Rally Software Development Corporation

Whitepaper

5 Levels of Agile Planning: From Enterprise Product Vision to Team Stand-up

Hubert Smits
Agile Coach and Certified ScrumMaster Trainer
hubert@rallydev.com

101706

Abstract:
This paper evaluates agile practices when applied in multi-team and multi-person-year projects. It does so through inspection of the requirements brought on by scaling projects, and applying basic agile principles to those requirements. In the introduction section the main agile principles are introduced, as well as the Lean principles upon which the agile methods are built. One of those Lean principles, Muri or overburdening of people, is addressed through the extension of the agile planning process. The extension of the most used agile planning technique (iteration planning) is described in detail, both the motivation for the extension as well as the collaboration practices behind each planning level. In the final chapter the impact of product complexities on the planning process is evaluated, and a solution to create a smooth flow in the planning/delivery cycle is presented.

Table of Contents

Introduction

Why Focus on Planning Activities?

Experience gathered during large-scale implementation of agile concepts in software development projects teaches us that the currently popular agile software development methods (like Scrum) do not scale to program, product and organization level without change. The fundamentals for changes to these methods are found in Lean principles, or: the future of agile methods is found in its origins. This paper describes a planning framework that has been used successfully in large-scale[1] agile projects and investigates the impact of introducing this framework on three core Lean principles: Muri, Mura and Muda[i].

Agile and Lean

The origins of Scrum are found in writings by Takeuchi and Nonaka[ii] and were developed into a methodology by Schwaber, Beedle and Sutherland[iii, iv]. In 2003, the principles of agile software development were linked to Lean principles[v], which concept was further explored by Sutherland, Viktorov and Blount[vi]. This thinking has been further developed into a process that adds practices to the Scrum framework to address complexities that arise when the Scrum framework is applied to software development programs with multiple teams and/or multiple projects[2]. This paper is based on these publications and experience from the author in multiple implementations of agile practices in large projects and programs.

Agile Vocabulary

This document is heavy on jargon, some often used terms are:

- Agile Methods – The collection of lightweight software development methods that have been developed based on the Agile Manifesto[3]. Examples are Extreme Programming (XP), Scrum and Feature Driven Development.
- Product Owner – The person responsible for the success of the product in the market, and therefore entitled to prioritize the needed features of the product.
- Delivery Team – The group of people responsible for the delivery of the artifacts that together make up the product. They are responsible for delivering the right quality and can therefore determine and estimate the tasks involved in the delivery of the product features.
- Muri, Mura, Muda or Load, Flow, Waste – Three core Lean principles that describe the need to plan work correctly (load), to create a regular pace in the team (flow) and to avoid delivering products that have no customer value (waste).
- Product backlog – A prioritized list of features that make up the product as desired by the product owner.
- Iteration – A one- to six-week period in which the delivery team produces working (accepted) product features. Also called Sprint in the Scrum methodology.
- Feature – A product component that the product owner and customer recognize and value. Delivering a feature is done in tasks, which are the more technical activities that are recognized and valued by the delivery team. Estimation of the work involved in delivering features and tasks is done by the delivery team, which is a major difference from plan-driven development where the estimating task is often taken on by the project manager.

Core Lean Elements – Muri, Mura, Muda

The three Lean principles that are used to define agile software development methods find their origins in the Muri, Mura and Muda concepts:

---

[1] The projects referred to as being 'large' typically last well over a year, and have 50 or more people involved to deliver the required functionality.

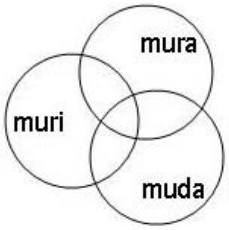
2 Sutherland, Tabaka & Smits – Program Management with Scrum – Two day course for ScrumMasters – 2006

3 http://www.agilemanifesto.org/

- Muri – Overburdening of people or equipment
- Mura – Unevenness in workload
- Muda – Waste or non-value adding activities

There is a relation between the three principles: mura creates muri leading to the inability to reduce muda[4]. Or: waste can be reduced through solving problems with imbalanced loading and overstraining of people[vii]:

All agile methods rigorously reduce existing software development methods to frameworks that are best described as 'barely sufficient'. When these minimal frameworks are combined with discipline and rigorous inspect-and-adapt practices, robust methodologies are created that deliver working software early.



The existing agile methods have a focus on small (single team, duration measured in months) projects, and the impact of large (multi-team, multi-year) projects on agile practices is not addressed in agile methods. This paper studies the requirements for the agile planning process in relation to the overburdening of individuals and teams (muri).

Planning in Large Scale Agile Projects

In agile methods, loading a team with work is done through iteration planning. Due to the shortness of the iteration (typically one to six weeks) a *plan* reduces in importance and *planning* gains in importance. For small projects it may be sufficient to plan only a single iteration at a time. The experienced disadvantage of iteration planning when applied to projects that run for more then a few iterations or with multiple teams is that the view of the longer term implications of iteration activities can be lost. In other words: the view of 'the whole' is lost. A solution is to add planning levels to incorporate the current view of 'the whole'. In plan driven and waterfall methodologies this problem is overcome through a large upfront design, aiming to predict accurately how much work is involved in each project task. This leads to a large investment early in the project, when it is by no means certain that the designed functionality is actually the functionality desired by the product owner. An approach with multiple levels of planning has to avoid the reintroduction of the big design up front.

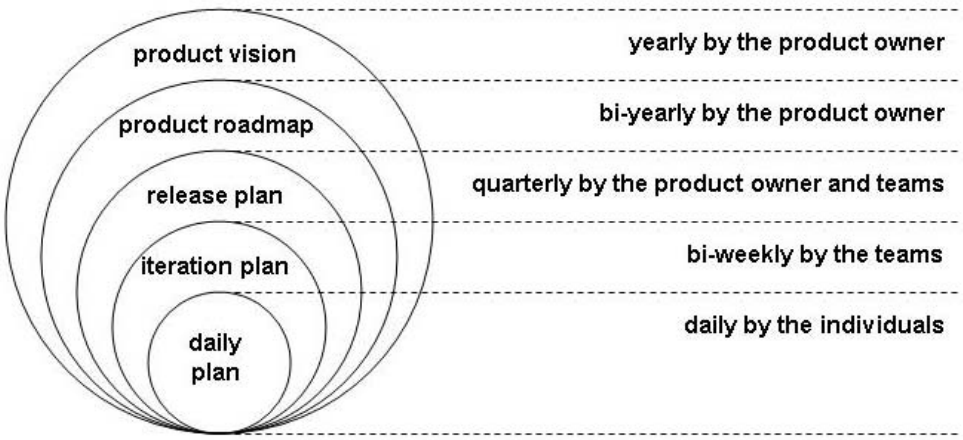
Planning activities for large-scale development efforts should rely on five levels:

- Product Vision
- Product Roadmap
- Release Plan
- Sprint Plan
- Daily Commitment

The certainty of undertaking activities addressed in each of the five levels increases, and therefore the amount of detail addressed (money invested), the number of people involved and the frequency can increase without running the risk of spending money on features that may not be built or may be built differently.

Each of the five levels of planning addresses the fundamental planning principles: priorities, estimates and commitments.

[4] Womack: http://www.leanuk.org/articles/mura_muri_muda.pdf



Product Visioning – Level 1

The broadest picture that one can paint of the future is a vision of a product owner. In this vision she explains how an organization or product should look. She indicates what parts of the system need to change (priority) and what efforts can be used to achieve this goal (estimates and commitments).

Product Visioning – How To

Possible structures for a visioning exercise are to create an elevator statement[viii] or a product vision box[ix]. The principle of both exercises is to create a statement that describes the future in terms of desired product features, target customers and key differentiators from previous or competitive products. Geoffrey Moore uses the following structure in his elevator statement: "For (target customer) who (statement of the need) the (product name) is a (product category) that (product key benefit, compelling reason to buy). Unlike (primary competitive alternative), our product (final statement of primary differentiation)." The product vision describes a desired state that is 12 months or more in the future. Further planning (design) activities will detail the vision, and may divert from the vision because the future will bring us a changed perspective on the market, the product and the required efforts to make the vision reality.

Product Roadmap – Level 2

The era of large-scale projects that deliver results in years is behind us. Customers demand more frequent changes and typical time-to-market timeframes are measured in weeks or months. The higher frequency and smaller timeframes force a product owner into thinking in steps, into thinking of a road towards the final product. Just like a journey is planned upfront and shared with the fellow travelers, a product roadmap is created and communicated to fellow delivery people. The goals for doing so are for the product owner to:

- Communicate the whole
- Determine and communicate when releases are needed
- Determine what functionality is sufficient for each release
- Focus on business value derived from the releases

The delivery team on the other hand will:

- See the whole
- Learn about the steps to realize the vision
- Learn the business priorities
- Provide technical input to the roadmap
- Provide estimates for the projected features

Product Roadmap – How To

The creation of the roadmap is largely driven by the product owner (or product owner team). This stage of the program has limited influence of technology constraints. In a meeting or series of meetings the roadmap

will be drawn by the product owner. This can be quite literally, through a graphical representation of the releases, or more formally in a written document outlining the dates, contents and objectives of the foreseen releases.

Product backlogs

In anticipation of the next planning stage (release planning) a list of desired features needs to be built - the product backlog. In its simplest form, such a backlog is a table (spreadsheet) of product requirements, briefly described so a delivery team can provide estimates for the realization of each feature. Most importantly, the list has to be prioritized. The success of an agile project depends on the early delivery of the highest priority features. Since the success of a project is measured in business terms, the prioritization of the feature list is the responsibility of the business, i.e. the product owner. Interaction with the delivery teams is required. Without a discussion of the features it will be hard for the delivery team to produce estimates that have an acceptable inaccuracy. Characteristics of a product backlog include:

- One product backlog for all teams (see the whole)
- Large to very large features (up to 20 'person days' to deliver a feature)
- Feature priority based on business priorities (discovered through market research)
- Technology features (sometimes called non-functional features, work required to make the product work in a desired way, e.g. the implementation of a certain DBMS in order to warrant a certain system performance) are limited to those that have a direct impact on the success of the product in the market

Release Planning – Level 3

In small projects (single team delivering a product increment in a few iterations) the product backlog alone can provide enough project overview. The size, duration and deliverables are easily recognized. There is no need to synchronize or group deliverables or teams. This changes when applying agile concepts to programs in which there will be dependencies between teams and between projects. Some of the teams may be operating according to different agile principles, or are operating in a waterfall style. The first moment to start grouping activities and allocate them to teams occurs during release planning.

A release is what the name implies: a set of product increments that is released to the (internal) customer. Some characteristics of a release are:

- Releases are defined by date, theme, and planned feature set. Release dates can be linked to events, like conferences or changes in the legal system
- Scope, not date or quality, is the variable, which highlights the need to use a prioritized product backlog as the basis of a planning event. When a release date is set in stone, and an increase in budget is unlikely and usually has no positive effect on a release, the scope is the only variable that can make or break the release date
- All teams must commit to the same rhythm of iterations. When all teams work to the same rhythm the discovery and management of dependencies occurs automatically during the planning activities.
- There are fixed release dates across all teams of the program: a typical interval would be two to four months.

Release vs. iteration characteristics

A release is defined at system or program level, usually in product owner vocabulary. The release theme, release date and prioritized features together form a release; all these concepts are defined by the product owner. When releases are seen in the perspective of the roadmap, then the high visibility and confidence is in the near term (release "current" and "next"). This is expressed through more detail in the feature descriptions, and a smaller size of the individual features. A release can stretch over six to nine months, although two to four months is more common. The purpose of a release is to provide both customer and team with a view of 'the whole' and have the product owner and delivery team make a first attempt of decomposing high level requirements into product features, guiding them to higher detail in features that are likely to be built.

The iteration, on the other hand, is defined at a feature level. The delivery team and product owner have agreed on the number of iterations in a release. Which features will be delivered in an iteration is determined by their priority; the number of features that will be delivered in the iteration is set by the velocity of the team and the team estimates of the features. Planning activities for the iteration is typically a single iteration at a time, or sometimes two. Planning only for the next iteration implies that the activities have a high likelihood to be undertaken. This warrants the investment of time by the delivery team into decomposition of features into tasks. Tasks are technical in nature; the product owner will not necessarily recognize them. Iteration lengths vary from one to six weeks, with two weeks seen as the most frequent duration. The purpose of an iteration is for each team to commit to delivering accurately estimated features.

Release Planning – How To

With the product vision and product roadmap available and regularly updated and reviewed by the product owner and the team, everyone is aware of the focus, or theme, of the next releases, and the desired release dates. Furthermore the product features have been collected in the product backlog and have been prioritized and estimated. Through these exercises, the product owner and the delivery teams have a shared understanding of what has to be delivered, when, and why in the agreed order. If the team has estimated the features in storypoints or T-shirt sizes[x] then they have to agree on their capacity (expected velocity) based on numbers from the past – or by plain guessing. Typical availability of team members is 60 to 70%, but numbers as low as 25% and as high as 90% are seen.

A release planning session typically takes place over a full day, sometimes two if the program is very large (100s of team members). All team members participate in the session: product owner, full delivery team, stakeholders. The session is highly collaborative and interactive; tools used are typically sticky notes, flip-charts and whiteboards. An agenda[xi] could be:

- Introductions, goals, agenda updates
- Product vision, product metaphor explanation by the product owner
- Time-boxes for the releases and iterations
- Impact of previous releases and iterations
- Capacity calculation by the delivery team
- Agreement of deliverables (when is a feature “done”)
- Moving features into the release during break-out sessions by the individual teams. This can be a literal ‘moving’ when the features are written on sticky notes, and the releases are represented by separate flip-over sheets. All team members participate in this exercise.
- Moving features from the release into iterations within the release by the individual teams
- Determination of dependencies by walking through the individual planning results, moving features to the required iterations or releases to solve the dependencies
- Final calculation of workload per iteration and comparison with the available capacity
- Review of discovered risks and issues
- Commitment to the release plan by all attendees
- Retrospective of the session

Iteration Planning – Level 4

During release planning the delivery of features was determined, taking dependencies between the features and between teams into consideration. Due to the lack of detail (no investment in design activities until the likelihood of feature delivery is close to certain) the estimates are rough and have an acceptable uncertainty. For iterations within the release, a planning session takes place. Before or during the session, detail is added to the features through decomposing into tasks. Adding detail increases accuracy of the estimates. The actual capacity of the individual teams is known with more certainty than during the release planning session. The combination of these increased accuracies makes certain that the team can be able to commit to actually delivering a number of features during the iteration, with a high degree of certainty of actually delivering them.

Iteration Planning: How To

The structure of the iteration planning session resembles the release planning session. Although the teams work individually to produce their iteration plan, synchronization between teams will provide an effective mechanism to detect and resolve dependencies. The agenda for the planning meeting bears great similarities with the one from the release planning meeting, with the prime distinction of the planning horizon: only a single iteration is observed in all activities. The core of the activities is carried out on a team-by-team basis:

- The individual teams determine their *actual* capacity, or the amount of work it can get done within the iteration.
- The individual teams decompose as many features as seem to fit in the next iteration into tasks – this can be done in preparation.
- Every task is estimated, with a typical task size of a half a day to two days.
- The "done" definition[5] has to be taken into consideration when decomposing and estimating features: a feature is not *done* until it has been fully *tested* and *accepted* by the product owner.

The results of the individual teams are inspected in a joint session to determine dependencies (or disappearance of them) that were not seen during the release planning session.

Daily Plan – Level 5

Many agile teams have taken on the habit of a daily meeting: the Scrum stand-up meeting. In a 15-minute session, the team members update their status ("what did I do yesterday"), commit to work for today ("what will I do today") and report impediments ("I am blocked by …"). This daily meeting is not often seen as a planning session, but certainly is. The people look a day ahead, have learned from the earlier days in the iteration and tell each other what they plan on doing. Issues are raised, possibly addressed, and the success of delivering the desired features within the iteration can be determined after the meeting.

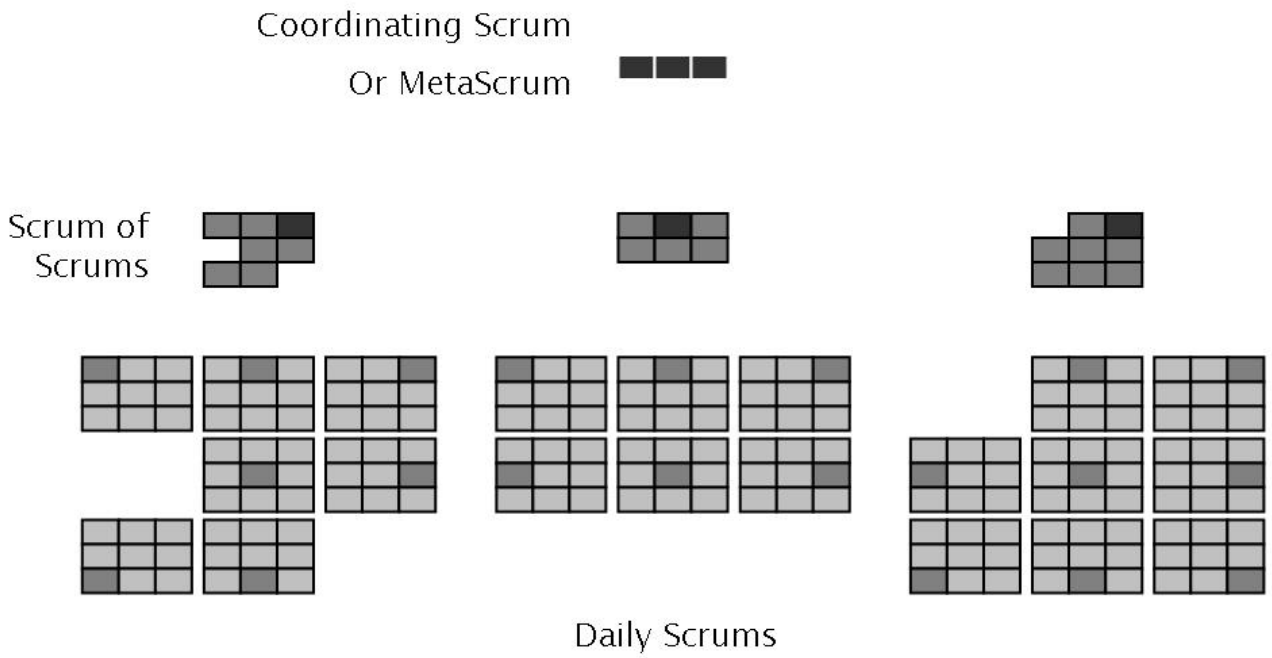
Stand-up Meeting: How To

Like other planning activities, the daily plans need to be synchronized between teams. This takes place in a coordinating stand-up meeting ('Scrum of Scrums' meeting in the Scrum methodology). Representatives of each team report the status, plans and impediments to each other in an identical format. The three questions answered are the same as in the individual stand-up meetings ("what did the team complete yesterday," "what will the team do today" and "what is blocking the team"). What is made visible is:

- How are all the teams progressing?
- What are the cross-team impediments?
- Who is taking the actions to remove them?

The principle of a coordinating stand-up meeting can be repeated to address large number of teams: representatives of 'teams of teams' report on the progress of the 'teams of teams'. These meetings typically coordinate efforts of teams that have no common ground. For example, all the IT delivery teams have a coordinating (daily) stand-up, as do the training teams, finance teams, pre-production teams, etc. On a weekly or (late in the release cycle) daily schedule, representatives of the teams meet to report progress, plan and impediments.

[5] A definition of the deliverables for a feature (when is a feature "done") lists the artifacts that need to be delivered with the actual code. This can include tests, test results, various types of documentation etc. Some of these artifacts will be defined at project level (such as documents required by auditing teams), others are team dependant (like user documentation for teams working on user interface related features).



Conclusion

This section addressed the load of the delivery teams, through planning of the delivery activities with the right level of detail at the right moments. Through the staged planning activities, a balance is found between long-term views and commitments and upfront investments in design and planning activities. Through the ownership of the release and iteration planning by the delivery teams, the risk to overburden teams and individuals is greatly reduced.

Evenness in the Planning Process

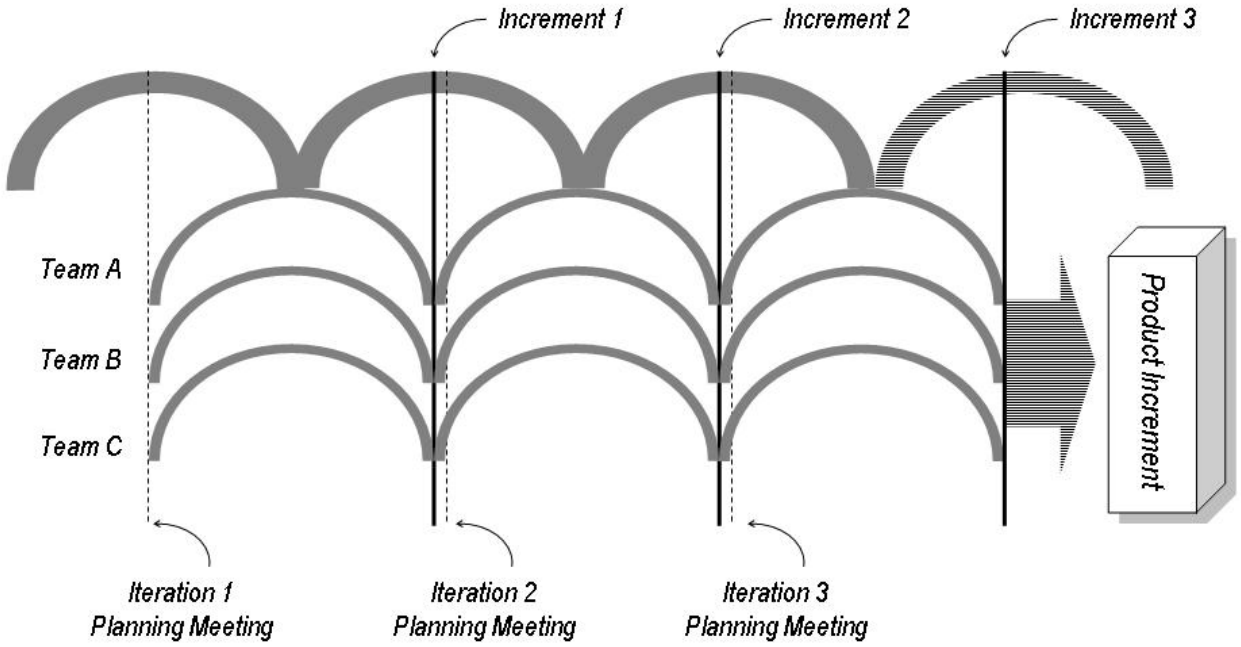
Unevenness in workload can be solved by always having an amount of work ready for the teams and individuals. This enables the teams and individuals to pull work when capacity is available.

In the paper by Takeuchi and Nonaka[ii], "The New New Product Development Game," they highlight that a start-stop mechanism introduces delay in the delivery process. Through an overlap of activities between teams, either through an early start of the next team (while the previous team is still working on their tasks), or a parallel run of most of the teams (sorting out their dependencies while progressing), the total duration of product development is shortened.

When a team is not familiar with the problem domain presented by the product owner, they require time to prepare estimates through decomposing features into tasks. This leads to delays between iterations: the first iteration delivers the requested features successfully and only then the team determines the impact of the delivery on the product backlog and starts studying and decomposing the next set of features. This work can be done in anticipation: team members (some or all) can plan for work that is likely to be undertaken in the next iteration. In a simple form (when members are familiar with the features) this will require a few hours of work to study the product backlog and question the product owner. When familiarity decreases, these activities grow to several person-days. The most extreme structure observed is where two team members spent the entire iteration studying features for the next one. At iteration planning time, they provided guidance to the team, and after the planning session they flowed back into the delivery team, taking on actual delivery roles. Two other team members then took on the preparation role. The effects of this approach are:

- Iterations overlap and are continuous without breaks
- Some features in the product backlog are pre-staged
- The top of the product backlog is always ready to be included into an iterations

This approach creates time to address problems that are caused by the lack of knowledge of the features by the delivery team. An example or activities that can be carried out during this pre-staging phase are building of prototypes of the application feature, which can be tested against a user group to study the user experiences. The result is an agreement between the product owner and the development team that backlog items are ready – the team is confident that they can estimate and decompose the features.



The development teams must set aside time to prototype for the product owner at least one iteration before

implementation. In other words, the advantages of a smooth planning process come at a price, but the effect can be significant. In the case described earlier (two team members are occupied with the preparation of the planning session), the delay between the end of an iteration and the start of the next can take the length of a full iteration, or pre-staging doubles throughput.

Conclusion

It is possible to adhere to the 'barely sufficient' principle of agile methods and extend the range of projects where the methods can be applied from single team, short-term projects to multi-team, long-term projects. The added levels of planning are not artificial or time-consuming, and they focus the right group of people on the product with the right level of detail, thus avoiding spending large amounts of time and money before the actual delivery of features begins. Understanding of the term "acceptably inaccurate" is of importance, when any member of the teams desires to hang on to details of work specification and planning then the agile implementation in on its way back to waterfall methods.

References

[i] Liker, "The Toyota Way", McGraw Hill, 2005, pp. 114-115
[ii] Takeuchi & Nonaka, "The New New Product Development Game", Harvard Business Review, Jan/Feb 1986
[iii] Schwaber & Beedle, "Agile Software Development with SCRUM", Prentice Hall, 2001
[iv] Schwaber, "Agile Project Management with Scrum", Microsoft Press, 2004
[v] Poppendieck & Poppendieck, "Lean Software Development", Addison-Wesley, 2003
[vi] Sutherland, Viktorov & Blount, "Adaptive Engineering of Large Software Projects with Distributed / Outsourced Teams," in International Conference on Complex Systems, Boston, MA, USA, 2006.
[vii] Liker, "The Toyota Way", McGraw Hill, 2005, pp. 114-115
[viii] Moore & McKenna, "Crossing the Chasm", Capstone Publishing, 1999
[ix] Highsmith, "Agile Project Management", Addison-Wesley, 2004
[x] Cohn, "Agile Estimating and Planning", Prentice Hall, 2006
[xi] Tabaka, "Collaboration Explained", Addison Wesley, 2006

