Using Agile Alongside the PMBOK

Mike Griffiths, Senior Technical Director, Quadrus Development

Abstract

As the use of both agile and traditional methods continues to grow, the need for guidelines on how the two approaches may be used together will increase. This paper identifies the growth of both agile and formal approaches to project management then examines the origins and key elements of the two methods. The characteristics of software projects that lead to challenges in applying traditional project management approaches are examined and agile alternatives introduced. Finally, recommendations for combining agile and traditional project management approaches are provided and the use of agile inspired project execution and control processes are discussed.

Introduction

Agile methods such as Scrum, DSDM, XP and FDD are gaining popularity in the software development sector. Increasingly companies are reporting large productivity gains and increased business satisfaction with the systems developed using these approaches.

The Shine Technologies "Agile Methodologies Survey" [1] of 2003 was one of the larger studies conducted, surveying 131 organizations. The results make an impressive case for the effectiveness of agile methods:

- 93% said team productivity improved
- 88% found the quality of applications was better
- 83% experienced better business satisfaction with the software

Other studies by Stapleton (1997) and Maurer (2002) also report improvements when using agile techniques and bolster the case for their wider adoption. However, agile methods promote project management techniques that are very different from that of traditional project management guidelines. Concepts such as:

- Not attempting to finalize the requirements early in the project
- Promoting the incorporation of change requests throughout the lifecycle
- Less emphasis on rigid upfront planning

The apparent disconnect between agile and traditional project management principles could be dismissed as an interesting anomaly if it were not for an impending conflict. Due to the increase in use of agile methods and a similar increase in adoption of traditional project management approaches, an escalating number of organizations are finding themselves attempting to resolve these seemingly diametric principles.

Traditional project management methods are also on the increase. PMI membership statistics indicate a continued increase in popularity; growth is exceeding 20% per annum with no sign of a slowdown. The result of an increase in adoption of both agile and traditional methods is that confusion and conflict often arises due to the mismatch between the underlying project management principles and techniques of each of these approaches.

Agile methods are not just a way to develop software that can be wrapped by traditional project management. Agile methods require the adoption of particular project management approaches to enable them to perform. The scope of agile methods goes beyond development team activities and changes the way sponsors, users and other stakeholders are engaged. Agile approaches also employ fundamentally different project planning, executing and controlling processes that are integral to their success. While it is perfectly acceptable to use existing traditional project management documents like Vision Statements to communicate externally, the internal processes for running an agile project are quite different than traditional approaches.

Why software development provides project management challenges

Software development projects present unique challenges for traditional project management approaches. First, software is intangible and difficult to explain well; rarely is the same system built twice which makes analogy to existing functionality difficult. These issues can lead to "**evaluation difficulties**", where mismatches develop between interpretations of original requirements and customer goals.

Iterative development has emerged as a technique for overcoming these evaluation difficulties by utilizing regular review points. In addition to reviews, closer collaboration between users and development teams and subsequent adjustments allows the system to evolve towards the true business requirements. When users are given an opportunity to refine requirements based on evaluating early prototypes, it is common for there to be significant differences between the originally stated requirements and the true business needs.

Secondly, the process of executing a software project is a complex and high risk activity. Unlike many construction-based projects, writing software in today's rapidly evolving languages is not a defined, repeatable process. Often issues arise when attempting to combine technology A release x, with technology B release y. This is less of an issue in industries that are more mature, we do not hear statements like "Nails3.2 is not compatible with wood 6.4, or steel 2.6 no longer works with concrete 5.1" yet similar problems occur when using new computer technology.

Software development is often an unprecedented, research and development based process. Attempting to create detailed task-oriented plans for software developers is likely to lead to fragile, soon abandoned plans or much project management time spent updating plans rather than managing the project.

Finally, well-structured software differs from conventional engineering by allowing certain changes to be made even late into the project. For example, validation logic could be moved from a client tier, to a middle tier, or even implemented as a trigger in the database tier with minimal impact on the remaining application. Such a change to the design is not usually possible in a physical engineering project. This "**extreme modifiability**" coupled with the inherent problem of defining requirements accurately the first time, allows software project to add or evolve requirements to deliver maximum business value against a backdrop of changing business needs.

Why agile methods work for software development projects

Agile methods have emerged that acknowledge the evaluation difficulties of software, leverage extreme modifiability and handle execution risk. They recognize that validation of requirements is best done early and often against an evolving prototype. Agile methods employ prioritized requirements schemes that encourage changes to be traded-off against original requirements throughout the lifecycle, thereby allowing the delivery of late changes that can provide competitive advantage to business.

Finally, agile methods do not attempt to micro-manage developers via prescriptive task plans. Instead mechanistic processes, premature decomposition and one way task communications that belie the volatile nature of software development are swapped for more humanistic, goal directed approaches that utilize people's ability to manage complexity.

Agile project management processes

The basic elements of agile execution and control processes are depicted in Exhibit 1. This diagram does not show all of the project activities, but serves as a guide to the different execution and control processes discussed in more detail later.

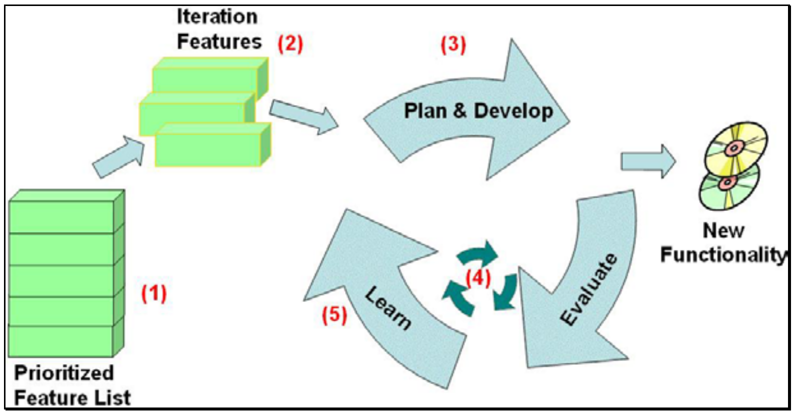

Exhibit 1 – Agile Control and Execution Processes

In Exhibit 1, (1) represents a prioritized list of requirements or features that are to be built. Requirements are assigned a priority by a business (user or sponsor) representative to rank their value or strategic importance. A subset of these prioritized requirements or features is then selected (2) for development. The selection is based on the highest priority features remaining to be developed.

This subset of features then undergoes analysis, development, testing and evaluation during a short, fixed time iteration (3). Recommendations for the durations of these iterations vary between agile methodologies, but 2-4 weeks is common.

Within this 2-4 week cycle there is a daily communications and risk assessment cycle (4). This is conducted as a short daily meeting where development team members briefly answer:

1. What have you been working on since the last meeting?
2. What are you working on today?
3. Do you have any problems, issues or impediments to making progress?

Via these short daily meetings, project stakeholders can hear about incremental progress. Team members learn what others are working on. Also, roadblocks and risks are raised quickly for removal and mitigation by the project manager.

Agile methods also perform mid-project retrospectives (5) where the lessons learned questions of "What went well?", "What did not go well?", and "Recommendations for the future?" are captured and factored into the planning of the next iteration.

Many people feel that these techniques appear moderately useful but are no replacement for the rigors of a traditional, structured approach to project management. However, the adoption of these approaches actually represents key elements of a shift to a more modern set of management theories.

Project Management Theories

The science and theory of project management is rooted in the theory of management and the theory of production. Since a project is a special type of production system that usually gets executed only once, there is value in understanding the history and evolution of production theory.

1900 – 1950 The Transformation View

Early in the 20th century Fredrick Winslow Taylor created his "time-in-motion" studies[2] which promoted the idea of specialization to achieve local optimizations and cost savings. Henry Ford was a fan of this approach and in the 1920s used it as the basis for the production line approach to manufacturing cars. The basic idea was that complex tasks could be repeatedly broken down into simpler and simpler sub tasks until all the basic Input-Process-Output transformations are identified and then addressed via specialized resources, this minimizing local costs..

1950 – 1980 The Value View

The next leap in production theory came in the 1950s when Peter Drucker introduced the holistic view and ideas of customer value[3]. The true value of a product is what the customer will pay for it and not based on the cost of production. Later in the 1970s, Michael Porter refined the view to introduce the idea of value-chains[4] and the concept of workers adding value to raw materials through the production process was born.

1980 – 2004 The Constraints/Lean View

The next revolution was in 1984 when Eli Goldratt introduced the Theory Of Constraints[5] which proposed that overall capacity was limited by bottlenecks or constraints. The true role of management is to identify and remove these constraints to improve the productivity of the system as a whole. Later in 1994, Peter Senge united the idea of value chains and considering the production system as a whole, in his groundbreaking book "The Fifth Discipline"[6]. He explained how value-chains and businesses are complex systems and in order to maximize the effectiveness of a business, the system must be considered as a whole and not the sum of its individual parts. This was a major break from the mass manufacturing ideas that were still based on Taylor's local optimizations theory. It occurred when western manufacturing was loosing ground to Lean based approaches from Japan and put an end to the view of local optimizations as the goal for mass production.

Production science analysis

The view today is that all of these ideas (Transformation, Value Chain and the Theory of Constraints) are in play and each approach can bring better insights into understanding the production process and the best places to invest effort to improve performance.

This production theory is of interest to project managers because as Hal Macomber[7] observes "*If the theory in use is obsolete, then it would explain why we do not get the results we are after. Like earlier astronomers who thought the sun revolved around the earth. There are anomalies of project performance that can't be explained. It is with the intent of explaining past behaviour and predicting future behaviour that we are interested in theory*."

Agile methods as examples of more modern management science

Re-examining the agile lifecycle we can see examples of some of the more modern management science in use. The development team are the production workers in the software production process. They create business value by transforming requirements into software functionality. The role of the project manager is to maximize this value delivery by removing impediments from the development team, as shown in Exhibit 2.

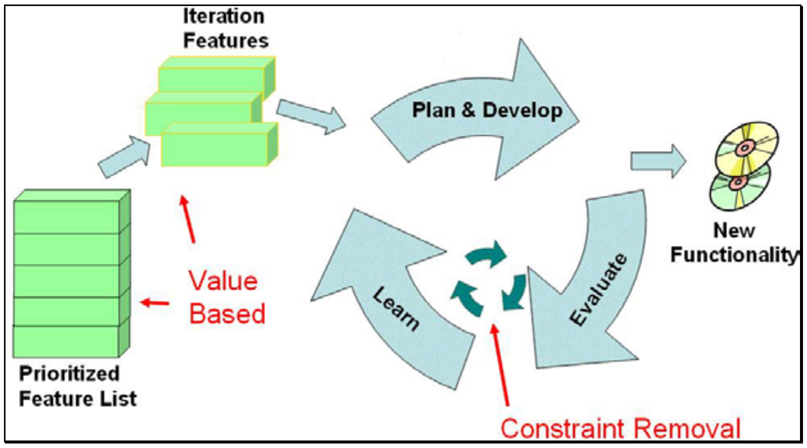

Exhibit 2 – Agile application of modern production best practice

The use of prioritized-feature-lists and the selection of the highest priority features for development in the next iteration is an example of Drucker and Porter's Value based views of production in use. Likewise, the third question in the daily team meeting which asks "Do you have any issues or impediments to making progress?" and the subsequent removal of these impediments is an example of the project manager taking a theory of constraints view to the project. This is achieved by looking for constraints in the developers working practices and then removing them to enable the developers to deliver more business value.

So, rather than agile methods simply offering some useful tools for software projects, they actually represent concrete application of modern production theory best practice.

Traditional Project Management Guidelines

These agile approaches are in contrast to traditional views on project management. The current PMBOK view of projects is dominated by the Transformation concepts of Taylor. It recommends early in the lifecycle, to break down the project into smaller and smaller pieces using techniques such as creating Work Breakdown Structures.

Current project management guidance is focussed heavily on planning. The overriding theme is that if you plan enough, track against the plan and take corrective action when work deviates from the plan, then you will be successful. If things go wrong then you probably did not spend enough time upfront planning and understanding all the requirements and risks. These recommendations are particularly problematic for software development projects where volatile initial requirements and unprecedented technology combinations represent major production risks.

Project management activities, as defined by the PMBOK, are divided into the process categories shown in Exhibit 3.

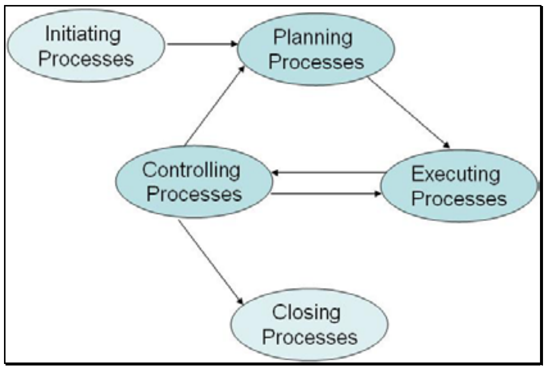

Exhibit 3 – Project Management Processes

Traditional project management guidelines are heavy on project Planning information, but light on project Execution and Control information. The assumption is that execution of the plan should be a simple matter of carrying out the tasks listed in the project plan. However, this is not the case for software development projects which have high risks associated with production.

The PMBOK promotes planning is the central role of the project manager and implies that communicating the planned tasks should be via a top-down dispatching model of work products. Control of the project is done via checking progress against the project plan and when necessary taking corrective action. This model of control is called the thermostat model. The assumption is that the underlying plan is correct and something has happened on the project that needs correction in order to bring it back in line with the plan.

The limitations of a management-as-planning approach, the dispatching model for task assignment, and the thermostat model of control are well documented by Koskela and Howell[8]. The remainder of this paper describes how agile approaches can be utilized alongside or instead of the PMBOK based approaches, for better project execution and control in software development projects.

Mapping of Agile and Traditional Project Management

The processes shown in Exhibit 3 will be used to group discussion around the project management best practices for software development. The following sections summarize author recommendations for managing software projects.

Initiating Processes

Recommendation: Use As Is: The early project activities focused on chartering and identifying preliminary scope work equally well for software development projects as they do for more defined repeatable projects. While writing the Project Charter, a description of the approach that will be used to deliver the project should be made clear. When using agile methods, the concepts of iterative development and the production of regular increments of software should be clearly explained to all stakeholders as this may represent a different way of working for some people.

Planning Processes

Recommendation: Use with Modifications: Planning is a critical step for both traditional and agile project management. However, its use should not be concentrated upon at the beginning of the project as this is when least is known about the project. Instead, planning needs to be iterative and ongoing throughout the project; guided by feedback from actual project performance, business changes and technical knowledge growth.

This important point is made in the PMBOK through the concepts of Progressive Elaboration and Rolling Wave planning. Progressive Elaboration speaks to the need to refine scope and evolve plans as more information becomes available. The related concept of Rolling Wave planning which states planning should be iterative and ongoing based on short time horizons.

Unfortunately these concepts are often overlooked; a contributing factor may be the artifacts produced by early planning. Gantt charts and Work Breakdown Structure are difficult and time consuming to update for non trivial projects. Also the tools used to create them do not readily support wholesale refactoring. As a result, the majority of software projects managed with traditional techniques do not undertake the rigorous plan refactoring required to keep pace with the true project work. Either detailed task oriented plans become outdated, or high-level, phase oriented plans are produced that offer little value in way of project tracking or forecasting.

For software projects that contain requirements and technical uncertainty, plans should be feature based instead of developer task based. Attempts to predict the detailed activities required for problem solving will likely fail. Plans should instead be maintained at a high level for the entire project; outlining feature themes (general areas of development) and at a detailed level for the current and subsequent iteration, defining the features selected for development. This switch from tasks to features also affords better visibility into the value of functionality developed to date. Non-functional requirements and risk mitigation activities can be listed as features and assigned a business value by calculating the expected monetary value of the risk they have been created to mitigate.

Executing Process

Recommendation: Use Agile Techniques: The PMBOK is an industry independent guide to project management best practices. As such it contains very little about how to actually undertake the work on any project. The intention is that the appropriate “doing” techniques will be used from the industry in question. For example, there is a wealth of information available externally about how to run civil engineering projects. Therefore, for software projects the most appropriate execution processes should be employed. These include:

Develop iteratively – build software iteratively and incrementally to address the “evaluation difficulties” of software. Iterative development also supports the rapid mitigation of technical risks by allowing elements of new technology to be trialed in early iterations rather than waiting until an integration phase towards the end of the project and then discovering issues when there is little time left of their resolution.

Iterative development also provides many additional advantages over single-pass approaches, including: better progress visibility, learning opportunities, and options for early benefits realization via interim releases.

Use meaningful metrics – The act of measuring a characteristic often influences its value, this is a trait called the Hawthorne effect [9] and it should be noted and used to good effect on software projects. Also, the metrics employed to track software projects should be “Simple and relevant to the goal” [10].

Attempts to track metrics that are not relevant to the goal such as Lines-Of-Code developed or developer-hours-worked are likely to lead to an increase in the lines of code written and increased hours worked (due to the Hawthorne effect). These are not relevant to the true goal of delivering valuable software and an increase in the lines of code written is undesirable as it leads to increased complexity and maintenance effort. Likewise increased hours worked can to lead to developer burn-out and an increase in defect rates.

Instead, metrics that are aligned to the true goal should be used as the primary measures. This is why metrics such as features-delivered and project-time-remaining are frequently used as agile project management measures. Features-delivered can be illustrated with a Cumulative Flow diagram as shown in Exhibit 4.

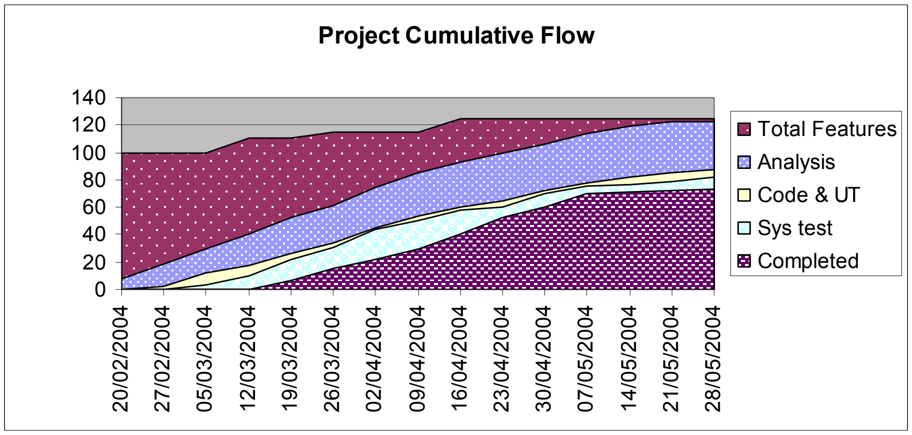

Exhibit 4 – Cumulative Flow diagram

In Exhibit 4, time is shown on the X axis and the total number of features to develop is shown on the Y axis. Increases in scope can be seen as steps in the overall features to develop and progress shown as the areas and gradients representing development work.

Empower the team – A traditional top-down, dispatching model for allocating work is problematic for software projects that are hard to estimate and often contain unpredictable dependencies. Attempts to micro-manage developers by assigning individual tasks are likely to fail. Instead it is better to leverage people's ability to manage complexity by assisting in the definition of high-level goals and objectives and then empowering the team with the responsibility of achieving the goals.

In practical terms this means engaging the team and business representatives in the planning of features, then letting the team select and develop the features for that iteration as they see fit. Progress may be made sequentially through features, in parallel, or via a combination of both as issues and dependencies are uncovered and solved. Overall progress can be tracked via the delivery of completed features but the complexity of feature dependency resolution is best managed by the team.

Controlling Processes

Recommendation: Use Agile Techniques: The controlling processes described in the PMBOK include "Controlling the project" and "Controlling changes". The core theme is looking for variances from the project plan and then taking the appropriate action. This is a thermostat model of control and implies the project plan is correct and a deviation from it requires intervention.

In software projects where there is a high likelihood that the initial plans are flawed due to evaluation difficulties and execution risk driven changes, a thermostat model is not appropriate. Instead scientific experimentation and root cause analysis is required before correcting the plan as opposed to correcting the project. In unprecedented software projects, plans represent today's best forecast of how we think the project will unfold, as opposed to a defined path that must be followed.

Iteration Reviews – At the end of each iteration progress, requirements, risks and process effectiveness are reviewed. Having these regular checkpoints assists with project steering and provides updated trend and goal setting information.

Change Requests, Defects and Risks – Change requests and defect reports can be prioritized in with the list of remaining features to be developed. This way the impact of scope creep is immediately apparent as new features need to be traded off against existing features planned for an iteration in order to stay within the iteration timebox. Using a similar approach, risks mitigation actions may be inserted into the prioritized feature list based on the expected monetary value of the risk the will avoid or mitigate.

Controlling Flow – By taking a holistic view of the development process, bottlenecks or constraints in the feature delivery may be identified and removed. In this way the delivery of features to the business can be maximized and the principles of Lean production and the theory of constraints utilized. Cumulative flow diagrams can be used to identify constraints in the development process. Widening portions in the graph highlight a slower activity and can alert managers to the need to exploit the constraint.

Closing Processes

Recommendation: Use As Is: The Closing Processes defined in the PMBOK provide good general guidelines for activities to consider when closing software development projects.

Recommended Project Management Guidelines

Some of the agile based ideas outlined in this paper are already being incorporated into the next version of the PMBOK. During the December 2003 "Call for PMBOK Revisions", a number of agile related suggestions were suggested. One of those accepted was the recommendation of daily meetings where the development team members are asked the 3 questions:

1. What have you been working on?
2. What are you working on today?
3. Do you have any problems or impediments to making progress?

The project manager then takes this list of impediments raised in question 3 away as a to-do list of issues to solve, which is an application of a constraints view of development and the lean principle of maximizing throughput.

Summary

Many software projects possess characteristics that challenge traditional approaches to project management including volatile requirements, evaluation difficulties and significant execution risks. Agile approaches address these issues through iterative development, goal seeking, and leveraging the extreme modifiability of software. When undertaking projects with high execution risks, agile methods should be used along side the appropriate traditional project management techniques to provide additional project execution and control mechanisms. Agile approaches to project management also offer alternative tracking and reporting metrics which do not create large change workloads as projects progress.

References

1) Johnson M (2003) Shine Technologies: Agile Methodologies: Survey Results
http://www.shinetech.com/resources/ShineTechAgileSurvey2003-01-17.pdf
2) Frederick W Taylor. (1985 Originally 1911). Principles of Scientific Management. PA: Hive.
3) Drucker P (1993 Originally 1954) The Practice of Management, NY: HarperBusiness;
4) Porter M (1985) Competitive Advantage, NY: Free Press
5) Goldratt E (1984) The Goal, MA:North River Press
6) Senge P (1994) The Fifth Discipline, Currency
7) Macomber H (2003) Reforming Project Management Weblog http://weblog.halmacomber.com/
8) Koskela1 L and Howell G (2002) The Underlying Theory Of Project Management Is Obsolete, PA: Project Management Institute
9) Gillespie R (1993) Manufacturing Knowledge: A History of the Hawthorne Experiments, Cambridge,UK: Cambridge University Press
10) Reinertsen D, (1997) Managing the Design Factory, NY:Free Press