Contents

The role of planning in agile development methodology....1
Introduction....1
What Makes Planning agile? 2
Goals of planning....2
Levels of planning....2
Techniques 4
Estimates....4
Prioritisation: Planning for customer value....5
Re-estimating and Re-planning....6
Re-estimating....6
Re-planning....6
Scheduling and Tracking progress....6
Scheduling: Backlogs and queues 6
Tracking 7
Advantages and potential drawbacks: Case study of a large scale distributed project (HP FutureSmart Firmware)....8
The problem....8
The solution: Taming the Planning Beast....8
Planning rhythm....9
Prioritise everything....9
Planning and Scheduling....9
Convincing and transforming the business....10
Results 11
Conclusion....12

The role of planning in agile development methodology

It is always wise to look ahead, but difficult to look further than you can see. Winston Churchill

Introduction

Traditional project planning methods are driven by the simplistic view that one can lock down budgets, schedules and scope before the project starts and then just execute these pre-planned steps. In this model project success is measured as on time, on budget delivery of an upfront precisely planned set of features. Agile planning approaches projects from a fundamental different angle. Uncertainty from estimates is acknowledged by changing requirements and priorities during a project but it embraces the idea that small planning efforts are worthwhile. The method balances the effort and investment in planning with the high probability that the plan will be revised during the project.

What Makes Planning agile?

Goals of planning

The purpose of planning is to find an optimal answer to the question of what system to build. The answer usually addresses features, resources, and schedules. As Cohn (2005) notes, estimating and planning are critical, yet they are difficult and error prone.
Traditional project planning methods use a structured phased approach and handle a projects as a sequence of steps to be completed according to Wikipedia (2013). After the project initiation follows a planning and design phase, the plan is executed and progress monitored. Finally, the project is completed. More flexible project models allow a repetition of the planning, execution and monitoring phases but generally detailed design and planning is done upfront. Changes in requirements during the project (scope creep or requirement creep) are regarded a major problem.

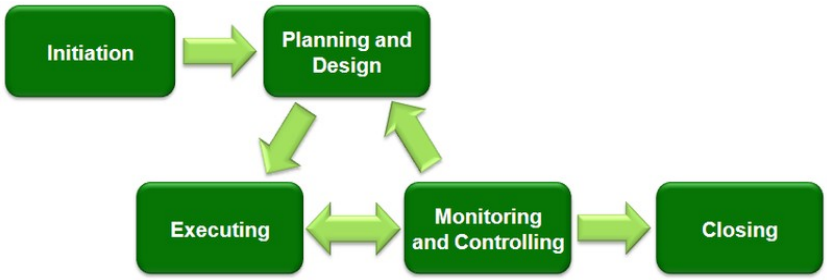

Figure 1: Project Management phases in traditional project planning methods, Wikipedia 2013

Despite our best intentions investing a large effort in planning upfront may not necessarily increase the accuracy of a plan beyond a certain level. Estimates given early in a project are expected to be dramatically less accurate than those given later. A project a schedule estimate is typically as far off as 60% to 160%, Cohn (2005). When we ignore uncertainty about how the product will be develop it may lead to missed activities in the project plan. The project then becomes more likely to be late, features must be dropped or major quality trade-offs made at the end. As planning efforts beyond a acertain level have diminishing returns we often need to expend just a fraction of that effort to get adequate results. A good plan must be sufficiently reliable to base decisions about the product and the project on it.
The fact that agile methodologies permit customers to change their requests and priorities during the project adds to the planning uncertainty but also allows for more flexibility. It also takes into account another aspect ignored by pre-planning: projects generate useful new capabilities and knowledge. Ideally, plans can be updated to include this new knowledge. As we learn more about the technologies and the team the expectations about our rate of progress and desired approach can be adjusted. Agile plans have to accommodate such changes and development teams accept that plans can rapidly become out of date. The knowledge that a plan can be revised shifts the focus from creating a perfect plan to creating a plan that is useful right now. Agile planning balances the effort and investment in planning with the knowledge that there is a high probability plan is revised throughout the project. Is focused more on the planning than the plan.

Levels of planning

Most organisations require an estimate of the costs and schedule ahead of a project. A schedule depicts the amount of time required to complete the project. There are several levels of scope to consider for planning efforts as depicted in the "planning onion" (Figure 2: The agile planning onion, Cohn (2005, p. 28)). Agile teams generally focus on the inner three levels of the planning onion each with a fixed length planning period: release planning, iteration planning, and daily planning.

Release planning is an iterative process that begins by identifying the product owner’s conditions of satisfaction for the project. These usually include goals for the schedule, scope, and resources. A common way for agile teams to express user needs are user stories. A release plan considers such user stories and covers the entire duration of the product release, typically three to six months. The release plan is usually updated throughout the project to reflect the current

expectations about the release but does not have to describe exactly what feature will be worked on during each iteration. For most projects it is sufficient to identify the stories that will be worked on in the first couple of iterations, leaving the remaining stories to be prioritised into specific iterations later.

The shorter iteration plans look in more detail at the specific work of a single iteration. This type of plan looks ahead at the start of each iteration which typically has a duration from one to four weeks. Each team can consider their unique situation to choose the right iteration length for a project. Based on the work accomplished in the last iteration, the product owner identifies high priority work the team should address in the new iteration. Large user stories of a release plan are decomposed into tasks on an iteration plan. Each task is estimated in terms of the number of ideal hours or story points the task will take to complete.

Most agile teams also use some form of daily stand-up meeting to coordinate their daily work and synchronise their efforts resulting in a daily plan.

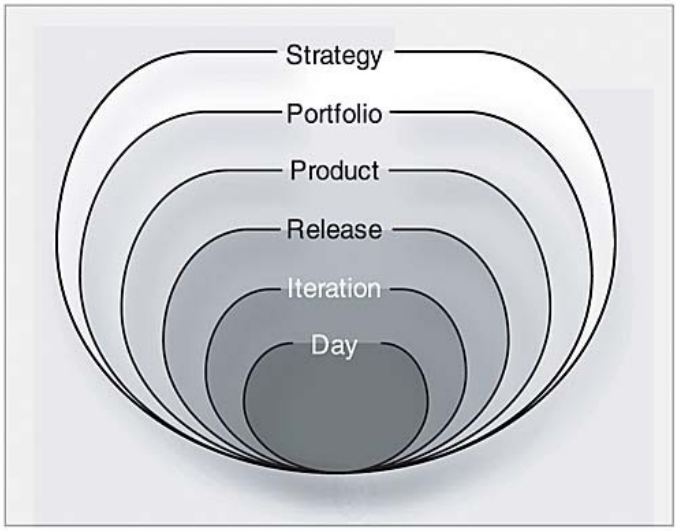

Figure 2: The agile planning onion, Cohn (2005, p. 28)

An important implication of the planning onion model shown above is according to Ambler and Lines (2012) that the planning efforts on different levels are interconnected. Iteration planning efforts must reflect elements of the release plan, such as a milestones or a dependency on an external team. In turn the iteration planning efforts could motivate changes to the release plan.

Instead of pursuing the best result possible, planning on all scope levels aim to not waste effort beyond meeting the product owner's conditions of satisfaction. During release planning a way of meeting the conditions of satisfaction for the release, including scope, schedule, and resources is identified. The product owner may need to relax one or more of her conditions of satisfaction Cohen (2005). Figure 3: Iterative release planning, (Cohen 2005, p.133) depicts the process of release planning discussed so far. A similar process occurs during iteration planning when the conditions of satisfaction are the new features that will be implemented and the high level test cases that demonstrate the features were implemented correctly.

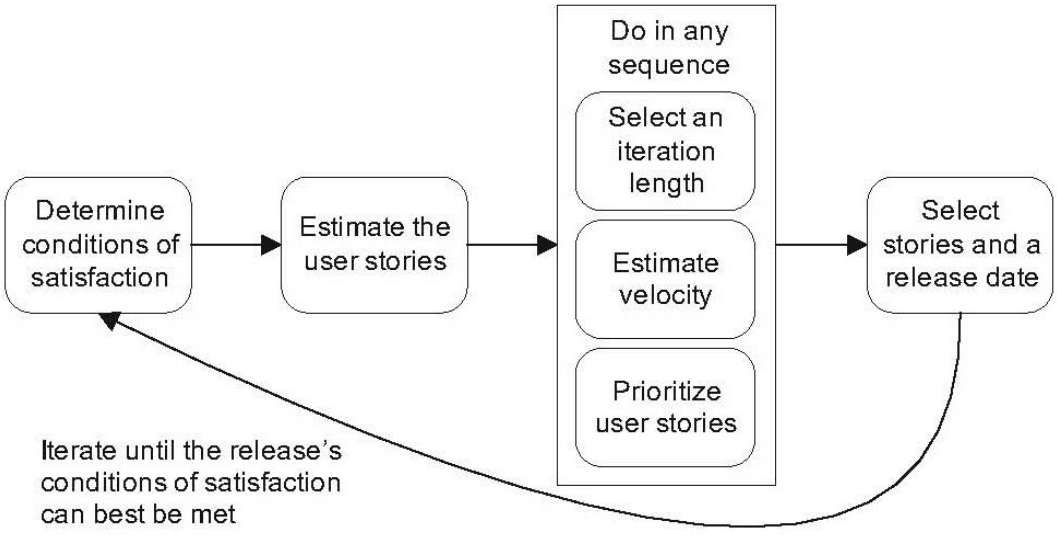

Figure 13.1 The steps to planning a release.

It is also essential to distinguish estimates from commitments. In many organisations a team that expresses an estimate is forced to commit to it.

Techniques

Estimates

Estimating provides substantial value as it helps determining cost, and it helps establish prioritisation and scheduling, Leffingwell (2010). While the size of an effort and the time needed to complete that effort are related additional factors may affect the duration. Agile features are usually estimated on a unit less predefined integer scale of story points (the absolute range of the scale does not matter) expressing relative difficulty rather than time. A story point combines aspects like knowledge, complexity, volume and uncertainty of a user story into a single value according to Leffingwell (2010). Cohn (2005) mentions a single estimate for each user story in ideal days (without any interruptions or other tasks) as an alternative to story points. In order to understand how story points relate to time the team's rate of progress "velocity" is taking into consideration. Velocity is calculated by summing the number of story points assigned to each user story that the team completed during the iteration. This way progress is measured at the team, rather than the individual level. The predicted amount of time that implementing a feature will take is simply a function of its size in story points and the team's rate of progress reflected by its velocity. The estimation process is depicted in Figure 4: Estimating the duration of a project from it's size, Cohen (2005, p.38).

Beck and Fowler (2000) recommend starting to estimate as soon as user stories get written to get some feedback as to the right level of detail for estimation.

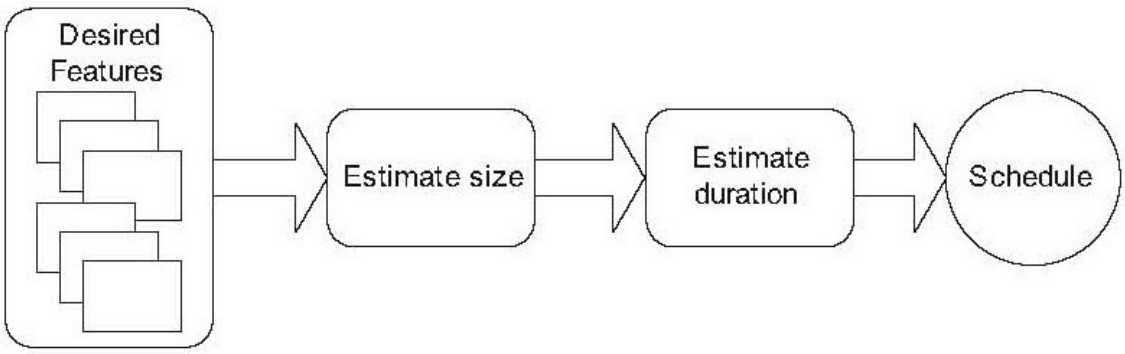

Figure 4: Estimating the duration of a project from it's size, Cohen (2005, p.38)

Cohn (2005 p. 59) suggest that estimating user stories should be a collaborative activity of the development team as he quotes evidence that the best estimates are given by those who will do the work. In an agile team it is not known in advance who in the team will be assigned so he concludes everybody should participate. In fact this idea is mentioned as the eleventh principle behind the Agile Manifesto by Beck and others (2001): “The best architectures, requirements, and designs emerge from self organising teams”. Regardless of the downsides to the approach of manager-driven planning it is still common, even on agile projects, according to Ambler and Lines (2012).

Common techniques for estimating the relative difficulty of user stories are expert opinion, analogy, and disaggregation. When estimating by analogy, the estimator compares a story to similar stories. An expert relies on experience and intuition to provides an estimate. Disaggregation refers to splitting a story or feature into smaller, easier-to-estimate pieces. The literature also discuses a method for combining all these techniques to a planning poker session, Cohn (2005). Each estimator is given a deck of cards with a point estimate shown on each. A feature is discussed and each player selects the card that represents his or her estimate. All cards are the shown and the estimates are discussed. The process repeated until agreement on the estimate is reached.

Prioritisation: Planning for customer value

A project rarely has enough time and resources to implement every desirable feature. Consequently, the most important features should be developed first to ensures essential features are not missing in case the project runs out of time. Beck and Fowler (2000) suggest using Cost, Quality, Time and Scope as prioritising criteria. They remark that the effect of changing one of these parameter can be both delayed and non-linear. For example, it is not possible to simply double the cost, hold everything else constant and halve the time. Cohen (2005) adds risk removed from the project and creation of knowledge about the project to the list of prioritising criteria. These factors are combined by thinking first of the value and cost of the theme (see Figure 5: Prioritising features by value and risk, Cohn (2005, p.85)).

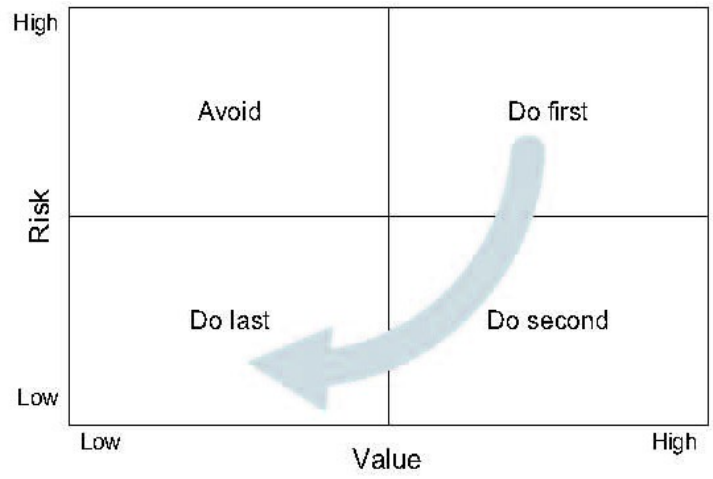

Figure 9.3 Combining risk and value in prioritizing features.

Figure 5: Prioritising features by value and risk, Cohn (2005, p.85)

Re-estimating and Re-planning

For a plan to be useful it must be accurate, but it is acceptable that early plans will be more imprecise. Periodic re-planning allows re-synchronisation to limit variance and misalignment to a single planning interval, Leffingwell (2010).

Re-estimating

Cohen (2005) cautions against re-estimate user stories and plans solely because development progress is slower than expected. He instead recommends a re-estimate of relative story points only when the team finishes only a portion of a story during an iteration or their opinion of the relative size of one or more stories has changed. The author recommends using re-estimating as a learning experience for estimating sizes of future user stories.

Re-planning

Re-planning serves to progressively remove plan imprecision. At the start of each new iteration a plan for only that iteration is created. Beck and Fowler (2000) suggest also to re-plan a release when a user story is added to the project or the priorities of user stories change. In agile planning customers do not have to commit to a detailed specification of everything they want before development begins. Therefore, priorities and requirements can change whenever they like as long as the goals are clear and the customer is informed about the consequences. By only committing to one iteration's worth of user stories at a time the customer can make adjustments to decisions based on the latest information about the priority and cost of the stories.

Scheduling and Tracking progress

Scheduling: Backlogs and queues

A foundational concept in Scrum is that requirements are managed as a prioritised stack called a product backlog (Ambler and Lines 2012). The contents of this backlog reflects evolving requirements, with the customers or product owner responsible for prioritising work on the backlog. The team takes just takes enough work to fit into the current iteration off the top of the stack at the start of each iteration as part of the iteration planning activity.

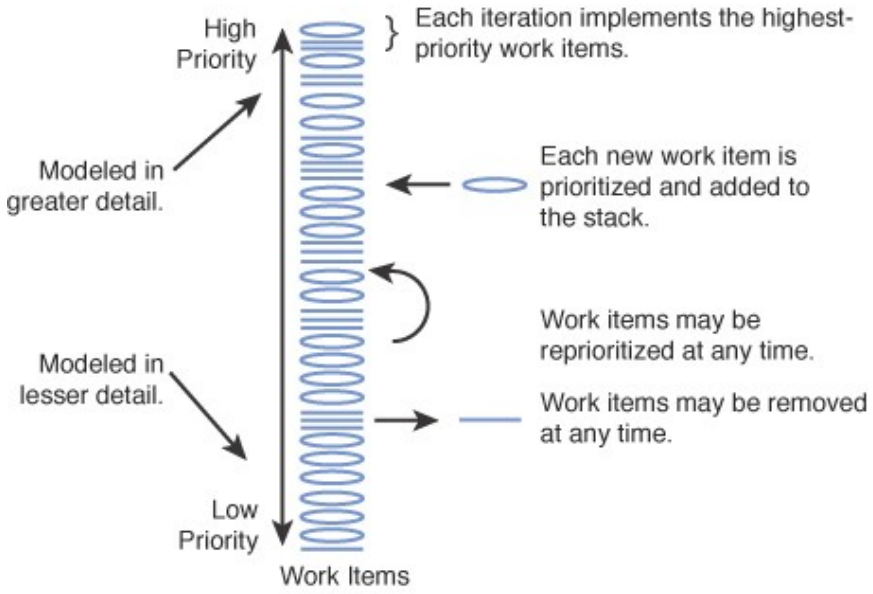

Figure 6: Work item stack, Gruver, Young and Fulghum (2012, ch. 8*)*

An extension of this concept is a work item stack (Figure 6: Work item stack, Gruver, Young and Fulghum (2012, ch. 8)). The Scrum's product backlog is extended to include all types of work items like requirements, defects, technical items and team collaboration requests (Ambler and Lines 2012). Work items are prioritised based on a variety of criteria. In order for backlogs to not interfere with the agility of the process Leffingwell (2010) recommends to keep them short, lightweight, and negotiable and to ensure to never overinvest in backlogged stories that may never be implemented.

Tracking

Once a schedule has been created it is critical to monitor development progress against it, communicate about progress, and then adjust the plan based on these observations. Ambler and Lines (2012) suggest using burn down charts to improve schedule estimates. A burn down chart shows the amount of work remaining on the vertical axis and time on the horizontal axis at release or iteration level (Figure 7: Burndown chart, (Ambler and Lines 2012, ch.16)). A release burn down chart shows the number of story points or ideal days remaining in the project as of the start of each iteration. A burn down chart may even show a burn up during an iteration. This means that even though the team completed some work they either realised the remaining work was underestimated or work has been added to the project.

The chart shows the amount of work the team has accepted for the current release and the projected time to complete the release past the current iteration. A team's burn down rate may vary over time because of inaccurate estimates or changes in estimates and in scope. While release-level burn down charts show in an aggregate basis where the team stand within the release and provides a sense of the probability of delivery of the release they do not provide any information as to which features may be delivered, Leffingwell (2010).

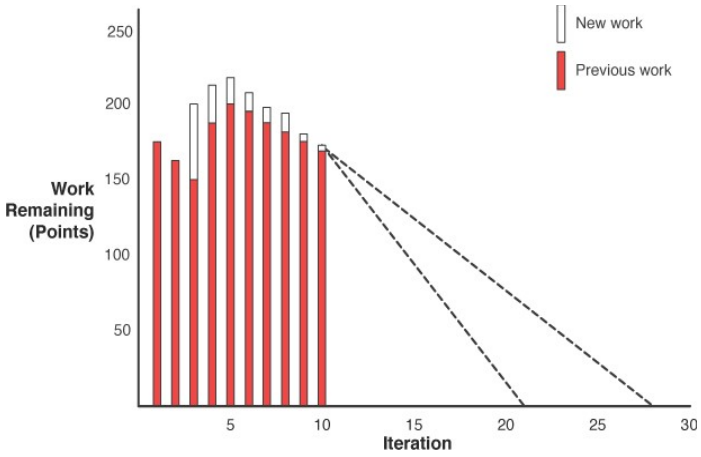

Figure 7: Burndown chart, (Ambler and Lines 2012, ch.16)

Advantages and potential drawbacks: Case study of a large scale distributed project (HP FutureSmart Firmware)

The problem

HP FutureSmart Firmware's R&D department develops embedded software to control Laser printer and copy machine hardware and solutions. According to Gruver, Young and Fulghum (2012) this firmware for enterprise-class printers and copiers can be as complex as PC operating systems. In 2008 the company faced major quality, technical and business challenges. Young (2012) describes development costs as out of control. Additionally, lengthy build integration and testing cycles made HP products lagging the competition. The development department was required to commit to a final feature list a year in advance and frequently failed to deliver and was unable to address late-breaking requests. The company spent 20% of their engineering resources on detailed planning to meet these early commitments. Often plans ended up being obsolete when requirements had changed months later.

HP FutureSmart Firmware was already a large and complex organisation with more than 400 developers in four different countries and a waste existing code base when adopting agile. Their experience was mainly about making agile work at a large scale which is commonly not regarded as a strength of such methodologies. While they made many changes to their process the following will focuses mainly on improvements of the planning aspects.

The solution: Taming the Planning Beast

Before learning about agile, HP FutureSmart Firmware introduced some early improvements but significant inefficiencies still existed. Gruver, Young and Fulghum state that *“The single most important thing we done to keep this from happening is to completely overhaul the way we do planning and scheduling. ...The only reason to do planning is to assure the business has what it needs to make financial and investment commitments. Every hour we spend planning a feature is an hour we don’t spend delivering it ”* (2012, p.67). While the authors acknowledge the importance of architectural and roadmap planning they challenge the amount of predictability the company needs. The authors of the study believe it lowers feature throughput and leads to false hope or unrealistic commitments.

A central element of in HP FutureSmart Firmware's planning process is now to manage all feature requests as a prioritised global input queue (called the unified 1-N List) similar to the “work item stack” discussed in Ambler and Lines (2012). The combined marketing teams get to the product owner and decide the feature order. This helps marketing teams to feel more in control and gives development a single point of contact across all features to help prioritise new features. System

engineers and marketing can collaborate closely on user stories in each area. Different business leaders come together regularly to review and approve the unified 1-N List.

Planning rhythm

To Gruver, Young and Fulghum (2012) the essence of agile is the planning rhythm. The company decided on a four-week sprint cycle, where objectives are set for the whole development organisation in terms of what will be accomplished for feature commitment, infrastructure improvements or customer-ready releases. HP FutureSmart Firmware aims for a user story granularity between two and four weeks of an engineer's time. Estimates are apparently made in ideal man days.
The authors feel rather long iterations keep the overhead for re-planning and reporting down. The company uses the velocity from previous iterations to project how far they can get in the "Input Queue"(1-N list) but avoid committing to anything low on the priority list. In their process, fixes for change requests or bugs are integrated into the system in the same way as new features.

HP FutureSmart Firmware found the daily updates to each task estimate in burn down charts time consuming and also consider burn down charts optimised for predictability instead of throughput. Instead, they developed alternative methods for tracking sprint progress by continuously tracking the status of requirements, change requests, and testing. Many of their development teams do daily standups but they do not involve the daily re-estimating or activities that takes away from feature throughput. They use sprint objectives and the sprint kick-off to look at the team's velocity for tracking and planning.

The company also found that asking teams to demonstrate results in front of management after a sprint finishes was a good way to build excitement and focus.

Prioritise everything

A global prioritisation scheme allows priorities to auto-steer the work within the organisation and establish realistic expectations. The company-wide prioritisation of high-level sprint objectives, of features and change requests helps resolve conflicts at the right level without a need for many people to collaborate or meet. All sprint objectives are ranked and published. Engineers and managers are expected to use them for daily trade-offs decisions. HP FutureSmart Firmware sacrifices lower-ranked objectives to higher-ranked objectives any time it is needed.

If a request for a new feature comes in the company make sure it is aligned with their product strategy and then prioritise it on the 1-N list where the business wants it in relation to the current feature backlog. A request that represents a critical customer commitment typically goes right to the top of the list, a system engineer starts defining it immediately, and it typically gets delivered in the next quarterly release.
The prioritisation scheme must not only include planned features from the but also any unplanned change requests and bugs. The company prioritise these unplanned request based on whether someone is blocked by them and how critical they are from a customer perspective. Typically, HP FutureSmart Firmware needs to respond to customer requests within two weeks. In the past this meant a cost/benefit analysis including an estimation effort. But now with a single firmware code base supporting several product lines the company no longer needs to do this. For change requests no planning or estimating is done at all, they are simply assigned a high priority in the queue if needed. The company now can frequently handle change requests and fully test them within a day. Most are fixed within a week after starting work on them.

Planning and Scheduling

When HP FutureSmart Firmware realised they were spending too much engineering resources on planning and still failed to be responsive to incoming feature requests they made changes to the responsibilities in the planning process and introduced the concept of Just-in-Time User Story definition.
Unlike the team based agile planning approach previously described requirement analysis in the company at is now done by only 2% of engineering staff, specialised “system engineers”. The system engineers deliver a clean feature definition a month or two ahead of when a feature is targeted to be delivered. They take new requested features from “new” to “investigated,” where they are well defined and ready for hand off to the technical team. This allows HP FutureSmart Firmware to evaluate incoming requests very quickly and keep the requesters engaged without disrupting development.
Dedicated marketing leads take the role of the Product Owner and immediately prioritise feature requests in the input queue as they come in so the company always know what features are most important. Architects then do early risk assessment of features and system engineers work closely with them to assure technology road maps are aligned with new requests.
Feature lead function as a technical lead for the whole feature, reporting to the project manager who acts as the end-to-end lead among all collaborating teams. The system engineer defines the “what,” the technical lead defines the “how,” and the requirement owner (typically the project manager of the team most involved in delivering the feature) with the technical lead coordinates integration, delivery, and qualification.
A feature kick-off meeting is held before the development starts and includes the requester, all engineers, test leads, and the feature technical leads to review and clarify the details of the feature.
One of the success factors was establishing the close collaboration of everybody involved in prioritising and defining features organised by the same end-to-end solution pillar areas. When new requests come in people in these roles can now efficiently accomplish both quick response and clean hand off by working on requests in a prioritised order.

Scheduling request becomes as simple as looking at the Input Queue relative to the size of the sprint-to-sprint delivery and do a projection. New requests may come in, but will be placed where marketing wants them in the 1-N list and deliver accordingly. While throughput has varied considerably so far (40 to 50 user stories per sprint) it is expected to stabilise more now that major changes in architecture are finalised.
Incoming requests for high level planning need to be evaluated and the organisational capacity for feature delivery evaluated. Early development in HP FutureSmart Firmware is driven by so called “R&D Initiatives” that are being pushed by the business prior to specific requests. To address these needs HP FutureSmart Firmware create a spreadsheet showing the capacity of each major group in the organisation twice a year. For each imitative a very rough estimates of engineering months of effort for each team is given. Additionally, architects review all initiatives for technology limitations and constraints to avoid major surprises. The results provided an accurate enough picture for initial decision making.

Convincing and transforming the business

HP FutureSmart Firmware's R&D department collaborates closely with product marketing, technical marketing, R&D partners and business leaders. Management and Marketing were initially sceptical about agile planning as there is no commitment when a certain feature will be delivered.

The R&D department was able to convince them by pointing out several benefits. The company no longer takes away engineering resources to plan. In most cases, a feature transitions directly from Input Queue to Work in Process to Verified using just-in-time planning. Reorganising the planning activities freed 15% of engineering resources to work on feature requests. Additionally, the Marketing department has now more input in setting priorities in the 1-N feature request list. Another advantage is the ability to accommodate late feature request quickly which can be critical for large sales opportunities or to match a competitor's new feature. The company can now deliver these requests which would have previously taken them 12-18 months to address. All that is required is to put them at the top of the list, ahead of all the other “input queue” features.

The re-organising also created clear responsibilities and a formal process for feature priorities that avoids conflicts. A key to this was getting a solutions marketing group set up who treated new features and solutions as their own roadmap. Although the product teams are no longer in control of every feature, this worked well for high-level features such as security, digital sending, extensibility and third-party solutions.

Results

The authors attribute the company's successful transformation to choosing the right model and tools, an investment in key roles in marketing and system engineering and establishing realistic expectations and relationships with all parts of the business. The overall financial results of HP FutureSmart Firmware's change in development methodology have been impressive. The total spending went down with a dramatic increase in the number of products developed and supported. By 2012 development costs had been reduced by 40% and the number of programs under development increased by 140% (Figure 8: Development of cost driven improvements for HP FutureSmart Firmware 2008 to 2011 (Young, 2012, Slide 11)). Also, the company can now focus more resources on innovation over maintenance and reduce the share of planing costs alone has decreased from 20% to 5%, Gruver, Young and Fulghum (2012). Beyond the financial success the change in process has improved internal collaboration and HP FutureSmart Firmware is now more responsive to new and evolving developments and customer requests.

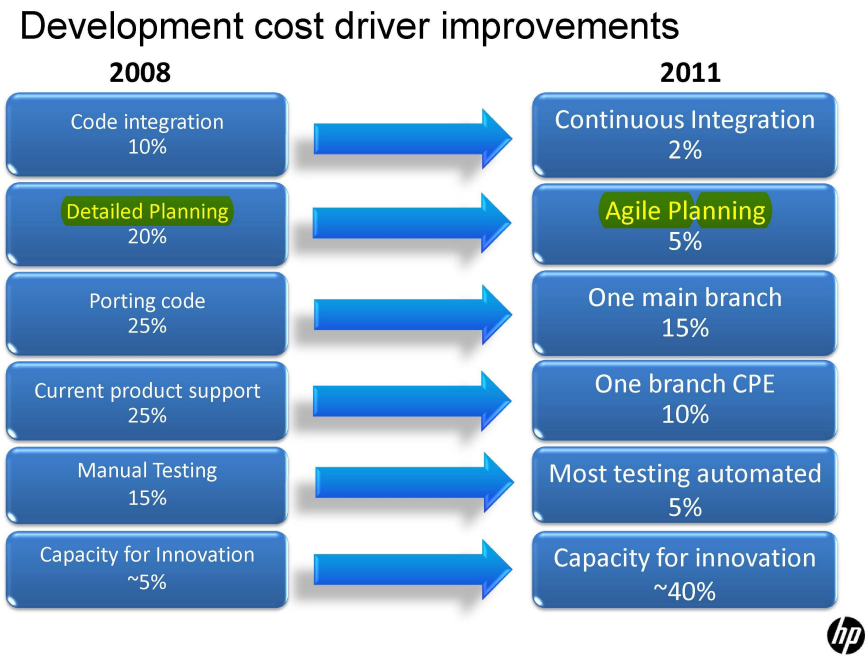

Figure 8: Development of cost driven improvements for HP FutureSmart Firmware 2008 to 2011 *(Young, 2012, Slide 11)*

While the company often follows “pure” agile practices discussed so far some of their processes have modifications tuned to fit their business objectives. The authors conclude that most of the remaining problems in their work flow may have to do with scaling the agile process beyond their 400 people and 12 products in development. They also discuss plans to speed up adding new products to their portfolio and synchronise agile methods across their many R&D and qualification labs and beyond the R&D operations.

Conclusion

The purpose of agile planning is to iteratively discover a solution for the overall product development questions. There are convincing arguments that an agile approach is more likely to succeed in finding a good solution by acknowledging uncertainty and planning for it from the start. Plans are made at different levels and replanning occurs frequently. Plans are based on features rather than tasks and size is estimated first and then duration is derived from the size estimate and progress is measured at the team rather than the individual level.

The high rate of unsuccessful software projects will probably make more companies consider alternative methodologies. Krigsman (2011) puts the project failure rate a 37% quoting a study by management consulting company PM Solutions. As there is no data available to directly compare project success of traditional and agile projects let's consider which common problems agile has potential to help with. The study identified problems with requirements definition, lack of resources, unrealistic schedules and insufficient planning and poor estimates as well as unidentified and not managed risks as common points of project failure.

While probably no development methodology can overcome a serious lack of resources and time agile is expected to help with managing changes in plans and requirements during the project process. It is hard to answer in general if risk management would be helped by agile planning. Therefore, using agile project management will not magically improve the outcome of any kind of project. It is most likely to be suitable for software projects that deal with high uncertainty in planning and requirement definition. In my opinion, this made agile work for HP FutureSmart Firmware despite there being some hints in the study of the company culture initially being more distrustful than cooperative.

Along this lines I expect companies who develop projects for quickly evolving markets and technologies to adopt agile methodologies widely in the near future. Combining agile development with some elements of more traditional approaches, as we have seen in the HP FutureSmart Firmware case study, seems to be a common practise in companies with a more traditional background. Ambler and Lines (2012) state that in their experience most agile projects make certain compromises that are not classically “agile” to get the job done. The authors suggest being upfront with this more pragmatic approach to using agile in their specific situation. As agile requires high commitments from team members and stakeholders I also think there needs to be a fit with the company's communication and management culture. Agile is a fundamentally cooperative methodology that may conflict with common roles and expectations of a very hierarchical organisation. Introducing agile in small steps may work better for an organisation if using the agile methodology involves major changes in culture for employees and managers at the same time.

In my personal opinion we will see more companies adopting and experimenting with these hybrid approaches combining agile with other approaches. These may make more sense for most companies rather than being dogmatic about purity of a certain methodology.

Reference list

Ambler, S. and Lines, M., 2012, Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise, IBM Press

Beck, K. and others, 2001, Manifesto for Agile Software Development , Available at <manifesto/principles.html>, [Accessed 16 March 2013]

Beck, K. and Fowler, M., 2000, *Planning Extreme Programming,* Addison-Wesley

Cohn M., 2005, Agile Estimating and Planning - A Practical Approach to Large-Scale Agile, Prentice Hall

Gruver, G., Young, M. and Fulghum, P., 2012, Development: How HP Transformed LaserJet FutureSmart Firmware (Agile Software Development Series), Addison-Wesley Professional

Krigsman M., 2011, Why 37 percent of projects fail Summary: New research identifies five important reasons that projects fail, Available at <http://www.zdnet.com/blog/projectfailures/cio-analysis-why-37-percent-of-projects-fail/12565> [Accessed 22 March 2013]

Leffingwell, D., 2010, Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise, Addison-Wesley Professional

Wikipedia, 2013, Project management, Available at <http://en.wikipedia.org/wiki/Project_management#The_traditional_approach> [Accessed 16 March 2013]

Young, M., 2012, A Practical Approach to Large-Scale Agile Development, Presentation for Agile Leadership Conference – Houston, Available at <www.agileleadershipnetwork.org/wp-content/uploads/2012/12/Young-LargeScaleAgielDevelopment-2012-01-20.pdf> [Accessed 16 March 2013]

Figures

Figure 1: Project Management phases in traditional project planning methods, Wikipedia 2013.....2
Figure 2: The agile planning onion, Cohn (2005, p. 28).....3
Figure 3: Iterative release planning, (Cohen 2005, p.133).....4
Figure 4: Estimating the duration of a project from it's size, Cohen (2005, p.38).....5
Figure 5: Prioritising features by value and risk, Cohn (2005, p.85).....6
Figure 6: Work item stack, Gruver, Young and Fulghum (2012, ch. 8) 7
Figure 7: Burndown chart, (Ambler and Lines 2012, ch.16).....8
Figure 8: Development of cost driven improvements for HP FutureSmart Firmware 2008 to 2011 (Young, 2012, Slide 11) 11