, at the overall project level the PMBOK Process Management Groups help to guide the definition and execution of overall project scope and a high-level release plan that includes a number of iterations. During the execution of each iteration, the Process Management Groups help direct the detailed planning of the iteration, implementation and testing of features, monitoring of iteration health on a daily and weekly basis, and final review of the iteration features.

Agile methodologies combine best practices in software development to address four of the top reasons for project failures in a very different way – through much greater user and executive involvement and a methodology that welcomes changing requirements. The impact of Agile processes is recognized by its appearance as one of the top 10 contributing factors for successful projects in the 2006 CHAOS report. Additionally, many of the practices of Agile methodologies lead to the implementation of other success factors also included in the 2006 top 10 list.

However, implementing Agile methodologies does not come without its own set of challenges.

- Top 10 Factors in Successful Projects**
- User involvement
- Executive management support
- Clear business objectives
- Optimizing scope
- Agile process
- Project manager expertise
- Financial management
- Skilled resources
- Formal methodology
- Standard tools & infrastructure

Figure 4

³ A Guide to The Project Management Body of Knowledge (PMBOK® Guide)

Project Chaos in the Agile World

Agile methodologies are touted for their effectiveness in delivering working software using an incremental approach. Customers see the value of what they are paying for and have a much greater ability to guide and shape the final product. However, the shorter cycles, rapidly changing environment, and need for greater user involvement increase the levels of chaos in the project.

Shorter cycles means more projects

In Figure 3, we discussed that all the project management process groups – Initiating, Planning, Executing, Monitoring & Controlling, and Closing – are being executed on an iteration-by-iteration basis. On a project with three iterations this means you are effectively executing three projects in a very compressed timeframe and coordinating an overall project release effort.

Changing and reprioritizing requirements

Before developing any new feature, developers have a conversation with customers to ensure clarity about the requirement. Through this conversation, the development team is actively opening the door to requirements changes and additional requirements. Delivering and reviewing working software with customers at the end of each iteration gives customers yet another opportunity to change their minds.

Shift in planning approach

Agile methods require a fundamental shift in the way projects are estimated, planned, and managed. The development team is asked to estimate effort without going through the steps of a detailed technical design. Project managers must plan iterations with the knowledge that requirements will not necessarily remain fixed. Customers must pay much more attention to the granular details of the project to make prioritization decisions about when requirements should be worked, based on effort estimates that are often subject to change.

Increased time commitment from customers and decision-makers

The process does not just open the door to allow greater customer involvement, it demands it. Agile projects cannot succeed without sustained user and executive involvement. Since there is no upfront sign-off on a very detailed requirements document, an Agile development team cannot go develop in a black box. Further, the process allows for requirements to change on a daily basis within an iteration that is no longer than 4 weeks. The project pace simply will not allow a once a month check-in from executives and decision makers.

Frequent builds and code integration

In Agile methods, code is constantly built and integrated. While this helps to identify issues early that might only arise in the system testing phase of a waterfall project, it also places an extra burden on developers to fix build issues as development is ongoing. Depending on the environment, there may also be an increased involvement from infrastructure groups to setup and support build processes.

Lack of comprehensive documentation

Documentation is not the focus of Agile development. The methodologies do call for code to be commented adequately and written in a structured way to provide for maintainability. However, they do not call for formally documenting functional requirement decisions or technical design that can be helpful over the maintenance lifecycle of a software product.

In summary, Agile projects come with their own share of complexities. Good Agile project managers must understand what some of these challenges are and what can be done to mitigate them.

Mitigating Agile Project Challenges

The nature of an Agile project challenges the project manager to juggle priorities to manage the ever-evolving needs of the customer. Additionally, the development team focuses more heavily on testing, learning, and sharing knowledge throughout the development process with the purpose of producing higher quality code in the long run. The table below summarizes some of the specific methods and tools that can be used to mitigate many of the unique challenges of Agile projects.

Challenges Mitigated	Methods & Tools
<ul style="list-style-type: none"> • Shorter project cycles • Changing and reprioritizing of requirements • Shift in planning approach • Increased time commitment from customers and decision-makers 	<ul style="list-style-type: none"> • Iteration Plans • Burndown Charts • Task Management Practices • Team Velocity • Daily Stand-ups • Weekly Customer Team Meetings
<ul style="list-style-type: none"> • Frequent builds and code integration 	<ul style="list-style-type: none"> • Automated Continuous Integration • Code Reviews • Test-Driven Development
<ul style="list-style-type: none"> • Lack of comprehensive documentation 	<ul style="list-style-type: none"> • Collaboration Tools • Knowledge Repositories

Figure 5

Iteration Plans

An iteration plan begins with a detailed, prioritized list of the requirements to be addressed in that iteration and an effort estimate. As part of the iteration planning process, these will be turned into a project plan that calls out the detailed tasks necessary to deliver on each requirement.

The iteration plan is crucial to helping customers understand what features are to be delivered and aids in making prioritization and slotting decisions. During the execution of an iteration, while the preference is to slot new requirements into future iterations, the iteration plan helps determine what gets pushed if the functionality must be included in the current iteration.

Burndown Chart

A burndown chart shows the amount of work remaining to complete an iteration and the entire project. The chart may take the form of a line graph or more granular table. The burndown chart shown below, details the number of units of work performed, added, and remaining for each iteration of the project.

	Iteration 1	Iteration 2	Iteration 3
Units of work at start of iteration	150	120	55
Units complete during iteration	50	55	55
Changed estimates	5	-10	
New units added	15	0	
Units remaining at end of iteration	120	55	0

Figure 6

A burndown chart across all iterations is particularly useful in both communicating changes in scope and showing the amount of work remaining to complete the entire project.

Task Management Practices

The changing requirements and plan necessitate a more dynamic mechanism for communicating task prioritization to developers and tracking status than a traditional project plan. Agile projects often use large whiteboards and/or requirements software (i.e. Rally, SharePoint) to keep track of requirements and priorities for an iteration.

In a self-organizing team, the units of work are ideally not assigned, but instead signed up for by developers from the overall team list. As tasks are completed, developers can update the shared whiteboard or community tracking system with the remaining effort. This level of transparency facilitates daily reporting as well as raises individual accountability. Overall status and individual performance is always clearly transparent to the team and customers walking by the whiteboard or logging into the system.

Team Velocity

The concept of velocity is a premise of Agile projects that helps in planning iterations and projecting completion dates. Initial planned velocity is usually an estimated number based on ideal days of work with a buffer factor for interruptions. Actual velocity is defined as the number of units of work completed in an iteration. After the execution of the first couple of iterations, the team should be ramped up and good data points are available to calculate a planned team velocity for future iterations. Understanding and being able to communicate the team's velocity also helps with prioritization and resource management decisions when planning timelines and iteration scope.

Daily Standups and Weekly Customer Team Meetings

Communications, including meetings, in Agile projects should be frequent, short, and pointed. There are definitely high expectations for the level of communication and involvement from all members of the team – developers and customers alike. While these time commitments cannot be avoided, informal communication throughout the week; frequent, smaller meetings; as well as staying focused on results can help to alleviate the burden.

The members of the development team meet on a daily basis in a quick daily standup with no formal agenda. While the name of the meeting varies across methodologies, the purpose remains the same – provide each developer a forum to raise issues, discuss roadblocks, and communicate the areas of focus for the day. In addition, the customer team convenes on a weekly basis to review priorities and progress. Well-organized iteration plans and burndown charts are used to move discussion points and make decisions quickly.

Automated Continuous Integration, Code Reviews, & Test-Driven Development

A cornerstone of Agile development is a continuous focus on quality control. The objective is to catch and correct defects early in the development process. As previously highlighted, Agile projects call for code to be compiled and deployed multiple times daily to catch issues. Managers are expected to not only serve as leaders for the team, but to perform frequent testing on iterations before they are released to the customer. Many Agile development teams also incorporate automated unit tests into the build process to further ensure code quality. A number of tools such as Cruise Control and MSBuild assist with automating the build and deployment process. Additional steps such as code reviews serve as a preventive learning step to proactively review code and reduce defect rates over time.

Collaboration Tools & Knowledge Repositories

While there is not a focus on documentation, Agile does emphasize team collaboration and knowledge sharing. Utilizing collaboration tools such as Microsoft SharePoint in the execution of a project can assist with both creating the transparency needed for success and almost incidentally ends up being a repository for team information that serves as project artifacts and documentation.



SharePoint lists are a good tool for managing and communicating requirements, including follow-up conversations with customers. The ability to search these lists provides future development and maintenance team members a requirements repository to understand decisions made by previous developers and customers. Additionally, while most Agile projects rarely have an upfront concerted effort to create extensive technical documentation, a Wiki can serve as a place to organically grow and store such documentation over time.

Many managers of Agile projects will choose to allot time at the end of each iteration for explicitly creating and updating some minimal level of documentation.

Conclusion

Project management has come a long way since the 1994 CHAOS report. Lessons learned between the 1994 report and the 2006 report have shown over a 100% increase in successful projects. Agile methodologies can help increase project success more by mitigating some of the critical impairments projects face.

Agile methodologies also come with their own set of new, unique challenges that must be addressed. As discussed previously, there are various tools and techniques that have been used to alleviate some of these challenges.

The CHAOS report will likely continue to report on chaos, well into the adoption of Agile methodologies and those methodologies that succeed Agile. As such, it should be noted that although Agile methodologies may increase the likelihood of a projects success, they alone are not enough to drive a successful project. Eric D. Brown, said in an interview with InfoQ, "It isn't the process or methodology used... it is the people involved that will help a project succeed. Of course, processes help, but without good people and an organization that helps those people succeed, any project management methodology will most likely fail."

Samir Ray is a Principal and Dipesh Patel is a Manager at Pariveda Solutions, Inc. based in Dallas, Texas.

About Pariveda Solutions, Inc.

Pariveda Solutions (Pär-ē-vā-da) works with organizations to improve their profitability through the deployment of process and technology. Pariveda delivers solutions in the areas of IT Strategy, IT Executive Advisory services, Program and Project Management, Application Development, System Integration, CRM and Business Intelligence. Pariveda's goal is to establish relationships with clients on a local level, offer and deliver high value solutions.

Pariveda Solutions was ranked the 16th fastest growing company in the Dallas Business Journal's 2007 edition of the Dallas One Hundred, comprised of the 100 fastest-growing private companies in the DFW Metroplex. Pariveda Solutions was also recently named to the Dallas Business Journal's "Best Places to Work" for 2008 as well as one of Consulting Magazine's 7 "Small Jewels" for 2008. Launched and headquartered in Dallas, Texas, Pariveda Solutions has grown to over 150 employees since 2003. The company has additional offices located in Chicago, Denver, Detroit, Houston and Seattle.



References

Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., Highsmith, J., et al. (n.d.). *Agile Manifesto*. Retrieved 6 29, 2008, from Agile Manifesto: <http://www.agilemanifesto.org>

Cohn, M. (2004). *User Stories Applied*. Pearson Education Inc.

Pariveda Solutions, Inc. (n.d.). Agile Toolkit. Pariveda Solutions.

Project Management Institute. (2004). *A Guide to Project Management Book of Knowledge*. PMBOK® Guide.

Standish Group. (1994). *CHAOS Report*. Standish Group.

Standish Group. (2006). *CHAOS Report*. Standish Group.