NETHERLANDS CHAPTER

PMI Netherlands Agile Local Interest Workgroup

White paper
Part 3 How to run an Agile project?
What tools and techniques are available?

Contents

Introduction

When answering Q2 we have advocated considering Agile project management as a separate (PM) project management method, to be used in project environments that are characterized by uncertainty. This is caused by the novelty of the domain of the project to the organization and/or new technologies that are being used (technology is unknown).

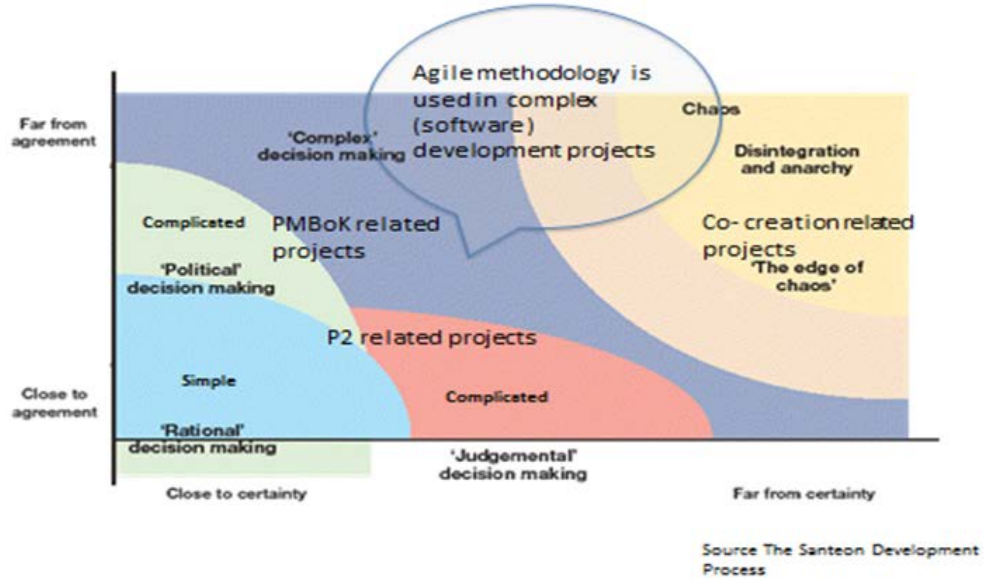

Figure 1 Positioning of PM methods, adapted model from R.D. Stacey

In this white paper we try to find an answer for Q 3: "How to run an Agile project; what tools and techniques are available?". When finding the answers to several sub questions we used the course book for passing the PMI Agile Certified Practitioner Exam. This book is written by Mike Griffiths.

In chapter 1 we focussed on finding an answer how to organize an Agile project.
In chapter 2 we focussed on finding an answer how to structure the Agile project management process. What activities and (management) products can be defined?
In chapter 3 we focussed on finding an answer on what developments methods are generally applied in Agile projects.
In chapter 4 we focussed on finding an answer on what makes Agile PM specific in regard to other PM methods.
We will refer to the six key success elements from white paper part 2 which should be part of a project management system or method.

Agile framework

Agile is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self –organizing, cross functional teams. These methods are all part of a conceptual framework that promotes a time- boxed iterative approach throughout the development cycle of an Agile project delivery process. This framework is suited for projects that are confronted with a lot of changes from a high dynamic environment. What are most known methods?

Scrum method

Scrum is a framework, structured to support complex product development. Each component within this framework serves a specific purpose and is essential to Scrum's success and usage.

Scrum is founded on empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience and making decisions based on what is known. Scrum employs an iterative, incremental approach to optimize predictability and control risk. Three pillars uphold every implementation of empirical process control: transparency, inspection, and adaptation.

- Transparency giving visibility to those responsible for the outcome. An example of transparency would be creating a common definition of what "done" means, to ensure that all stakeholders are in agreement.
- Inspection involves timely checks on how well a project is processing towards its goals looking for problematic deviations or differences from the goals.
- Adaption involves adjusting a process to minimize further issues if an inspection shows a problem or undesirable trend. There are four planned opportunities for inspection and adaption within the scrum framework: sprint retrospective, daily scrum meeting, sprint review meeting, sprint planning meeting.

The Scrum framework consists of roles, events, artifacts, and collective definitions. Based on this framework the scrum team iteratively builds increments of the solution, involving the customer frequently to ensure they are creating the right product.

Figure 2 SCRUM process

Events

A sprint is a time boxed iteration of one month or less to build a potentially releasable product. During a sprint no changes are made that would affect the sprint goal, although the scope may be clarifies or renegotiated as new information comes available. The development team members are kept the same throughout the sprint. SCRUM identifies four types of events:

1. A sprint planning meeting is used to determine what will be delivered in that sprint and how the work will be achieved. The product owner presents the backlog items and the whole team discusses them to create a shared understanding. The development team forecasts what can be delivered based on estimates, projected capacity, and past performances. The team then determines how this functionality will be built and how the team will organize to deliver the sprint goal. Output of this event will be a sprint backlog and plan.
2. A daily scrum is a 15 minute time boxed daily meeting. During this meeting the development team synchronizes activities, communicates and raises issues. It is held at the same place and time and each development team member provides answers for the following three questions about the work he or she is doing during the sprint: what has been achieved since the last meeting, what will be done

before the next meeting what obstacles are in the way. The daily scrum meeting is used to assess progress toward the sprint goal. The scrum master makes sure these meetings happen and helps to remove any identified obstacles.

3. A sprint review is a meeting held at the end of the sprint to inspect the increment or evolving product, that was built and to change the backlog if necessary. The development team demonstrates the work that is done and answers any questions about the increment. The product owner decides what is done and what is not yet done. The product owner and the team discuss the remaining product backlog and determine what to do next.
4. A sprint retrospective which is held at the end of the sprint to reflect on the process and look for opportunities for improvement. The retrospective occurs after the sprint review and before the next sprint planning meeting. This timing allows the team to incorporate the product owner's feedback from the sprint review and also allows them to factor improvements identified during the retrospective into the next plan. The team focusses their evaluation on people, relationships, processes and tools.

Products/artifacts

SCRUM identifies three important artifacts:

1. A product backlog, which is the ordered list of everything that might be needed for the product. It serves as the single source for requirements. This backlog is dynamic and evolves as the product evolves. It contains prioritized features to be build, requirements, quality attributes (often referred to as nonfunctional requirements), enhancements and size. Higher ranked items are more detailed and therefore the estimates for these items are more precise. Low priority items may not get developed or they may be deferred in favor of higher priority work. Grooming the backlog is the process of adding more detail and order to the backlog and refining the estimates of the backlog items. This effort is done by the development team and the product owner.
2. A sprint backlog, which is the set of features from the product backlog that were selected for a specific sprint. The sprint backlog is accompanied by a plan of how to achieve the sprint goal so it serves as the development team's forecast for the functionality that will be part of the sprint. It is a highly visible view of work being undertaken and it may only be updated by the development team.
3. Collective definition of "Done". When a backlog item is described as "done" everyone must be in agreement about what done means. To remove any ambiguity the team should collectively create the definition of done for the items before they begin work on them.

Extreme programming / XP

While SCRUM focusses more on a project management level when prioritizing work and getting feedback, XP focuses on software development good practices. The core values of this methodology are: simplicity, communication, feedback, courage, and respect.

- Simplicity focuses on reducing complexity, extra features and waste. The team should keep the phrase "find the simplest thing that could possibly work" in mind and build that solution first. Communication focuses on making sure all the team members know what is expected of them and what other people are working on. The daily stand up meeting is a key communication element.
- Feedback focuses on getting impressions of suitability early in the execution. Failing fast can be useful especially if in doing so we get new information while we still have the time to improve the product.
- Courage is needed when making the work entirely visible and transparent to others.
  Respect is essential where people work together as a team and everyone is accountable for the success or failure of the work they perform.

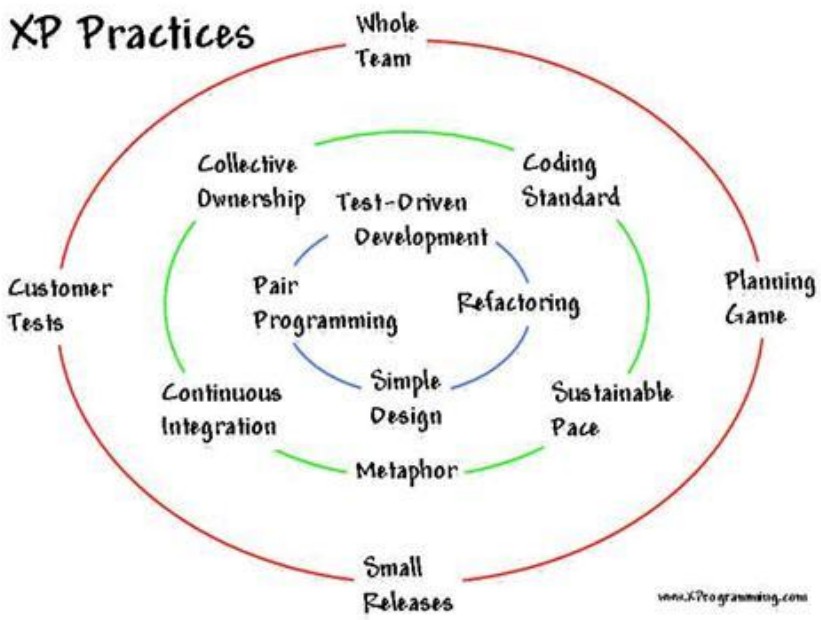

Figure 3 XP core practices

Release

Important inputs for release planning and iterations come from user stories (light weight requirements) and architectural spikes.
A release is a push of new functionality all the way to the user. An Agile project delivers one or more releases. During release planning the customer outlines the functionality required and the developers estimate how difficult the functionality will be to build.

Iterations

Within a release several short development cycles, iterations, take place. Iteration planning is done at the start of each iteration or every two weeks. In this planning session the customers/ product owner explains what functionality they would like to see in the next two weeks. The developers break this functionality into tasks and estimate the work. Based on these estimates and the amount of work accomplished in the previous iteration, the team commits to what work they aim to complete in the two weekly period.

Architectural spikes

These iterations are used to prove a technological approach. Spikes are periods of work undertaken to reduce risks. The spikes are blended into the release planning process.
Developers work in pairs (using benefits from the larger knowledge base of two people and early awareness of issues) to write code during the iterations. All software developed is subjected to rigorous and frequent testing; tests are written prior to developing the new code (test driven development). Then, upon approval by the on-site customer, the software is delivered as small releases. As part of defining the required functionality, the customer describes one or more tests to show that the software is working. The team then builds automated tests to prove to themselves and the customer that the software is working.

Feature driven development (FDD)

This is a simple to understand yet powerful approach to building products or solutions. The development team will first develop an overall model for the product, next create a feature list, plan by feature and then moves through design & build & test iterations to realize the features. The next best practices are used:

- Domains object modeling, as a result from exploring the business environment of the problem to be solved.

- Developing by feature which results from breaking down functions into two week or even shorter chunks.
- Individual class (code) ownership in order to have a single owner for consistency, performance and conceptual integrity (it differs from XP's collective code ownership aiming to spread knowledge to other team members).
- Small and dynamically formed feature teams that vet designs and allow multiple design options to be evaluated before a design is chosen (this helps to mitigate risks associated with individual ownership).
- Inspections to help ensure good quality design and code.
- Configuration management involving labeling codes, tracking changes and managing the source code.
- Regular builds to make sure the new code integrates with existing code (and allow easy creation of demo's).
- Visibility of results and tracking progress based on completed work (for instance by using one page summaries and parking lot diagrams).

Dynamic Systems development (DSDM)

This is one of the earlier Agile methods covering a complete project lifecycle from feasibility and business case to implementation.

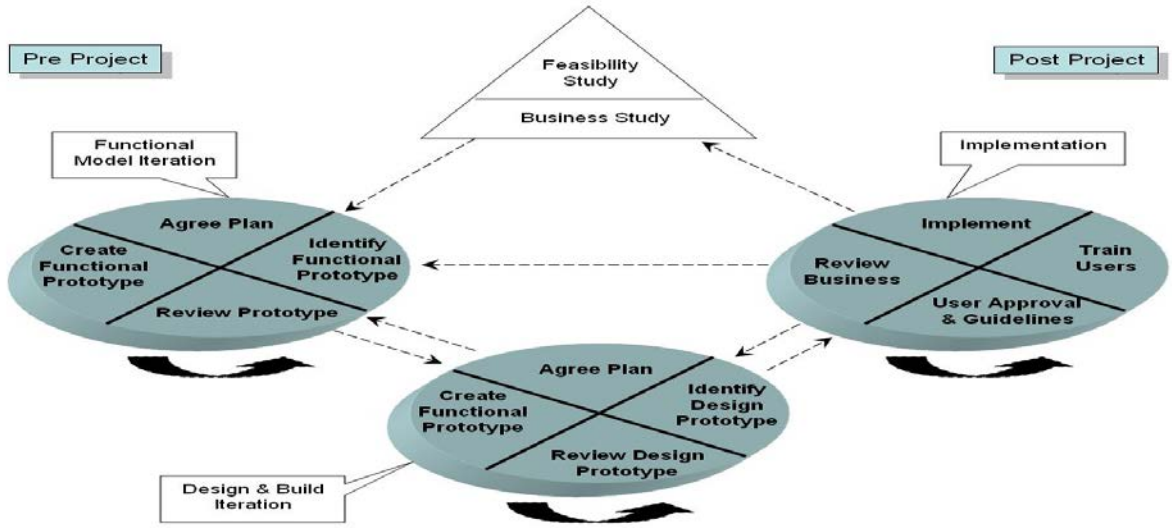

Figure 4 DSDM method

The method is based on eight principles: focus on business need, deliver on time, collaborate, never compromise quality, build incrementally from firm foundation, develop iteratively, communicate continuously and clearly, demonstrate control.

Crystal

This is a family of methodologies designed for projects ranging from those run by small teams developing low criticality systems (crystal clear) to those run by large teams building high criticality systems (crystal magenta). Besides this "tailoring", crystal methods are based on principles like:

- Frequent delivery and accepting of increments of a solution.
- Regularly checking for ways to improve.
- Team members are co- located (osmotic communication) to allow them to efficiently share information.
- Creating an environment where people can safely raise issues or questions.
- Focus on what to work on and have the time and peace of mind to work on it.
- Easy access to expert users.
- Efficient technical environment (automated testing, configuration management, frequent integration).

Lean software development

Strictly speaking lean software development is not an Agile method, but Lean and Agile values are closely aligned. Lean development focuses on seven core principles: empower the team, eliminate waste, deliver fast valuable software and iterating through designs, optimize the whole (system based thinking), build quality in (continually assure quality throughout the development process), defer decisions (balancing early planning with making decisions and committing to results as late as possible; use last minute information), amplify learning.

Agile project management process groups

We have advocated to consider Agile project management not only as a framework for project delivery, but also as a separate Project Management method covering the complete project life cycle. A (project-) management method or (project-) management system is an integration of three important components:

1. Guiding (project-)management principles; see our paper
2. (Project-)Management process groups
3. (Project-) management elements (in PRINCE2 PM Method called themes and in PMBoK method called knowledge areas).

Ad 1. Agile management principles

(Referring to chapter 5 in our paper in research Q 2), We have identified 11 guiding management principles: satisfy the customer through early and continuous delivery of valuable results, except continuous change from a highly dynamic environment, deliver ready to use results frequently, close cooperation between business people and developers throughout the project, build projects around motivated individuals, most effective way to communicate is face –to- face communication, ready to use results are the primary measure of progress, continuous attention to technical excellence and good design, simplicity, the best results emerge from self-organizing teams supported by servant leadership, continuous learning and improvement.

Ad 2. Agile process groups

Before going in detail on techniques and tools related to Agile project management elements let's define and explore the Agile project management (PM) process groups and related management products.
Referring to figure 6 in our paper on Q 2, we compared the Agile project management process groups with PMBoK project management process groups, that are more suited for less dynamic environments.

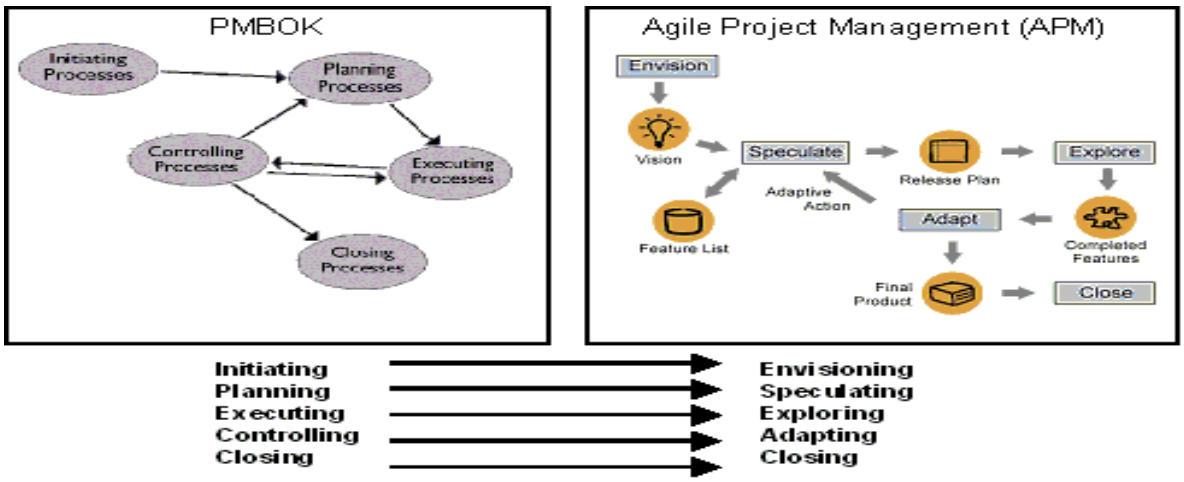

Figure 5 Comparing PM process groups

Let's have a closer look at these process groups.

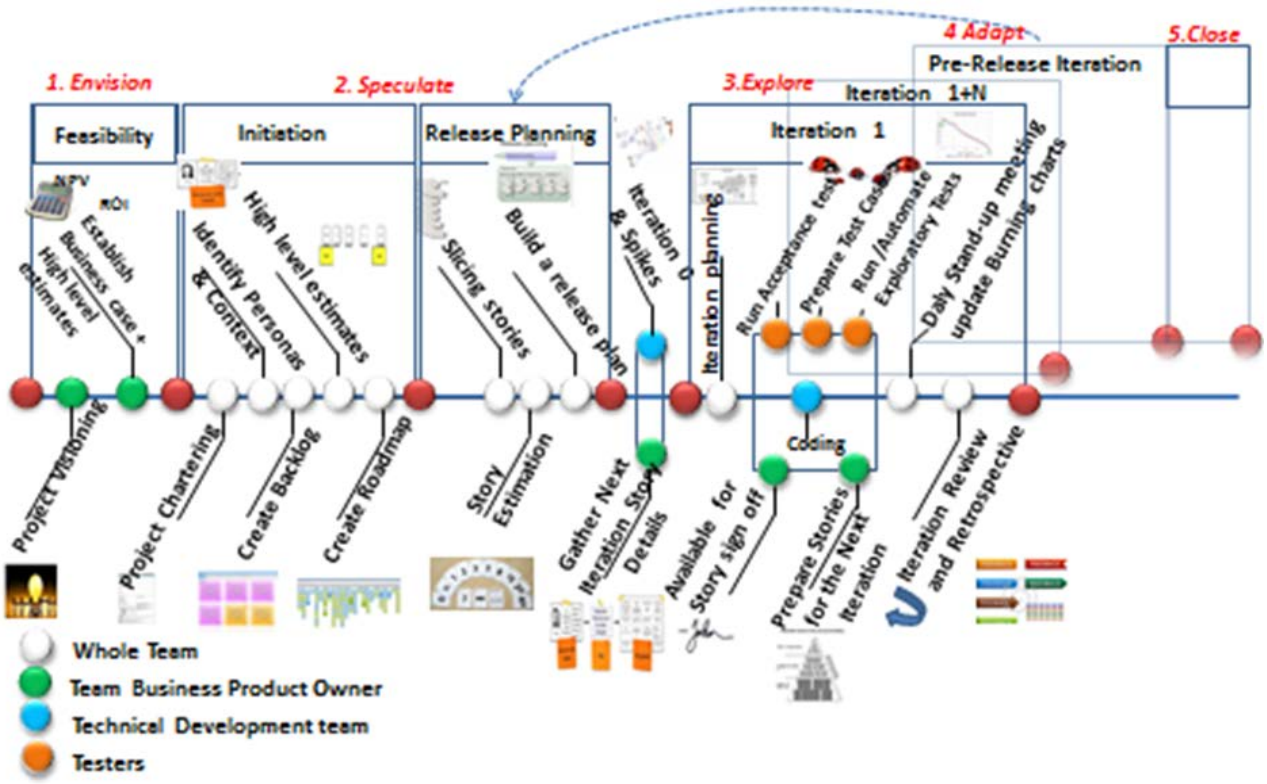

Figure 6 Agile project management process groups in more detail

Envision phase

The Envision PM process (or management phase) defines the beginning of a project for which the kick-off event might be the approval of a business idea research result. Both development and product team members should be involved in this envisioning process in order to identify what is to be done, what participants/stakeholders should be part of the project community and what project strategy (Agile project or not) matches best.

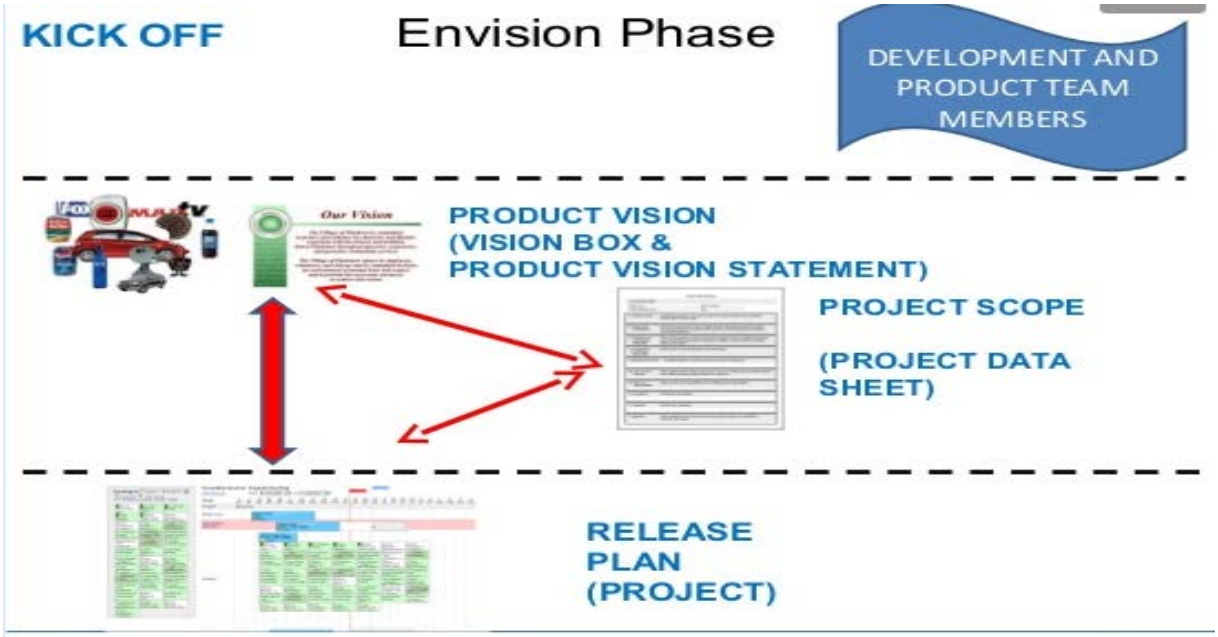

Figure 7 Envision phase

Product Vision statement

A product vision (defined by a product vision box elevator statement) galvanizes members of the product team into focusing their often disparate views of the product into concise and short textual form. It helps align the stakeholders behind a common mission, goals and success criteria.

Figure 8 Example product vision

Product Vision Box

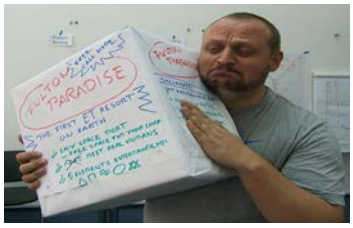

Figure 9 Product vision box

- Graphic
- Product selling points (Front)
- Detailed Feature Description (Back)
- Operational Requirements (Back)

Project Scope

Project Scope explains how a project will deliver on the product vision. Single-page summary of key business and quality objectives, product capabilities and project management information.
Simple, condensed format document that constantly reminds of the strategic aspects of the project.

Project charter

The project charter describes the project's goal, scope, product vision, involved stakeholders, outline business case, strategy. It provides authorization from the sponsor for the project to proceed. Agile charters acknowledge that scope may change and that initially some aspects of the project may be unknown. Therefore rather than trying to fully specify the scope, Agile charters characterize the goals envisioned for the project.

Project Data Sheet.

Project Data Sheet

Project Name: CRM Development					Project Manager: Braxton Quivera
Project Start Date: 3/1/2004					Product Manager: Roger Jones
Clients:					Executive Sponsor: Andrian Poledra
Marketing					Client Benefits:
Call Center					Better customer service
Sales					Reduce paperwork
Accounting					More accurate order processing
					Better customer account management
Project Objective Statement:					Performance Attributes:
The objective is to build a web-based CRM application that includes sales tracking, order management, sales management, and marketing.					Call Center volume of 3,500 calls per day
					Worldwide web access
					<1/2 day training required
The system needs to be operational by 6/30/05 and cost less than $2.5 million.					No severity 1 defects
Trade-Off Matrix:					
	Fixed	Flexible	Accept	Target	
Scope		x		12,500 FP	Product Architecture:
Schedule	x			+/– 6 weeks	Interface with ERP system
Resource			x	+/– $.5 M	Windows XP, SQL Server, .NET
Stability			x		
Project Delay Cost per Month: $50,000					
Exploration Factor: 8					
Business Subject/Activity:					
Sales Management					
Sales Analysis					
Prospecting					
Territory Management					
Marketing					
Lead Generation					
Lead Follow Up					Issues and Risks:
Advertisement Placement					Development costs are difficult to calculate.
Call Center Service					Sales staff reluctant to embrace new system.
Order Management					Requirements agreement among user groups may be difficult.
* See the Feature Breakdown Structure for details, including the individual features					
Major Project Milestones:					
Marketing except Call Center	9/30/04				
Call Center	1/15/05				
Sales Management	3/30/05				
Order Management	6/30/05				

Figure 10 Project data sheet

Speculate phase

Based on a series of workshop activities from product team and development team members the product vision is broken down into a draft feature list. It is prioritized based on business value and risk and ultimately form the product backlog. This backlog is input for the iterative development cycle in the explore phase. The product road map is a visual overview of a product's releases and its main components. A project compromises one or more releases. A release plan shows how the completion of valuable deliverables on the project is realized by a series of iterations.

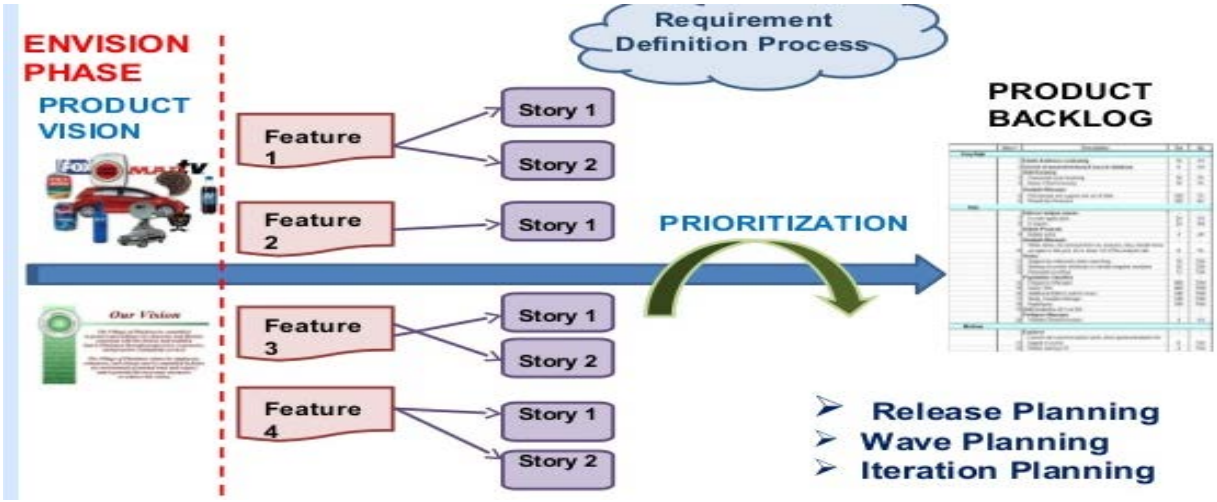

Figure 11 Speculate phase

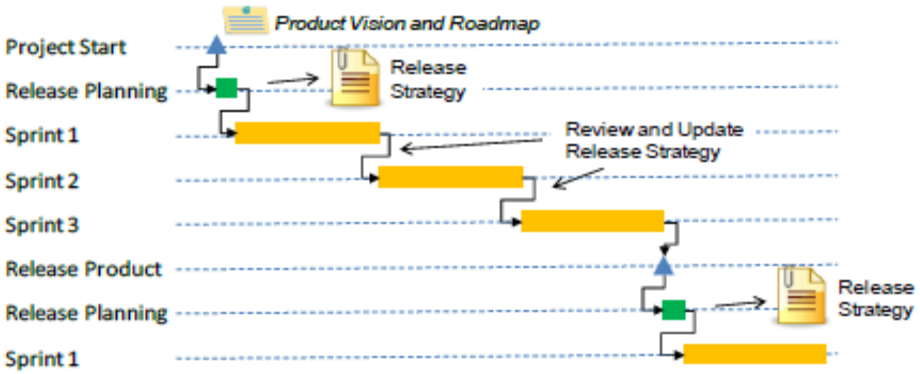

Figure 12 Release planning and iterations

Explore phase

Based on an iteration plan running, tested and accepted stories are delivered. Information radiators keep the product team and other stakeholders informed on progress. The transition from the Envision phase into the Speculate phase shows that release planning done in the Speculate cycle connects to iteration in the Explore cycle.

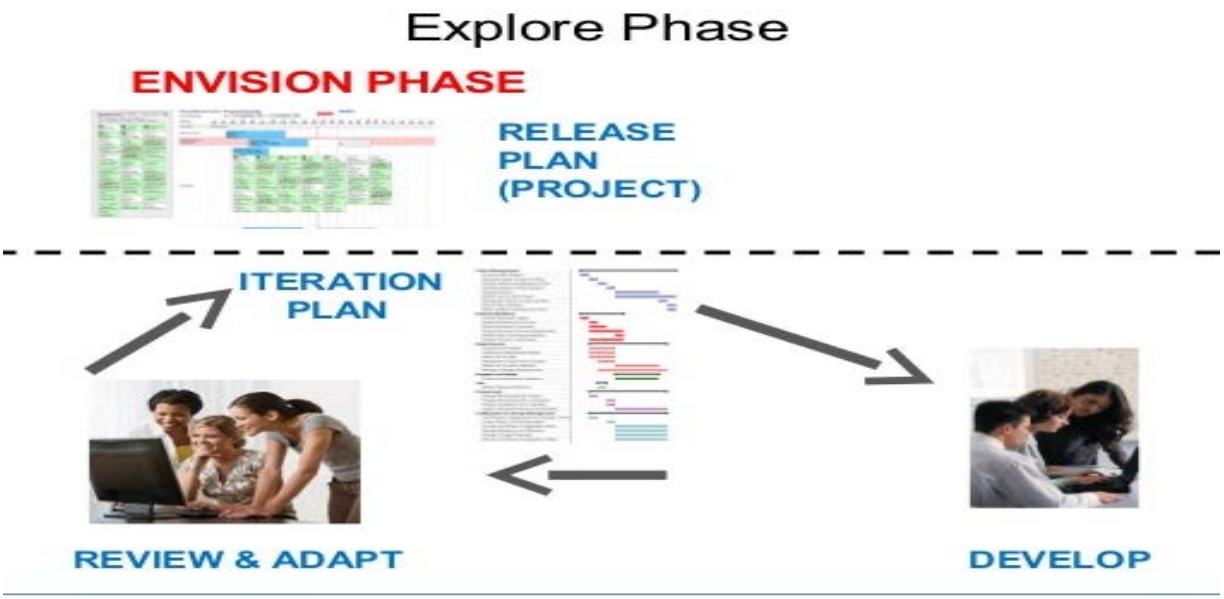

Figure 13 Explore phase

Major Feature	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6
	<Time line>	<Time line>	<Time line>	<Time line>	<Time line>	<Time line>
Authenti-cation	1. Login Screen					
	4. SSL Encryption					
Master	2. Product Master		9. Address Master			
	3. Rate Master					
Order Entry		5. Product Selection				
		7. Product Preview	8. Product Comparison			14. Order Track
				12. Product Review		
Checkout		6. Checkout		10. Shipping Choices	13. Coupons	
				11. PayPal Integration		

Figure 14 Example Iteration planning

Adapt phase

This PM process demonstrates and reviews the delivered results, evaluates what is “done” and what should be done next, evaluates the team’s performance and iteration process and adapts/ re-aligns the release plan as necessary.

Close phase

This PM process reports on remaining “technical debts”, the project justification, important lessons learned to pass along, celebration of project end.

Ad 3. Agile project management elements

Based on best practice research on project success factors, we have defined (see research Q2 chapter 3) six key management elements:

1. Clear defined project vision and business goals.
2. Managing expectations and getting commitment from stakeholders to invest in project realization.

3. Active user involvement.
4. High performing teams with motivated and competent staff.
5. Leadership based on formal roles (project management organization) and behavior.
6. Planning and control mechanisms (time, costs, quality, scope, risks)

Using Mike Griffith's course book for passing the PMI Agile Certified Practitioner Exam we identified several tools and techniques that can be applied to transform these elements into successful Agile project management elements.

Project Vision

Value driven delivery

A project vision presents the desirable situation or "goal" in the future, after implementing project results and related behavioural changes in the running business. This high level goal is the project executive's tool to generate passion and meaning for the project and hence to realize changes. The high level goal is related to the benefits expected from the business change. Benefits are an important element of the "business case" definition.
The added value of business changes is often based on economic criteria. In principle the economic value of every single project initiative can be "translated" in cash flow or money. (For instance cost savings, more turn over, higher turnover rate due to shorter operations lead time). The economic value of project initiatives can be assessed using techniques like Return On Investment (ROI % = average net cash flow/investment x 100) or Net Present Value (sum of discounted net cash flows during product lifecycle should be positive). The impact of risks with medium or high probability should be part of this economic assessment.
Besides the economic value other values could also be taken in account, for instance social, environmental, cultural, and aesthetic.

Characteristic for Agile project management is delivering as many highly prioritized value components or benefits as soon as possible. By delivering these high value elements early, the team demonstrates an understanding of the stakeholder's needs. Active involvement of stakeholders, for instance in retrospectives and planning meetings, is important to be connected with their values.

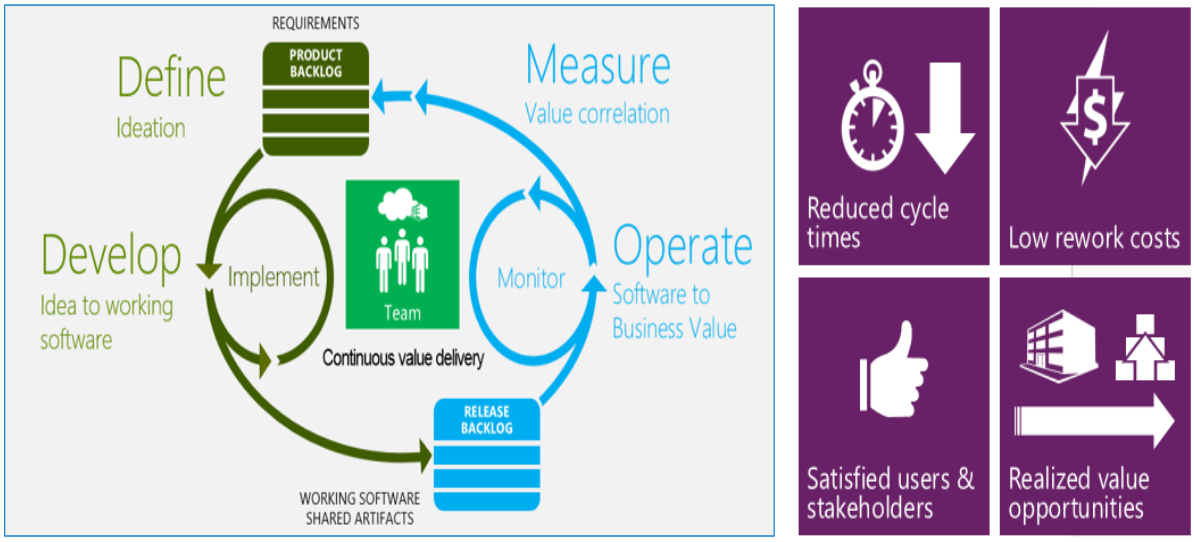

Figure 15 Value driven delivery

Project charter

Value driven delivery starts with a project charter. Compared to more traditional PM methods an Agile charter focusses more on how the project will be run rather than defining exactly what will be built. Agile charters answer questions like:

- what is this project about in terms of mission, goals and objectives,

- why is it being undertaken,
- when will it start and end,
- who will be engaged,
- where will it occur,
- how will it be undertaken.

Value stream mapping.

When creating value for the customer, the team should maximize value delivering activities and minimize "waste" or non-value delivering activities. Value stream mapping helps the team in their search for value. It follows next steps:

1. Identify the product or service that you are analysing.
2. Create a value stream map of the current situation, identifying processes and information flows.
3. Review the map to find delays, constraints, waste (for instance less productivity caused by multi-tasking, nice to have features, un-used documentation, unnecessary approvals, waiting for approvals); non value adding activities.
4. Create a new value stream map of the desired future state to remove or reduce no value adding activities.
5. Develop a roadmap for creating the optimized state.
6. Plan to revisit the process in the future to continually tune and optimize it.

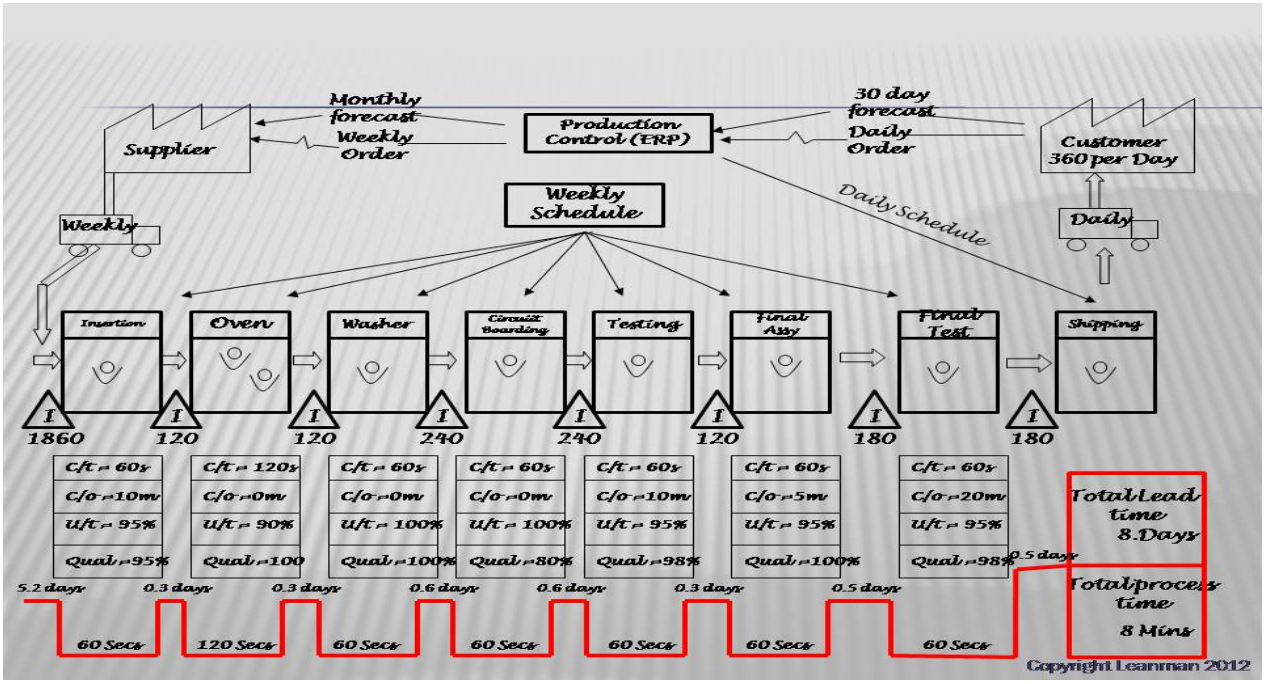

Figure 16 Example value stream mapping

Customer valued prioritization

When creating value for customers it is important to focus on items that yield the highest value to the customer as soon as possible. Prioritization is essential for the team to be able to adjust the scope to meet budget or timeline objectives, while still retaining the set of functionality that is complete enough to be useful to the user or market (MMF; minimally marketable feature).

In SCRUM this priority list is called product backlog, in FDD feature list and in DSDM prioritized requirements list. Late breaking changes can be accepted, at the expense of lower priority features. A well-known prioritization (DSDM related) tool is the MoSCoW scheme (must have, should have, could have would like to have, but not this time). Other prioritization tools can be 100 point method or Kano's customer satisfaction categories exciters, satisfiers, dis-satisfiers and indifferent features.

Work in progress

Value driven delivery aims to optimize the throughput of work, not to optimize resource utilization. Limiting the amount of work in progress (WIP), this represents money spend with no return, is important in order to prevent waste caused by “waiting time” due to multi-tasking. By delivering
the “plain vanilla” version of a product or service while working on the more complex elements, the team has another opportunity to start realizing the benefits of the product and get an early return on investment.

Risk adjusted backlog

Because Agile projects are both business value- and risk- driven, prioritization focusses on functional features as well as risk response actions. The importance of the response action is based on the expected monetary value (risk probability as a percentage times risk impact in money). When this priority setting of risk response actions is combined with the prioritized feature list a risk adjusted backlog is created. If a money value is attributed to the features as well, the team can discuss with the business sponsors on the importance of risk management actions related to the added value of features to be realized.

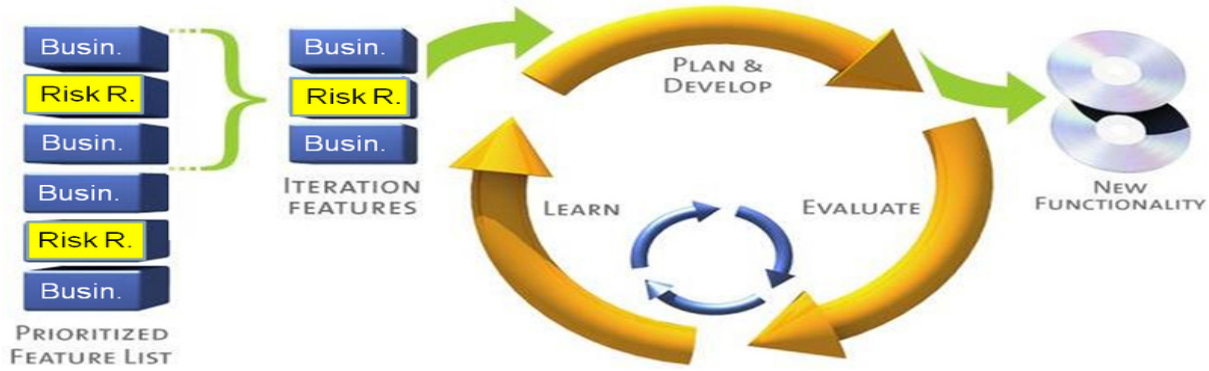

Figure 17 Risk adjusted backlog

Stakeholder commitment

Stakeholders are "all Individuals or groups who have an interest or some aspects of rights or ownership in the project, can contribute in the form of resources, knowledge, insight or support, or can impact or be impacted by the project”.

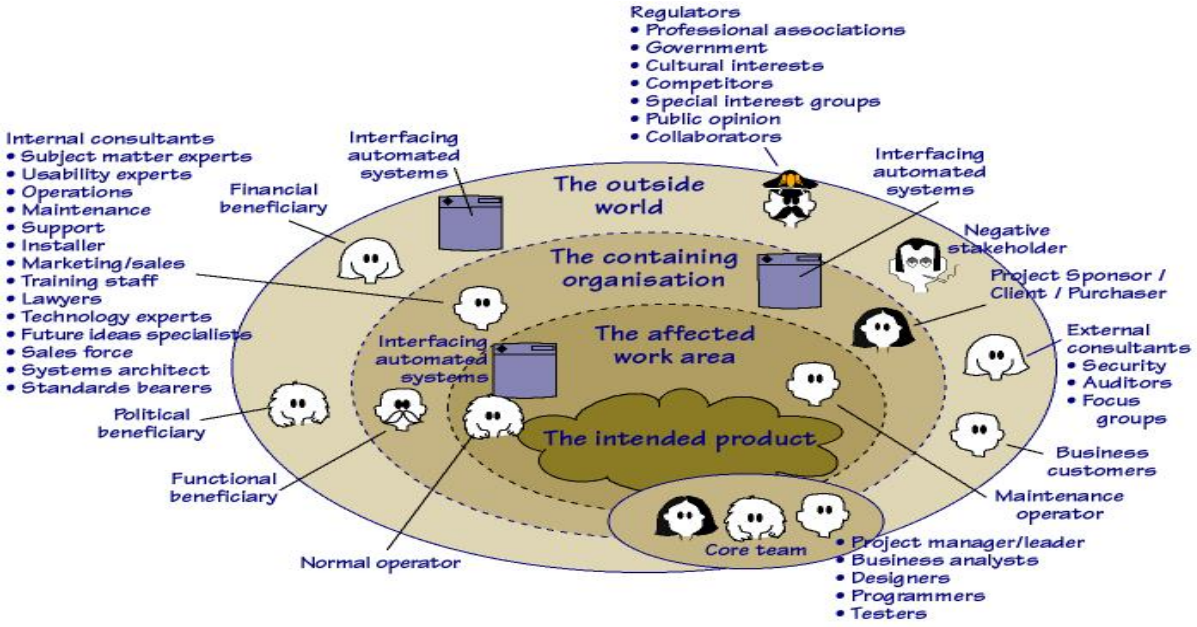

Figure 18 Example stakeholder map

The dynamic environment, that is characteristic for Agile projects, is caused by the interests, related and conflicting values and impact of these stakeholders. Instead of categorizing stakeholders as more or less risk factor for the project, Agile project management acknowledges differences in values and beliefs. An Agile project facilitates stakeholders' interaction processes in their search to get the best/the most value out of the project. The frequent, time boxed iteration loops offer a perfect opportunity for aligning team's efforts and stakeholders prioritized value realization.

Face to face communication with the stakeholders is essential in this interaction process. Below a diagram is shown that indicates the effectiveness of face to face communication compared to other communication channels. This effectiveness is based on factors like immediate feedback opportunities and awareness of nonverbal communication. Soft skills like conflict handling and win – win based negotiation are also relevant in communicating with stakeholders.

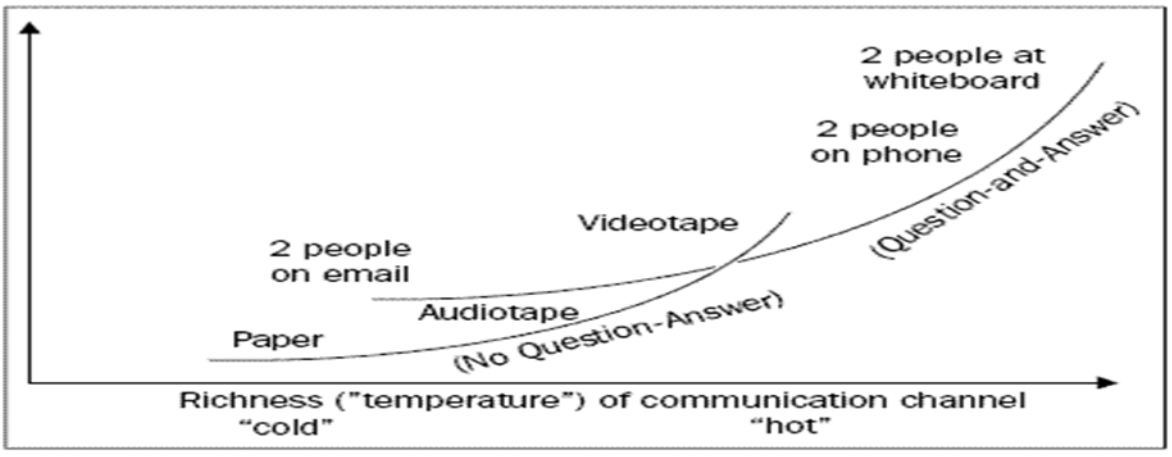

Figure 19 Effectiveness of communication channels

The interaction process starts in the envisioning phase of the project. Techniques/tools like product vision statement, product vision box, project charter, project data sheet were already mentioned. Next additional techniques are presented to increase business stakeholder's insight in the product and its realization.

Elevator statement

This statement is a short description of the goal, benefits and discriminators. When starting a project stakeholders are invited to join a session for developing this statement. It helps them to explore the value of a new product idea. The results can be used when defining the business case and the project charter.

For:	Target customer Example: Project manager
Who:	Need (opportunity or problem) Example: Who want to become agile project leaders
The:	Product/service name Example: Leading Agile projects training
Is a:	Product category Example: Three days course
That:	Key benefit/ reason to buy Example: Make project managers understand Agile principles ad processes and develop servant ad adaptive leadership skills
Unlike:	Primary competitive alternatives Example: Agile courses from generic training organizations
We	Primary differentiation Example: Use instructors with Agile project management experience to ensure they can answer all how to do questions ad our supplementary mayerials including tools, case studies and practical instructions

Figure 20 Example elevator statement

Personas

These are quick guides or reminders of the key stakeholders on the project and their interests. The tool helps team members empathize with stakeholders related to results from the project. In software projects personas are created for the different types of people who will use a system being built. Personas may be based on profiles of real people or compositions of multiple users. Personas are no replacement for requirements but instead augment them. In order to be effective personas should:

- provide an archetypal description of users,
- be grounded in reality,
- be goal oriented,
- be specific and relevant,
- be tangible and actionable,
- generate focus.

Figure 21 Example personas

User stories

These are bite- sized, understandable chunks of business functionality. Teams commonly rely on user stories and a backlog of these stories to help align team priorities with the needs of the business.
User stories are often written in the following format: As a (role) I want (functionality) so that (business benefits)…. For example: As a movies online customer, I want to search movies by actor, so that I can more easily find movies I would like to rent.

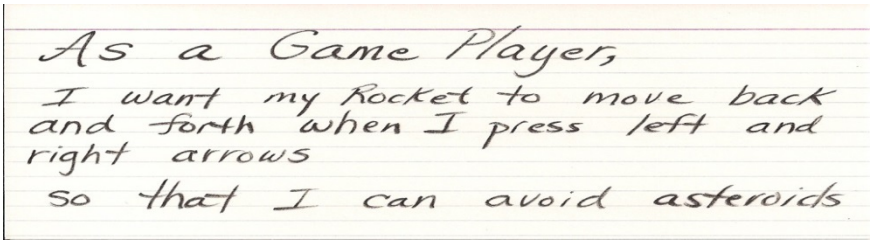

Figure 22 Example user story

This template forces the project to identify the user (who is asking this) and the business benefit (why are we doing this) for every required piece of functionality. In order to be effective a user story should be:

- **Independent;** ideally we want to reprioritize and develop user stories in any order,
- **Negotiable;** we want user stories be discussable with business representatives and enable trade-offs based on cost and function,
- **Valuable;** we want user stories with clearly understood business benefits, otherwise it will be difficult to prioritize since backlogs are usually ranked on business value,
- **Estimable;** we want to be able to estimate the effort needed to realize the functionality in order to prioritize the user story based on its cost /benefit trade -off,
- **Small;** we want small (half a day up to ten days) user stories so they can be estimated more easily and they can be completed within an iteration,
- **Testable;** we want user stories to be testable in order to come to a for formal acceptance and to track progress based on accepted products.

Story map and product roadmap

A prioritized user story backlog helps to understand what to do next, but is a difficult tool for understanding what the whole system is intended to do. A user story map arranges user stories into a useful model to help understand the functionality of the system, identify holes and omissions in the backlog, and effectively plan holistic releases that delivery value to users and business with each release.
Once the features are placed on the map according to their importance and sequence, the customer's priorities are balanced with the team's capacity to deliver and the releases that will make up the product roadmap are identified. The product road map is a visual overview of a product's releases and its main components.

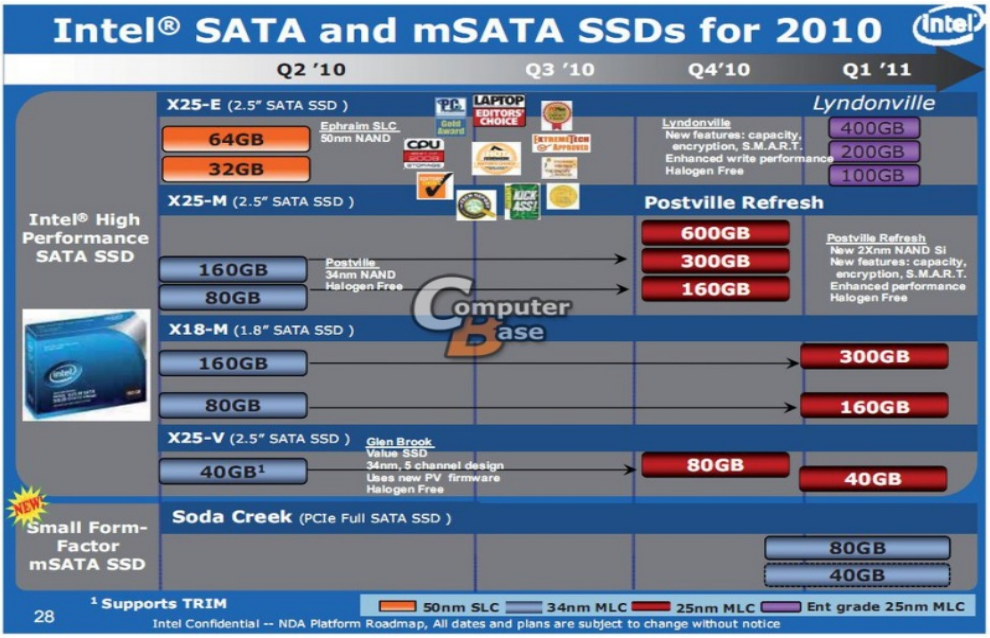

Figure 23 Example of a product road map

Information radiators on product realization

A. Cockburn[1] used this term "radiator" to emphasize the contrast to the practice of locking the project information away in an "information refrigerator". These information radiators, also sometimes referred to as "visual controls", quickly inform stakeholders about product's realization status. These visual controls are usually displayed in "high traffic" areas to maximize exposure. The sort of data that might be displayed on information radiators includes:

- The features delivered to date versus the features remaining to be delivered
- Who is working on what
- The features selected for the current iteration
- Velocity and defects metrics
- Retrospective findings
- Risk registers

Kanban boards

These boards are primary thought of as planning and monitoring tools, but can also be used to visualize and communicate on value delivery.

[1] A. Cockburn Agile software development: the cooperative game, Adisson Wesley 2007

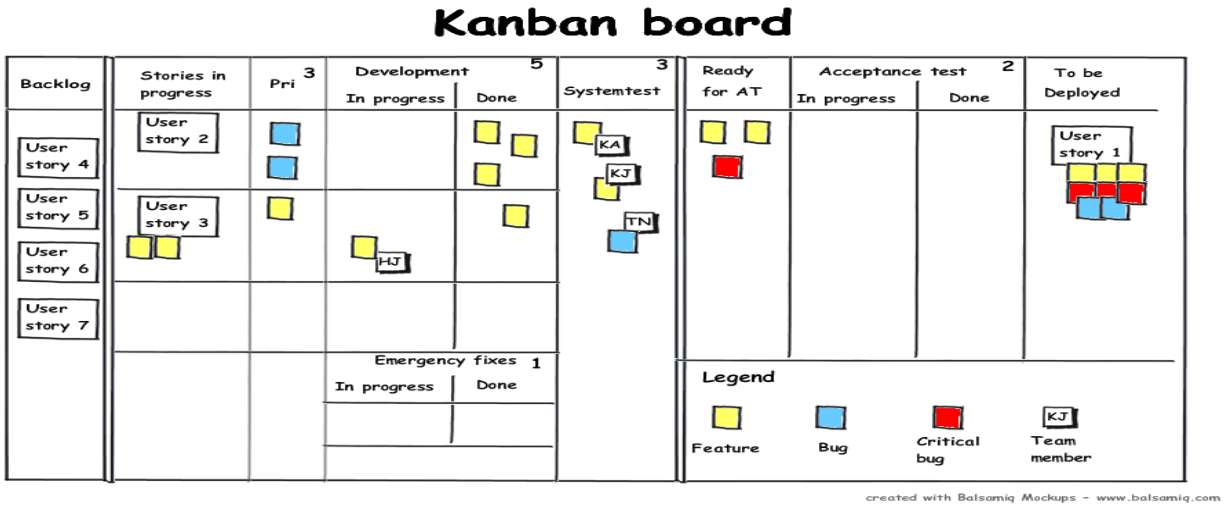

Figure 24 Example Kanban board

Notes placed on this board enable to track all the activities that need to happen for a release to production. The sponsor can be reassured to see a growing collection of completed work as the deadline approaches. It also focusses every one's attention on the remaining issues.

Velocity

This is the measure of a team's productivity, how much work/user story points can be done per iteration. It is based on experiences in past iterations. This metric provides a way to communicate to stakeholders what the team has accomplished, what they will likely be able to accomplish and when to expect the project (or release) to be completed. (This can be simply calculated by dividing the total amount of story points the product backlog contains by the average velocity of past iterations). After an increase of velocity in the beginning iterations due the learning cycle impacts, the velocity rate more or less stabilizes as the project progresses. One reason for this is that, as the product being built gets bigger, there is more to maintain, refactor and possibly support if earlier versions of the product have been deployed.

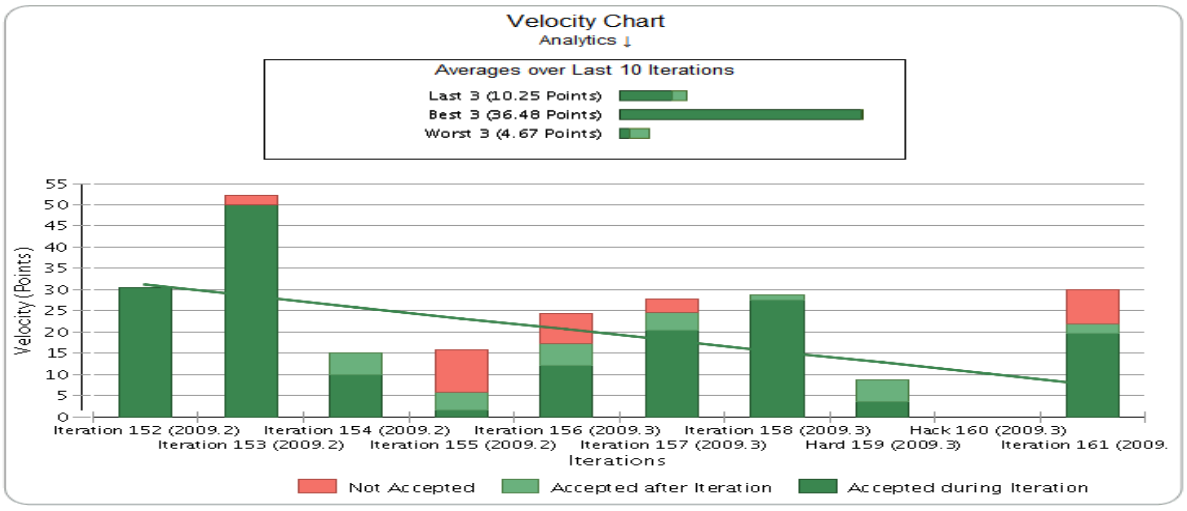

Figure 25 Example velocity chart

Burn down and burn up charts

These charts show progress to stakeholders in a very simple way. Burn down charts show estimated effort remaining on the project. As more work is completed a burn down chart will show a progress indicator moving downward to indicate the reduced amount of work (in hours or in points) that still needs to be done. Burn up

charts show what has been delivered. This chart will move upward to show the increasing amount of work completed. In a burn up chart scope increase (for instance in terms of story points) can be easily shown using an extra project scope line besides the actual delivery line. If work in progress is also added in the burn up chart we created the earlier mentioned cumulative flow diagram.

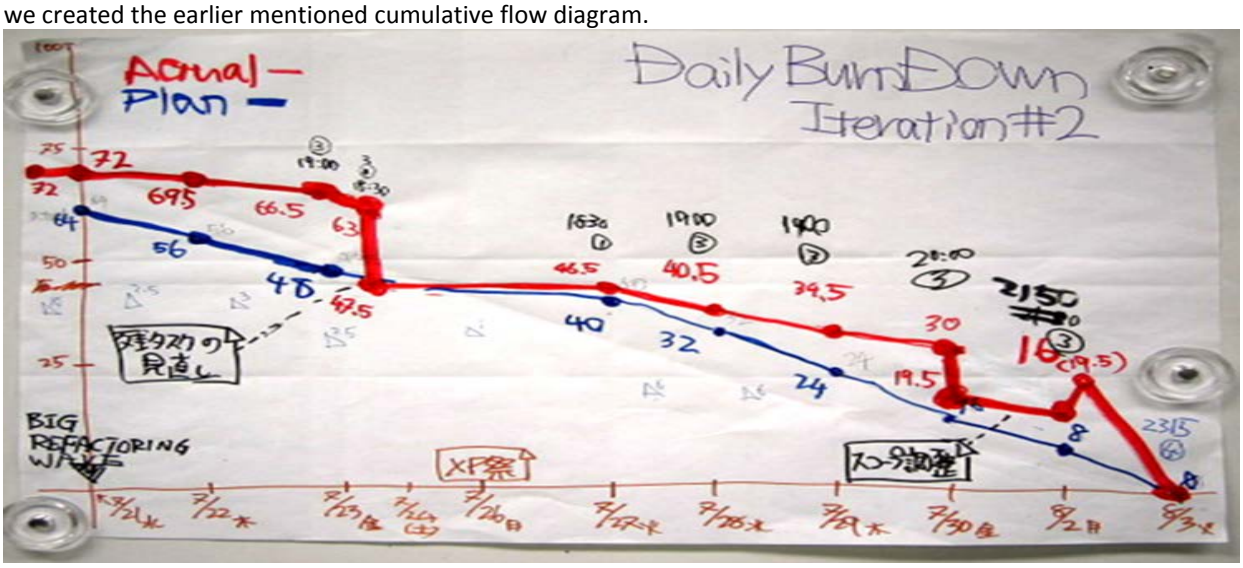

Figure 26 Example burn down chart

Active user involVement

The user is the person who will use the project results. (The customer acquires ownership over such use). In order to generate economic benefits for sponsors the user must accept the results and if necessary even change current ways of behavior. (For instance the use of a CRM application by sales staff is not just a matter of training their skills in using the application, but also change their behavior in regard of reporting outcomes of visits to customers and accounts). If users are not convinced of the added value of a new product/system/service in their daily operations, the result will not be used. This commitment from users is essential for project success. That's why user participation is a common element of most project methods. Since for the most part results are new for users, definitely if their behavior is also subject of change, users can not be simply asked what functionality they need. Characteristic for Agile project management is the way users are facilitated to learn more about requirements and priorities.

Iterative delivery

An iteration is a short development period (2-4 weeks) within a release, that results in the completion of a valuable deliverable on the project. So as opposed to a waterfall development approach, were it takes a long time before users are confronted with concrete results, iterations enable delivery of functionality (increments) in a very short time span. This continuous feedback loop helps users to understand better what the ultimate result of the project will be, (based on IKIWISI principle: I will only know it when I see it). It also offers an opportunity for discussion with users and getting their commitment. Doing so an extra opportunity emerges to gradually improve the ultimate value of the result based on growing knowledge about functionality.

Wireframes

When there is a lot of uncertainty about the scope and the proposed solution, it makes no sense to describe in words what the product should look like and what it should do. So instead of creating detailed specifications, Agile projects use visualizations to help users better understand what the result of the project will be. Wireframe models[2] are a popular way of creating a quick and cheap mock-up of the product. These mock- ups

[2] See for tools *www.sitepoint.com/**tools-prototyping**-wireframing*, http://hackdesign.org

can easily be changed until consensus is achieved. It prevents teams from investing a lot of time in building (potentially wrong) increments of the product.

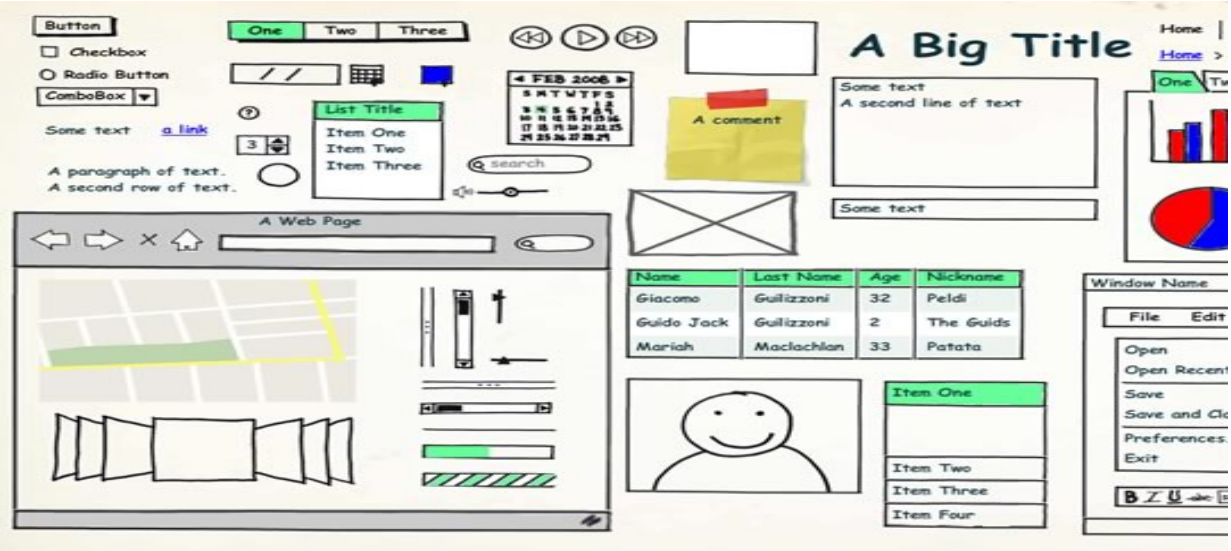

Figure 27 Example wireframe

Demonstrations

(Simulated) demonstration of functionality as part of the sprint review is critical to confirming success, definitely in intangible products like software. Therefore users need the opportunity to look at something and try it out to be able to confirm whether the functionality is suitable. The true requirements may only emerge once the product is demonstrated and used. In addition to helping clarify requirements, demonstrations can uncover the need for new features. So when teams demonstrate functionality two things happen. First the teams learn about the differences between what was asked for and what was interpreted and built. Second the team learns about new or adjusted functionality. Giving users a chance to experience and use something helps uncover the true business requirements, which may be quite different from the originally stated requirements.

Figure 28 Example product demonstration

High performing teams

The goal of a high performing team is to deliver results with demonstrable value for stakeholders. In order to become such a high performing team, it must pass several development stages.

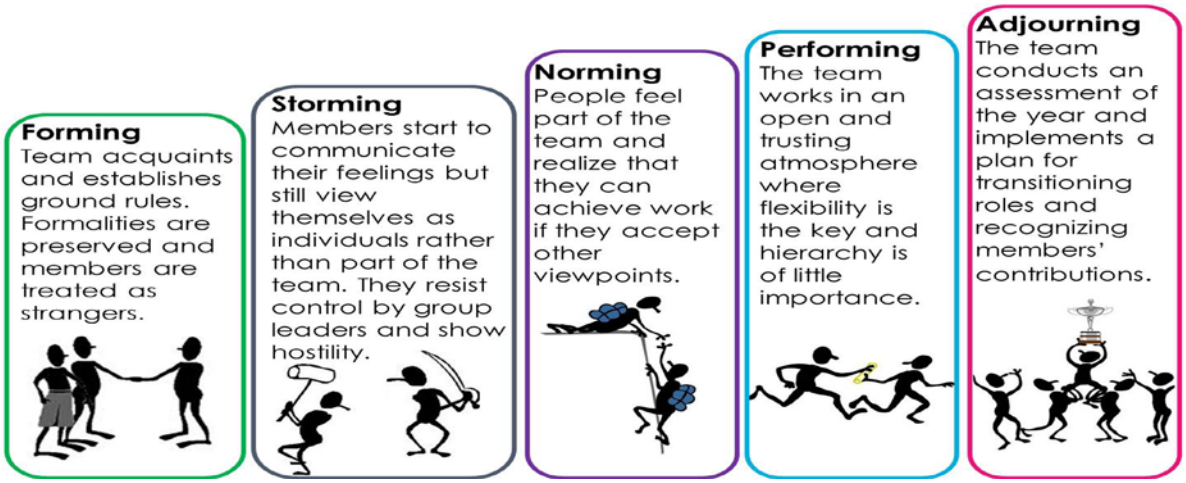

Figure 29 B. Tuckman Team development stages

In the forming stage specialist come together as a group of individuals. They orient themselves on the assignment and the task each individual has to perform. In the storming stage conflicts arise in the daily cooperation. This is an important learning stage in which the team members learn about themselves (see for instance M. Belbin's team roles), learn about importance of a common vision on the team's goal, how to interact effectively, how to solve conflicts, how to get a balance in task division, how to solve complex problems, how to take decisions. Based on their experiences the team gradually develops common codes of conduct for effective internal and external (stakeholders) cooperation based on a common vision on the results to be delivered. In this norming stage team members become comfortable in their roles and relationships, based on experiences from retrospective sessions the team becomes productive and aims for continuous improvement and aligning their work with other teams and stakeholders values. This is called the performance or maturity stage. Along this team development process the team becomes self-organizing.

Self - organizing and self - directing teams

Characteristic for Agile project management is a team work concept based on self-organizing and self-directing. Members of empowered teams are free from command and control management. They use their own knowledge to determine how best to do their job. Allowing teams to self–organize enables projects to tap into people's natural ability to manage complexity. Moreover the self–organizing concept is motivating for the team members. Specialists work harder and take more pride in their work when they are recognized as experts of their domain. When self- organizing teams select work items from the queue of waiting work. They have the expertise to perform the work that will bring them toward the iteration goal.

Self - direction means that teams not only figure out the best way to accomplish the work they committed to for an iteration, but they also resolve many of the daily issues that crop up along the way. This recognition that the team is in the best position to control the project work is logic. Agile projects are dynamic, so the team which is in continuous interaction with the stakeholders know best how to respond to new situations as soon as possible (See our reference to Ashby's law of requisite variety in white paper on Q2). This does not mean the project manager abdicates responsibility to the team. Instead it means the team is given freedom within the confines of iteration. If the team's estimates are way off or if they make poor technical decisions, these items will be detected and discussed at the iteration retrospective. In order to perform in this self-organizing and self-directing way teams should be "mature". So it is a goal, we do not start here. Teams need support and guidance as they come to grips with the project scope, tools and storming development issues. In white paper Q 4, when explaining leadership concepts we will go in more details how to build and coach high performing teams and how to motivate the team. Awareness of the cultural context is also important. In hierarchical organisations management resist to self-directing concepts. However when directing the project in a top down way, it may become a "self-fulfilling prophesy".

Daily stand- ups

These meetings are a core practice of Agile teams. They are short focussed meetings that negate the need for most other team status meetings. Daily stand- ups help to keep everyone focussed on the agreed scope and iteration goal. Daily stand-ups are time boxed to 15 minutes or less and are kept on schedule by having attendees only answer three questions:

- What have you worked on since the last meeting?
- What do you plan to finish today?
- Are there any roadblocks or impediments to your work?

Team space

This is the designated environment where teams conduct their everyday work. Since Agile prefers face to face interactions as a means of communication, co- location and collaborative team space are promoted. This team space should enable to share information, monitor progress and helping each other solving problems (see use of creativity techniques). There should be plenty of wall space for white boards to be used during collaborative discussions and to post information radiators of project metrics. When team members need some quiet time or privacy, so called caves (private offices) should be available.

Osmotic communication

This refers to the useful information that flows from team members as part of everyday conversations and questions when they work in close proximity to each other. Co-location enables this (“overhearing”) easy flow of information. So when projects are challenged of working with physically separated team members (or when team members are working off site in so called distributed teams), Agile methods recommend that project managers remove as many barriers to face to face osmotic communication as possible. Examples of tools which can be used: video conferencing, web based meeting facilitators, VoIP headsets, instant messaging, and interactive whiteboards.
If Agile teams are “distributed”, meaning teams are working at several locations, the Agile project manager should ensure there is enough debate and collective decision making early on in the project for the team to fully work through the several development stages.

Leadership

As stated before leading the project is based on formal organizational roles and personal behaviour.
The roles in a (temporary) project organization define who is responsible for what. These roles are related to directing a project, operational management of a project, delivery of products and related work packages, project support and project auditing. Unambiguous operational and decision procedures should support the coordination of the project.

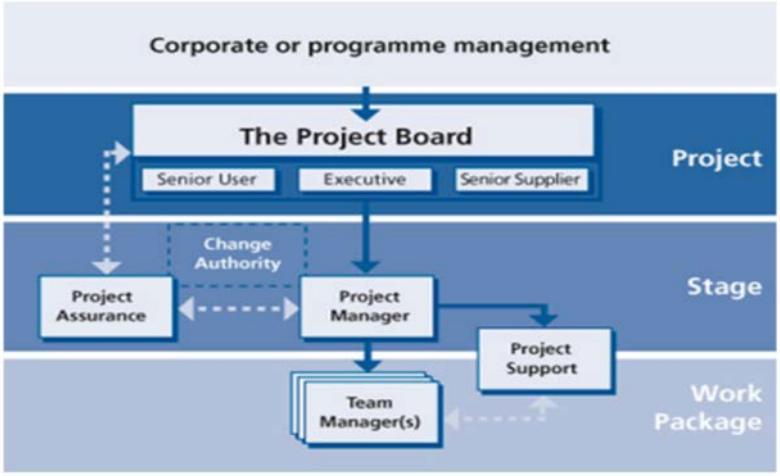

Figure 30 Project management organization

When Agile project management is based on a development framework for project delivery, leading the project will be based on the formal organization roles. Definitely in simpler Agile projects this will often be the case. (However we should be aware that in SCRUM the formal role of the project manager is not identified. Instead the role of the SCRUM master is defined).
When considering Agile project management as a separate project management method, the importance of formal organizational roles for projects leadership will diminishes. Instead of a hierarchy of roles, the project organization is seen as a team of players.

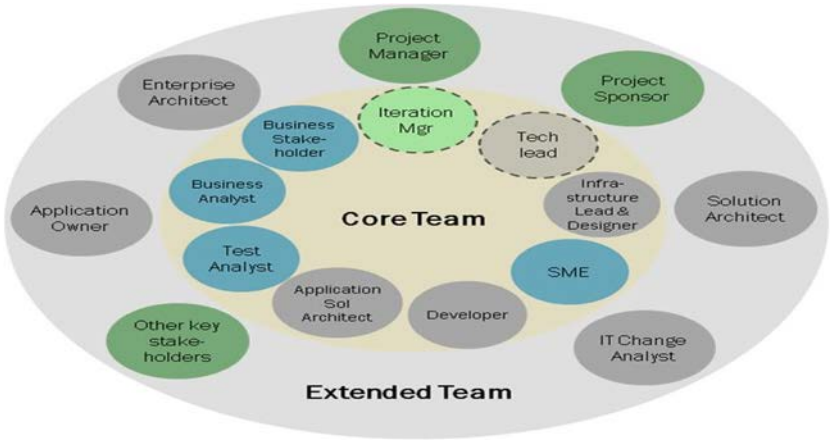

Figure 31 Agile's team based project organization

In a team based view on project organization, leadership behaviour becomes essential. This behaviour is based on a more "organic, are we doing the right things" approach instead of a more "mechanistic, are we in control and doing things right" approach. This is illustrated by the Agile value of "individuals and interactions over processes and tools". Next table show some differences:

Management focus	Leadership focus
Tasks /things	People
Control	Empowerment
Efficiency	Effectiveness
Doing things right	Doing the right things
Command	Communicate
Practices	Principles

Figure 31 leadership versus management

When answering Q 4 "what consequences does Agile project management have for the project manager" we will explain this leadership concept in more details.

Planning tools

Realistic planning and control is necessary to manage time, quality, costs, risks and scope in a feasible way during project delivery. Because Agile projects are characterized by a high dynamic environment; unforeseen issues and high rates of change are daily business. That's why planning is an ongoing process in Agile projects. This way of planning differs from the more static planning approach from stable and predictable projects, where most of the plan is created up front. The initial plan is only changed in response to exceptions and change requests.

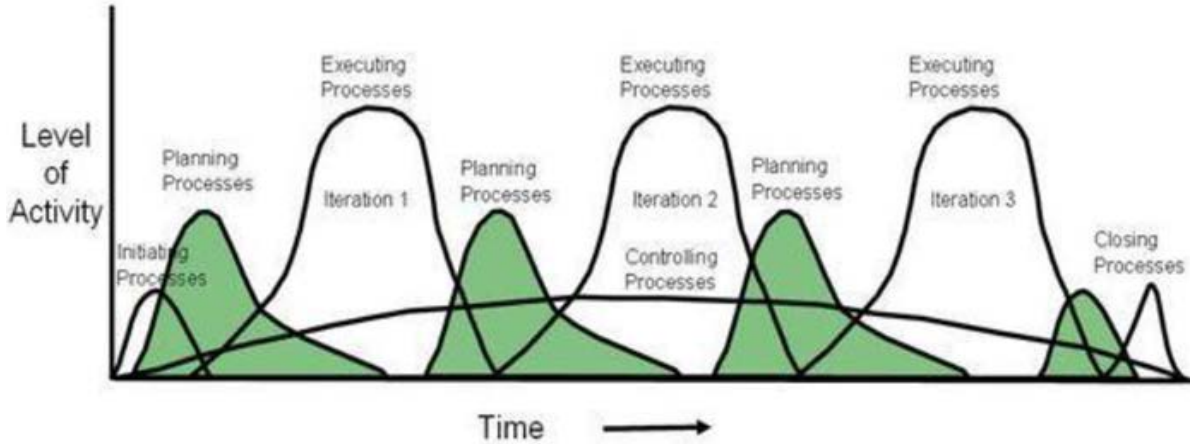

Figure 32 Agile planning as an ongoing process

Figure 32 also shows that the total amount of planning on an Agile project is often more extensive than on more traditional projects.

Before going in detail about Agile project planning techniques, let's present some more planning concepts behind Agile planning.

Time boxing

Time boxes (for instance 15 minute daily stand up meetings or two weekly iterations) are short, fixed duration periods of time in which work is undertaken based on (MoSCoW) priorities set by the business.

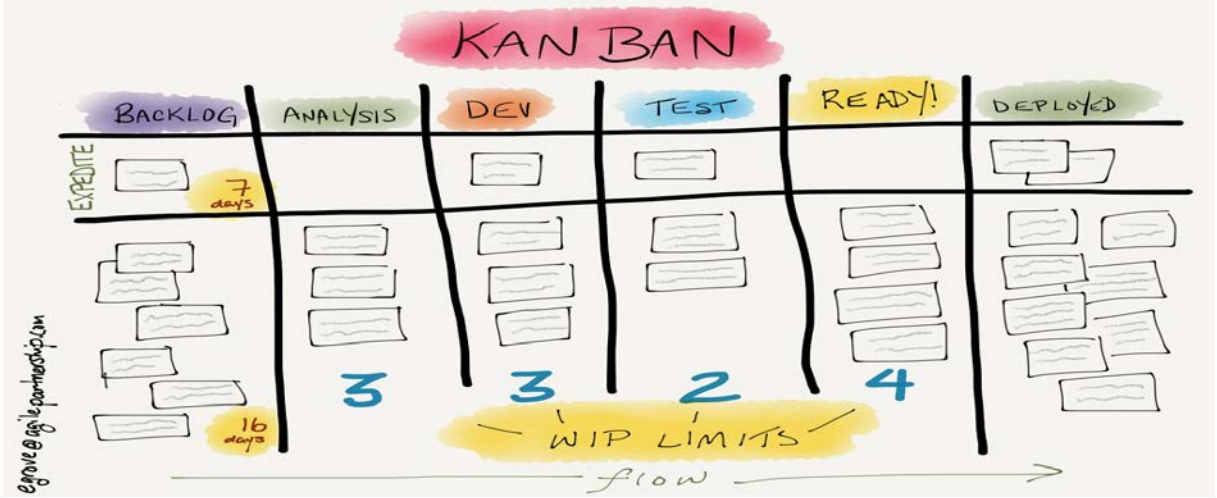

Figure 33 Activities in a time box

Besides time also capacity or costs ("money box") are fixed. This Agile planning approach differs from the waterfall based planning approach in stable and predictable projects, where planned (serial phased) results are fixed. Contrary to this if features or user stories planned for a time box are not finished when time runs out, the team stops what it is doing and moves the uncompleted work into another time box.
Time boxes help to bring some level of order and consistency to an otherwise variable work environment. They offer an opportunity to assess results, gather feedback and control the costs and risks associated with an endeavour.

Progressive elaboration

This name is given to the process of continual updates as information emerges. Progressive elaboration can for instance be used for risk assessments or definition of requirements, but also for planning and estimates. At the beginning of a project a team needs to plan and estimate the work involved to determine how big the endeavor is likely to be. At the same time at the beginning of a project the team knows the least, because there has not yet been any "learning by doing". So planning and estimation is not limited to the start of the project. By creating time spans of work/ iterations (the more uncertainty, the shorter a time span of work should be), the team is enabled to continually refine their plans and estimates as the project progresses and new details emerge. Based on retrospective reviews at the end of each iteration, the team can also improve and tailor project strategy and processes. So Agile planning is more like a "guided missile" approach suited for continuous moving targets.

Minimally marketable feature (MMF)

When planning a release of features, the release has to make sense, be useful and be valuable. The term MMF refers to this package of functionality that is complete enough to be useful to the users or the market, yet small enough that it does not represent the entire project. For instance, for a cell phone a MMF could be a phone that can be used to make and receive calls, store contact names and numbers and access voice mail, but it need not have a camera, internet connectivity or a music player in its first release. Instead these set of functionalities could be added in subsequent releases and evaluated independently. By using these MMF concepts lead time to market shortens and deliverance of value can start sooner.

Agile plans

Agile projects are divided into releases and iterations.
An iteration is a short (2-4 weeks) development period. A release is a group of iterations that results in the completion of a valuable deliverable on the project. It may be date driven ("we need something to demo at a trade show") or functionality driven ("once we can capture and process customer orders we can go live"). The project itself has one or more releases.

Figure 34 release planning

When planning a release we should ask: "what proportion of the user story backlog can be delivered in the release?" After having selected all features and related user stories for the upcoming release, the question then becomes: how likely is it that we will be able to complete this work by the release date? We initially rely on the team's estimates for the first release. Then after the team has gone through a few iterations, we can

start to look at velocity trends (How much story points can be realized per iteration?). For the most likely planning estimate, average velocity rates are used instead of the velocity rate of most productive iterations. As presented earlier we can use story maps and ultimately a product road map to show release plans. Story maps allow to lay out and group stories, first by dependencies (the "backbone" and "walking skeleton" representing the elements that we need to have on the project) and then by functionality.

When planning an iteration or sprint the team is asked to select user stories that the customer/product owner has indicated as high priority items (on the top of the backlog) and that can be developed, tested and delivered within the iteration. When getting on with the next iterations, the self- organizing principle should be respected by the project manager. So while the project manager could of course question any iteration projections that vary from the team's average velocity, it is important to allow the team to plan their own iterations.
In the first part of (a SCRUM-) sprint planning meeting the product owner (based on a freshly prioritized product backlog) describes the items he/she would like developed in the sprint. The team members select a set of items that they think are achievable based on their velocity rate (of course actual available capacity should be taken in account).[3] In the second half of the meeting, the team breaks down the selected backlog stories into their constituent parts to form the sprint backlog of action items. They then discuss how the work will be done, making local and external commitment to undertake the work within the sprint time frame.

Estimating

In order to determine which pieces of work can be done within a release or iteration, estimation is necessary. Estimates should be stated as ranges, (for example €4.000,- to €5.000,- or 16 to 18 weeks) to manage expectations about the project. The more uncertainty the wider the estimate ranges. Compare for example up front estimates which are the least accurate and have a wide estimate range. The estimation should become more accurate when the project progresses. Estimation is a continuous process. Team members who will be doing the work are involved in the estimation process. Four steps can be identified:

1. Determine the size of the project in story points or ideal days. (Agile techniques like wideband Delphi, planning poker, story points are used to estimate the size of the project).
2. Calculate the effort for the work in hours or person days.
3. Convert the effort into a schedule by factoring in the team size, required resources and dependencies between user stories.
4. Calculate the cost by applying labor rates and adding in other project cost elements.

Wideband Delphi

This is a group – based estimation approach. A panel of experts is asked to submit estimates. The quality of estimates is higher because this estimating is done anonymously. This minimizes "band wagon effect"(where people tend to agree with a prominent viewpoint) and the "halo effect" (where people tend to follow expert's or superior's opinions). Input for the wideband Delphi session is the team's problem specification, identified assumptions and constraints, and the outline of the process for subsequent rounds of estimation. Before beginning to create estimates, the invited participants read the problem specification and have an opportunity to raise and discuss qualification questions. Next they receive sheets of paper with spaces for entering estimates related to defined tasks. The facilitator gathers the estimates and plots them on a chart. The participants then discuss the different tasks and any assumption or other significant factors that influenced their estimation. Next the group repeats the anonymous estimation process. After several rounds, more consensuses are usually shown. Once the estimates have come together enough for the group to reach their exit criteria (for example the highest and lowest estimates must be within a range of +/-20 percent of the median estimate) the process stops. A single task list is then derived from everyone's individual task lists. If any tasks were excluded from estimation then those tasks are added to the master list. The result is reviewed to make sure everyone agrees upon the final task list and estimate range.

---

[3] Example: A team with 3 specialists working for 10 days in a sprint can produce 25 points. If actual available capacity is less, for instance 5 days, because of refactoring or planned absences, then the team can only produce 15/30 x 25 points.

Planning poker

This technique combines all of the elements of wideband Delphi in a fast, collaborative process. Planning poker uses playing cards with numbers on them. The numbers represent sizing units such as developer days or story points. Each planning participant receives a set of cards. Once the cards have been distributed, the product owner reads a user story. This story is discussed briefly in the group before each estimator selects a card to represent his or her estimate for the user story. The participants all turn over their cards simultaneously, so that everyone can see the numbers. If the range is small and there is little debate about the estimates, the process moves onto the next story to keep the game moving quickly. If however the range is big, outliers (with a substantial higher score than the median) will be discussed. Based on this extra information the estimation process is repeated. The goal of the exercise is not to create precise estimates. Instead this technique helps the team quickly and cheaply achieve consensus around reasonable estimates to move the project forward.

Figure 35 Planning poker

Story points/ relative sizing

This technique helps to solve common problems with estimation. First of all when factoring all kinds of unforeseen tasks or issues into estimates, the estimators would be criticized for padding the estimates. It makes estimation unpopular; you are never doing a good job in the eyes of others. Besides this, people in general are not good at predicting the absolute size of the work in concrete time units like hours of days. Making comparative estimates the team uses concrete experience from chunks of work already executed to estimate new pieces of work. For example based on the experiences with needed capacity for developing a simple input screen, the team gives this effort a score of 2 points. The work to change a screen can be given 1 point because the team thinks it's only about half as much work as developing the simple reference screen. Of course taking a comparative approach to estimating does not stop unexpected issues from happening or keep activities from taking longer than anticipated, but switching the estimation unit from concrete hours to relative story points makes it easier to accept this reality. Story points (also called "points" or "gummy bears") can be more easily related to result or value driven language than hours spend.

Ideal time

When estimating work, the topic of how best to factor in interruptions, diversions, non-project work (like checking e- mails, attending staff meetings) usually crops up. We can simplify the discussion by talking about ideal time. This means we ask team members to estimate as if there were no interruptions. This is obviously not the real life situation, but the purpose of talking about real time is to simplify estimation processes by taking the variable of availability out of the equation. In doing so, we get a more accurate sense of the effort involved in the work.

Affinity estimating

This technique is often used to group similarly sized user stories together. It provides a comparative view of the estimates and gives the team an opportunity to do a reality check. By placing user stories into size categories (for example in categories of 1, 2, 3, 5, 8, 12 story points), it is easier to see whether user stories that are assigned similar estimates are in fact comparable in size. As a new user story is estimated, it should also be placed in the appropriate column and compared to the cards that are already there. It helps the team make sure they have not gradually altered the measurement value of a story point.

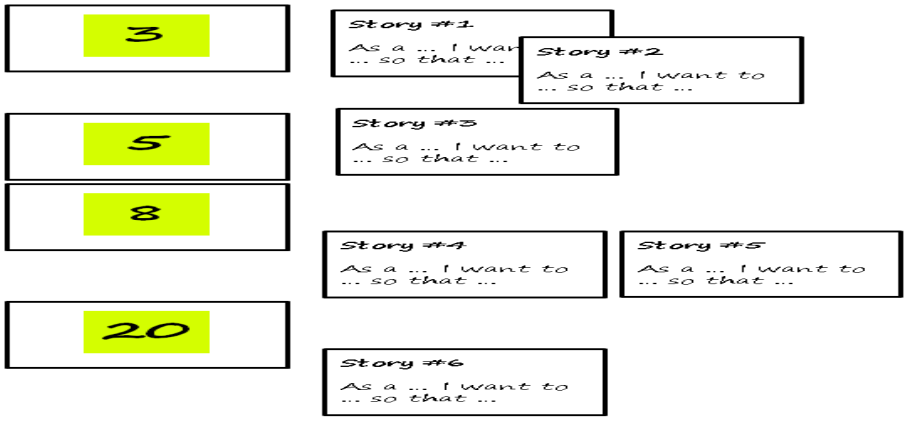

Figure 36 Affinity planning

Control tools

In traditional project management methods the initial project plan from the beginning of the project lifecycle is leading for monitoring and controlling project progress. Timely delivery within given budget and quality specifications is leading in control. When confronted with uncertainty a stage approach is used. Changes will be encountered from a control perspective.
Agile projects are characterized by lots of uncertainty. Changes are seen as a normal phenomenon linked to business (-stakeholders) learning based on growing knowledge as a project progresses. Delivering business value as soon as possible is important in a rapidly changing environment. Value delivery can be monitored by several techniques.

Earned value analysis

When emphasizing value driven delivery it is important to monitor the rate at which product features are being delivered to make sure we are on track to create value for stakeholders. The earned value indicates the budgeted value of work performed. In a double S-curve graph we show an actual spending/cost line (tracked against a budget) and a feature based work performed line (tracked against tasks/ story points that are related to the scope). The curve of this (earned value) work performed line indicates velocity/ productivity; where it raises steeply lot of story points are developed where it is flat progress was slow. Adding a background to the graph can be a useful way to show the functional areas of the project.

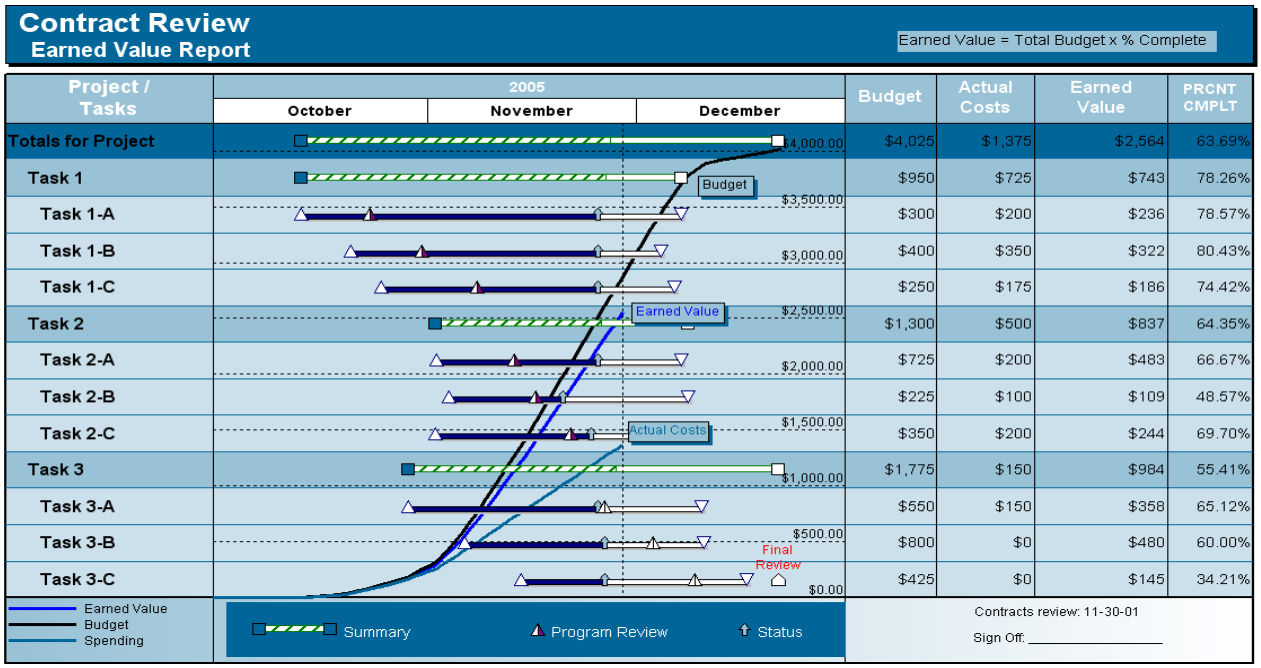

Figure 37 Example earned value analysis

Cumulative flow diagram

In a cumulative flow diagram we can see in a presented timeline the number of features to be built, the number of features in progress and the number of features completed. On the area showing the work/ features in progress (WIP) we can see how many items are in the queue by looking at the vertical distance (length of the queue) and by looking at the horizontal distance we can see how long it will likely take to complete the features in the queue.

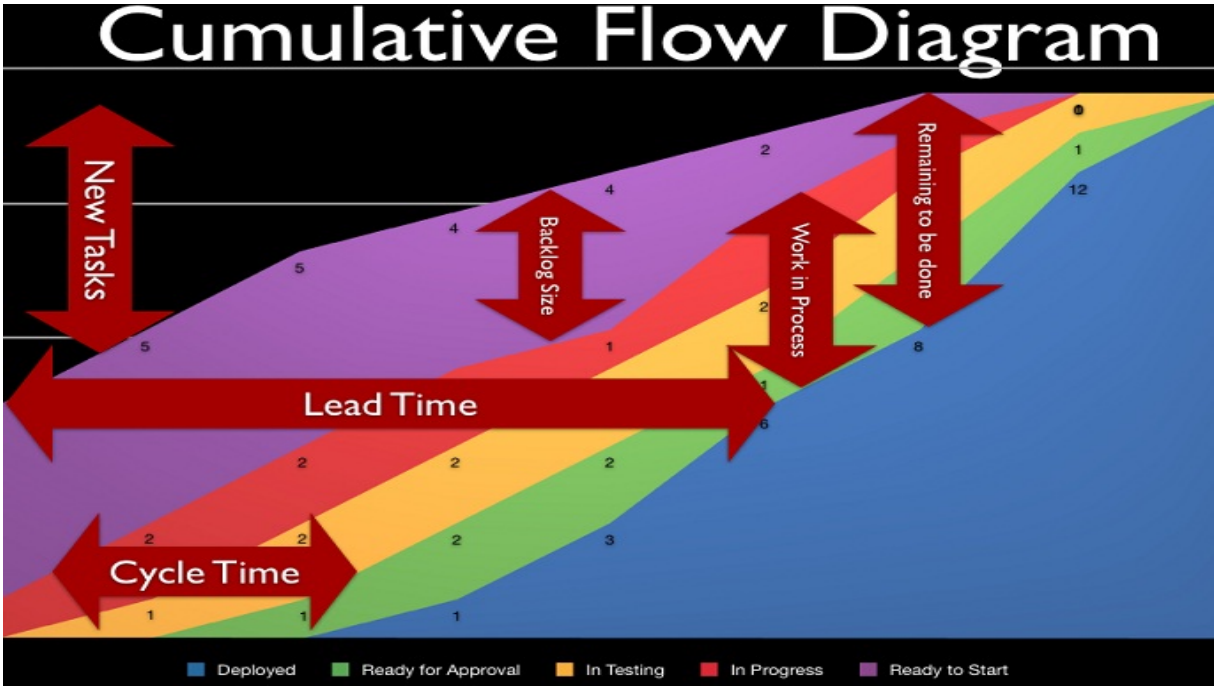

Figure 38 cumulative flow diagram

Risk burn down chart

In a time line we can show the effectiveness of risk management actions relating to the risk score scale of 1-5 for probability and 1-5 for impact.

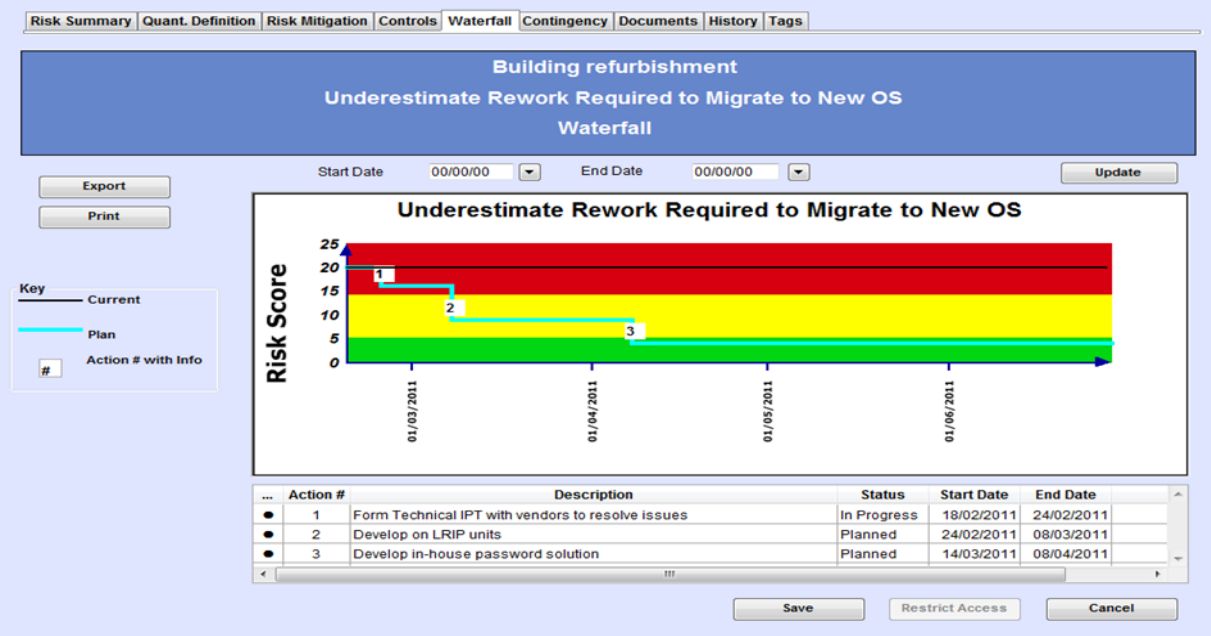

Figure 39 Example risk burn down chart

Retrospectives

Characteristic for Agile project management is the frequent evaluation of the short, time boxed iteration loops that deliver tested and accepted results. This enables collective learning as a way to handle dynamics due to uncertainty. Retrospectives and sharing knowledge play an important role. These meetings take place after each iteration. Contrary to the lessons learned reports with advices for similar projects in the future, retrospectives offer immediate value to the current project. For example by improved productivity and team member capabilities or increased functionality.

The retrospective process (approximately 2 –hour session), goes through the following five steps:

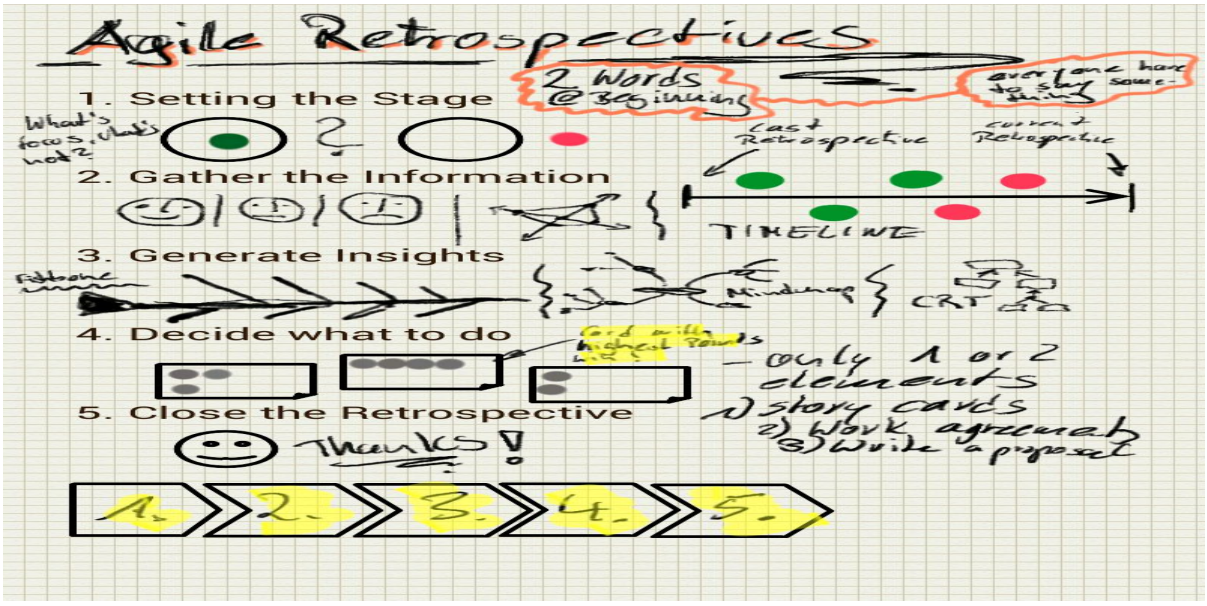

Figure 40 Retrospective process

1. When setting the stage it is important to create an atmosphere where people feel comfortable speaking about things that may not have gone so well on the project. The focus is improvement, not

blaming people for what went wrong. Definitely when doing this session for the first time, it is important to get every participant actively involved in the process. The facilitator invites everyone in the early beginning of the process to outline what they hope to get from the retrospective or to say one or two words that describe how they felt about the iteration and the progress made. Next the retrospective approach (the procedures, interaction principles for productive communication) and topics for discussion are outlined.

2. When gathering data the participants create a shared picture of what happened during the iteration. Events (for instance based on daily stand up meeting experiences and collected using brain writing techniques) are written on colored sticky notes (color indicates good, problematic or significant) and placed on a timeline. Below the timeline feelings about the events are recorded and a trend line can be drawn to indicate the feelings.
3. When generating insights gathered data from the previous step are analyzed to make sense of them. Brainstorming and five time why techniques can help to identify causes of problematic issues. A fishbone diagram can help to link causes and effects. Using dots causes can be prioritized. Creativity techniques can be used to generate ideas for improvement.
4. When deciding what to do the team moves from thinking about the iteration they just completed into thinking about the next iteration. What to keep and what to change? Based on generated ideas for improvement the team creates more detailed action plans and sets measurable goals to achieve desired results.
5. When closing the retrospective the team members reflect on what happened during the retrospective, express appreciation for everybody's dedication and summarizes (plus / delta process improvements) what the team should do more of (things that are going well) and what the team should change(things that are not going well).

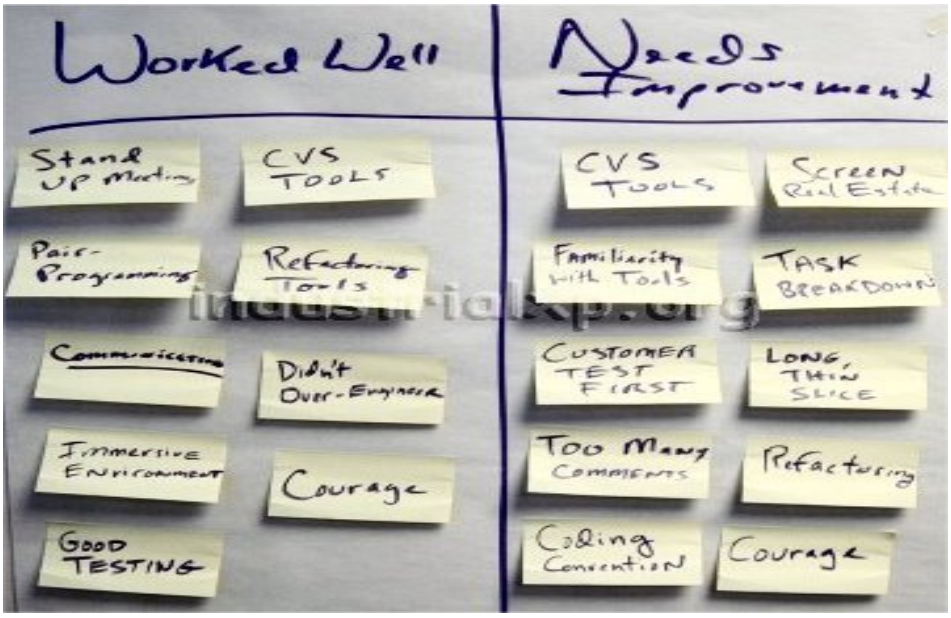

Figure 41 Example results from retrospective

Knowledge sharing

This is a key component of Agile methods. The retrospective process we just described is a perfect illustration of sharing knowledge. Other examples are common code ownership and pair programming.

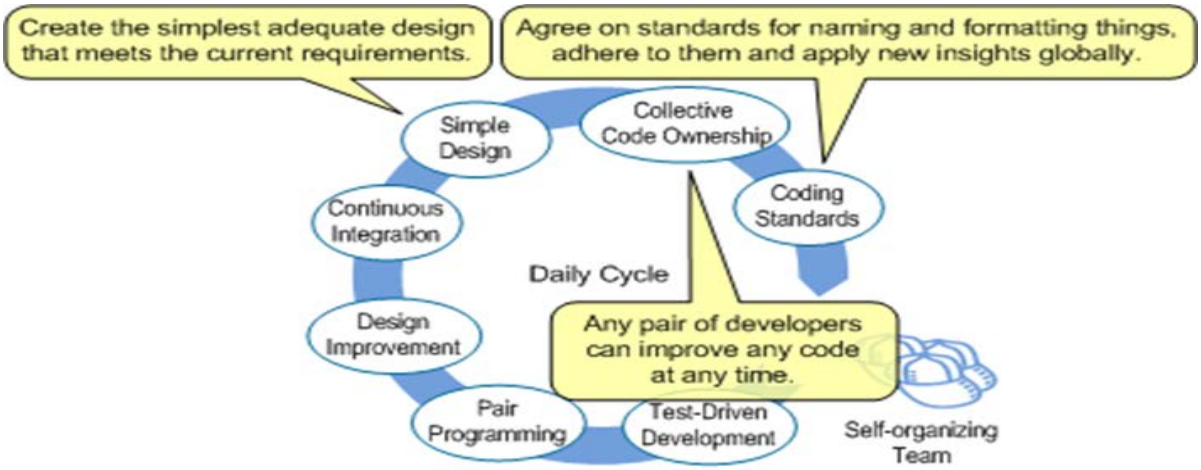

Figure 42 Sharing knowledge

Another example of knowledge sharing is the communication between the team and the customer when demonstrating a product. This could lead to conversations like: “here are what we as a team thinks you asked for and what we have been able to build. Please tell us if we are on the right track”. Customer to team:” I like these bits and this is ok, but on second thoughts you got this piece wrong. O and that reminds me we really need something over here to do X”.

Another example of (less obvious) way to share information is “overhearing” in co- located teams. The daily stand up meeting is not done via smart phones because productive information sharing is better stimulated by making it a team effort. Information radiators or cards on a wall or story points are other examples of simple ways to exchange information.