Agile IT – Project Management

DI Philipp Rosenberger
philipp.rosenberger@fh-campuswien.ac.at

Content of todays lecture

- Agile Mindset
- SCRUM
- Hybrid/ Collaborative Model
- The agile Project Manager
- KANBAN in IT projects

Introduction

IT Project Management Mindset

Brainstorming:

When is a project a project?

>

>

What defines a project as IT project?

>

>

Definition of a project? (according PMI)

A project is **temporary** in that it has a defined beginning and end in time, and therefore defined scope and resources.

And a project is **unique** in that it is not a routine operation, but a specific set of operations designed to accomplish a singular goal. So a project team often includes people who don’t usually work together – sometimes from different organizations and across multiple geographies.

IT Project Management Mindset

Brainstorming:

What defines an excellent project manager?

>…

>…

What's the difference between an excellent project manager and an excellent IT project manager? What are additional skills of IT project managers?

>…

>…

The agile mindset

Empirical and the unknown

Empirical projects:

Definition “empirical”:

1)

a. Relying on or derived from observation or experiment: empirical results that supported the hypothesis.

b. Verifiable or provable by means of observation or experiment: empirical laws.

2. Guided by practical experience and not theory, especially in medicine.

em·pir’i·cal·ly adv.

Means: <u>We need to gain the information along the way!</u>

<u>When to use?</u>

we use AGILE methodologies if the cost of what we don’t know is higher, then the cost of experimenting until we know.

Planning versus experimenting

Comparing planning to experimenting (agile). Experimenting does not mean just try and error. Its empirical processes to discover what we need to know to make good decisions.

Do Plan based project have a discovery process?

Yes. When we discover that our plans were wrong? Possibilities:

-) Ignore: Because it makes our plan look bad

-) Change: sometimes the plan is too rigid and it makes to much trouble to change.

-) Sometimes we plan for failure: So we plan problems and uncertainties and hope that we plan enough. So there is risk. So we push all these uncertainties all upfront and take risk. But what about all the information developing throughout the way? Why don’t we accommodate that? So better then planning every catastrophic situation upfront, why don’t we allow decisions to be made later and use the latest up to date information for these decisions?

Structure of Discovery

So to do this learning and discovery we need structure. Why do we need structure?

Because we are part of a collaboration. Part of a team. And we need alignment with other members. It’s the shared language we use.

So if everyone makes experiments and hope to learn for his own, that’s chaos. Agile needs processes.

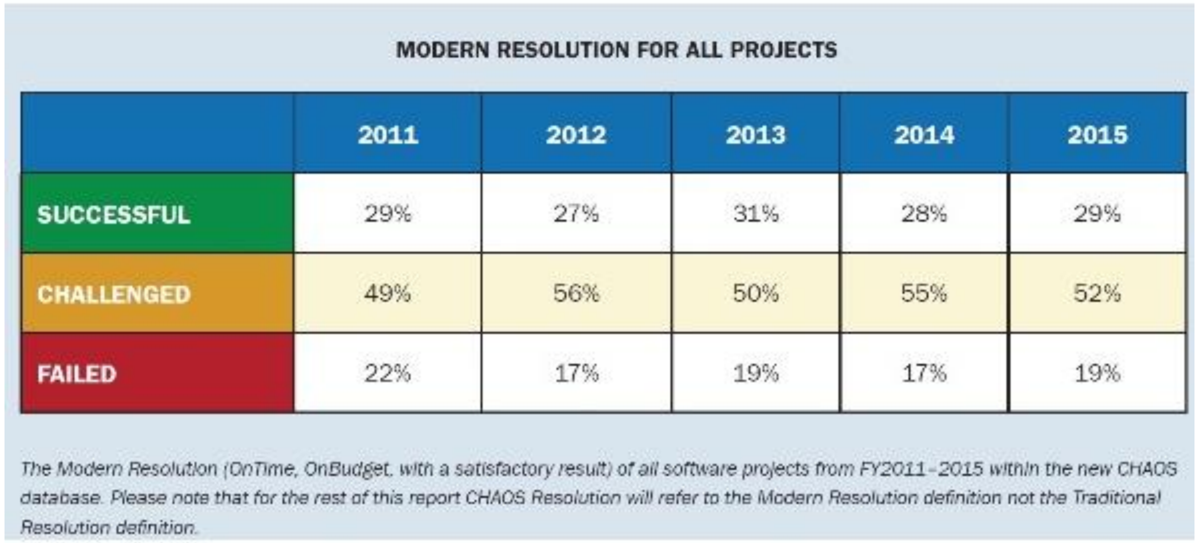
4 pillars of structure:

- Vision of Outcomes (what is the software going to do?)
- Method of enactment (when do we work, how do we work,…)
- Alignment and collaboration (Agile: it’s the people who have the good ideas. The process needs to support the discovery.)
- Measure of value. (Clear understanding of measuring what you are creating)

Why all this effort to find new methods?

Where are we today?

Standish Group (http://www.standishgroup.com) (für 2015):



MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

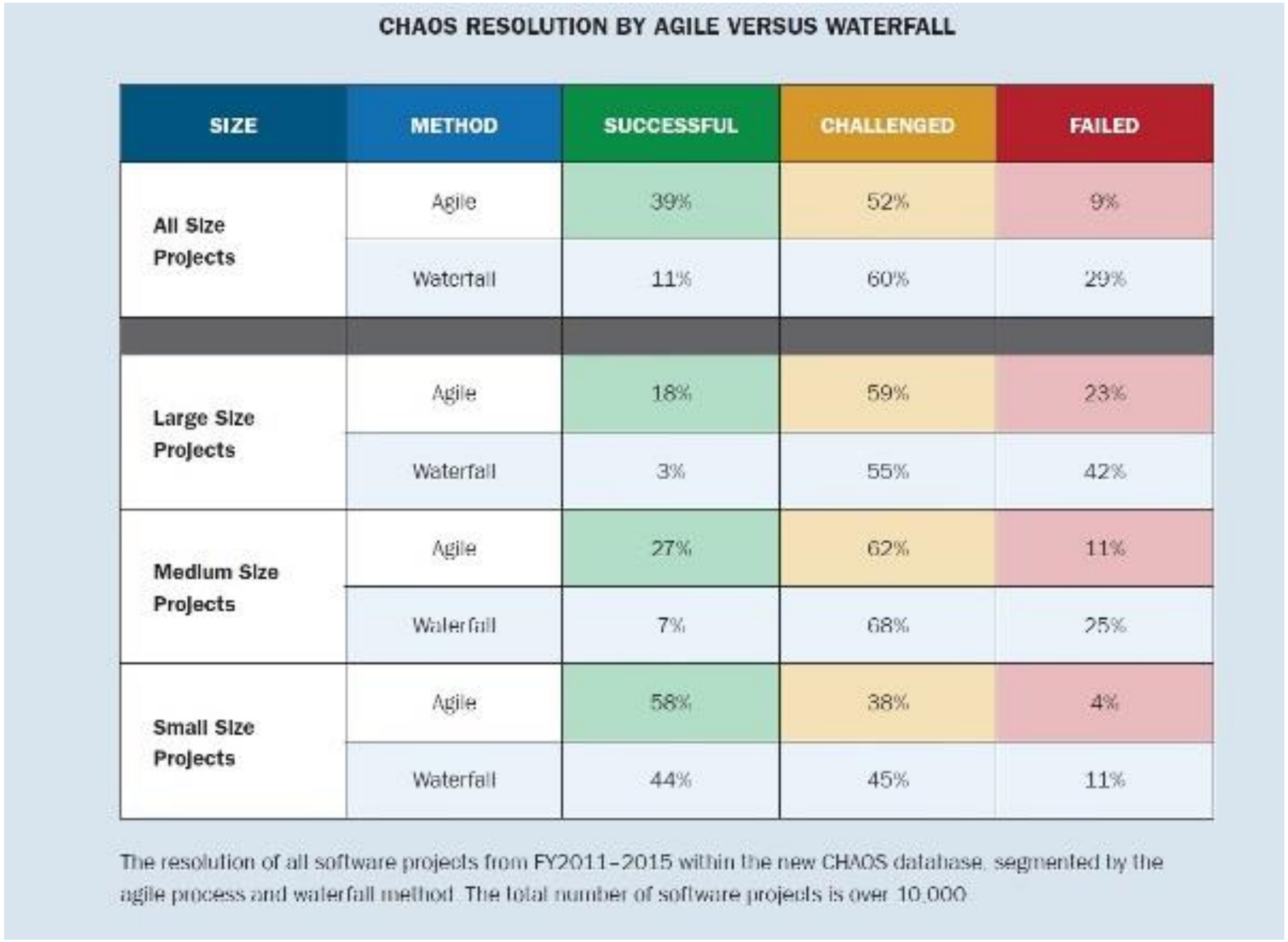
The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011-2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

http://www.infoq.com/articles/standish-chaos-2015

Reasons why projects fail:

Unclear and anconnected requirements and high complexity!

Why all this effort to find new methods?



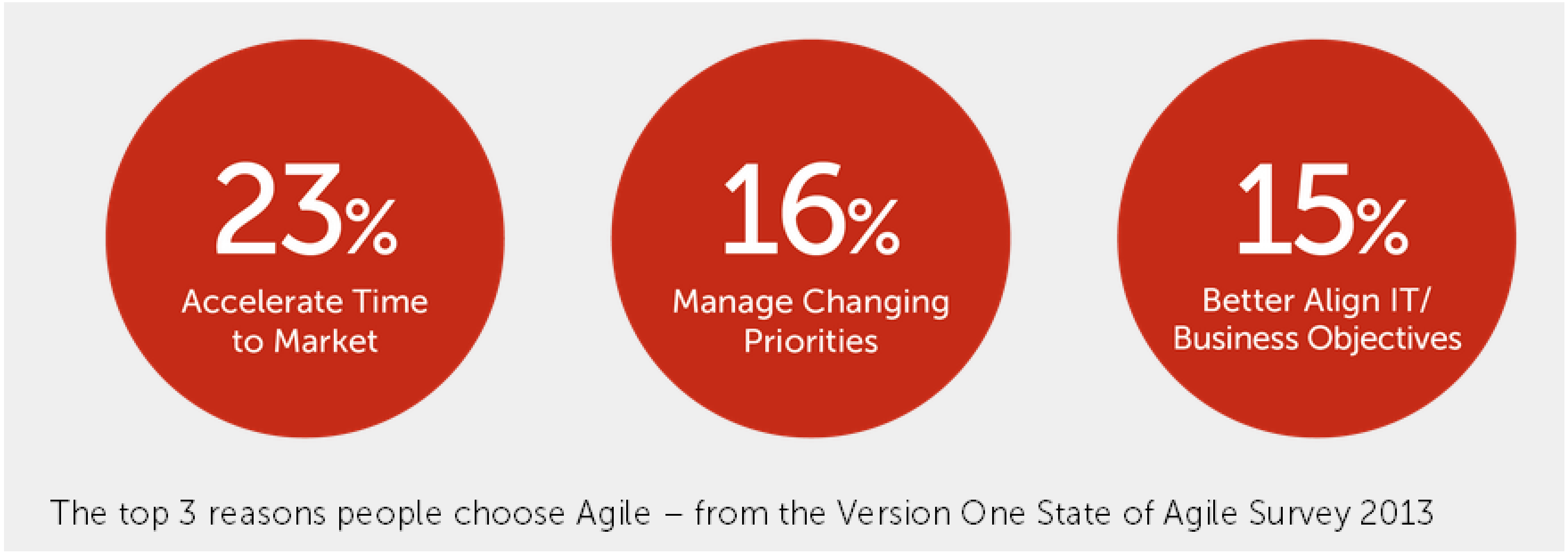
CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

IT Project Management according SCRUM

Why SCRUM?



The top 3 reasons people choose Agile – from the Version One State of Agile Survey 2013

SCRUM Guidelines

Transparency

Significant aspects of the process must be visible to those responsible for the outcome. Transparency requires those aspects be defined by a common standard so observers share a common understanding of what is being seen.

For example:

- A common language referring to the process must be shared by all participants; and,
- Those performing the work and those accepting the work product must share a common definition of “Done”.

SCRUM Guidelines

Inspection

Scrum users must frequently inspect Scrum artifacts and progress toward a Sprint Goal to detect undesirable variances. Their inspection should not be so frequent that inspection gets in the way of the work. Inspections are most beneficial when diligently performed by skilled inspectors at the point of work.

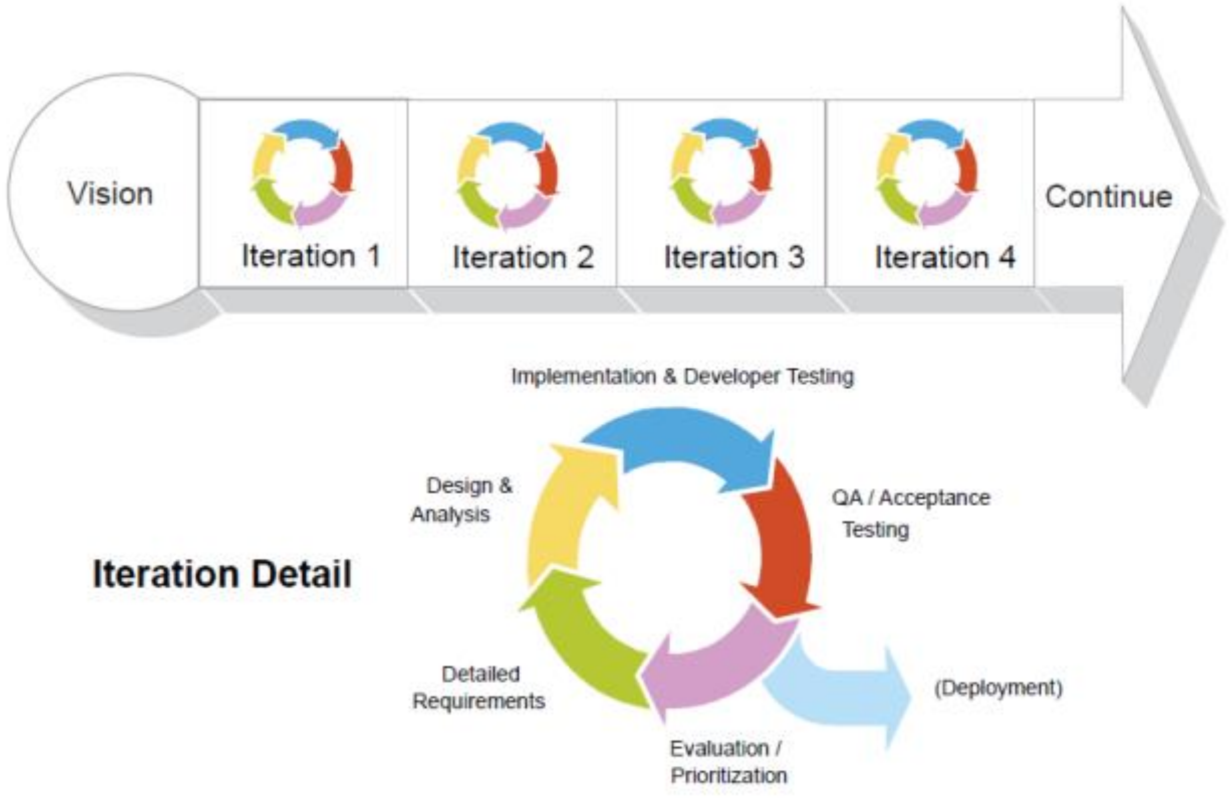
SCRUM Guidelines

Adaptation

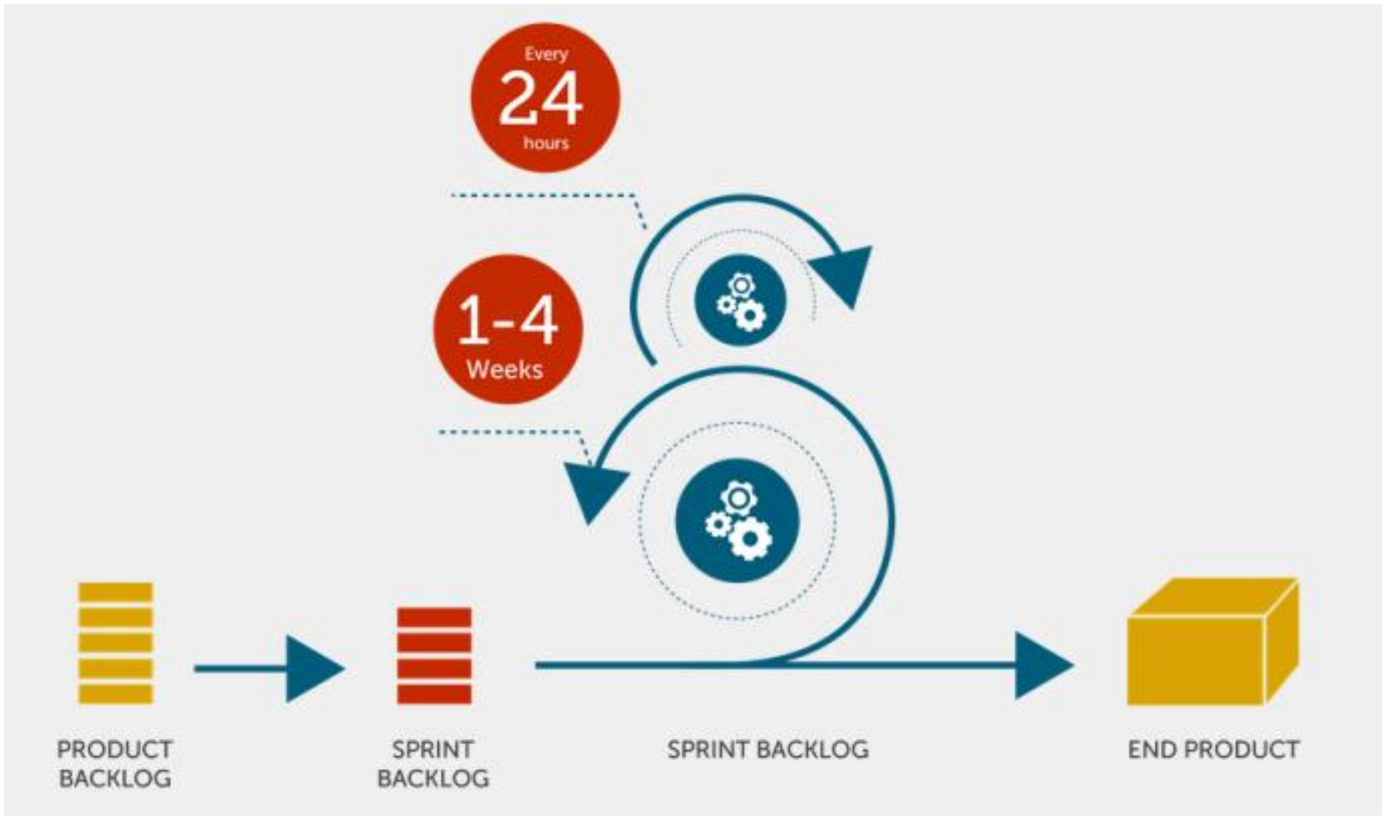
If an inspector determines that one or more aspects of a process deviate outside acceptable limits, and that the resulting product will be unacceptable, the process or the material being processed must be adjusted. An adjustment must be made as soon as possible to minimize further deviation.

Definition of SCRUM

Accord. Michael James Scrum Reference Card



Overview „Scrum – Sprints“



Scrum Roles

Product Owner

- Single person responsible for maximizing the return on investment (ROI) of the development effort
- Responsible for product vision
- Constantly re-prioritizes the Product Backlog, adjusting any long-term expectations such as release plans
- Final arbiter of requirements questions
- Accepts or rejects each product increment
- Decides whether to ship
- Decides whether to continue development
- Considers stakeholder interests
- May contribute as a team member
- Has a leadership role

Scrum Roles

Scrum Development Team

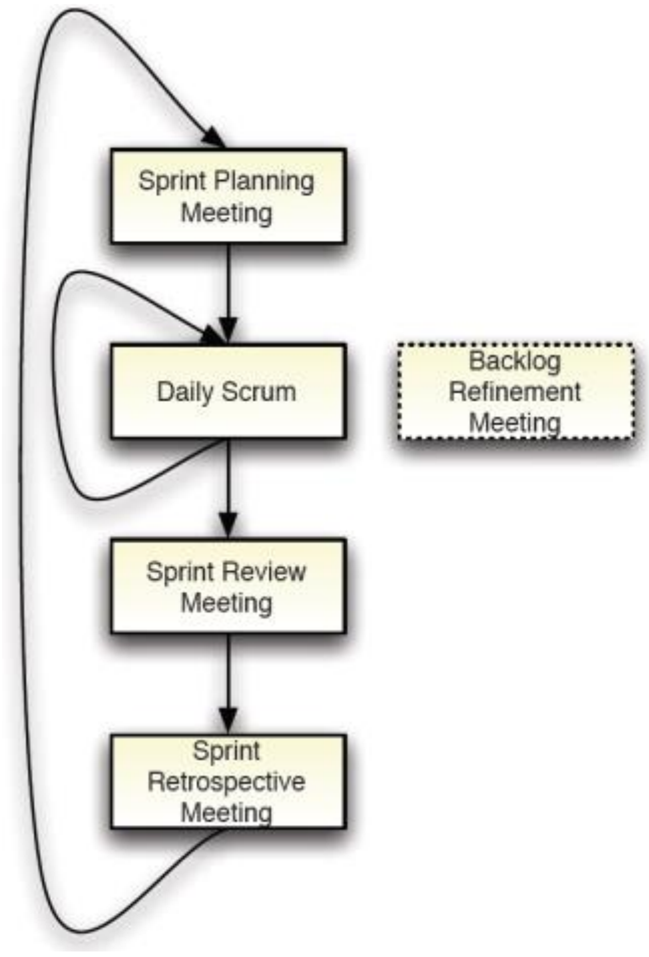
- Cross-functional (e.g., includes members with testing skills, and often others not traditionally called developers: business analysts, domain experts, etc.)
- Self-organizing / self-managing, without externally assigned roles
- Negotiates commitments with the Product Owner, one Sprint at a time
- Has autonomy regarding how to reach commitments
- Intensely collaborative
- Most successful when located in one team room, particularly for the first few Sprints
- Most successful with long-term, full-time membership. Scrum moves work to a flexible learning team and avoids moving people or splitting them between teams.
- 7 ± 2 members
- Has a leadership role

Scrum Roles

ScrumMaster

- Facilitates the Scrum process
- Helps resolve impediments
- Creates an environment conducive to team self-organization
- Captures empirical data to adjust forecasts
- Shields the team from external interference and distractions to keep it in group flow (a.k.a. the *zone*)
- Enforces timeboxes
- Keeps Scrum artifacts visible
- Promotes improved engineering practices
- Has no management authority over the team (anyone with authority over the team is by definition not its ScrumMaster)
- Has a leadership role

Scrum Meetings



Sprint

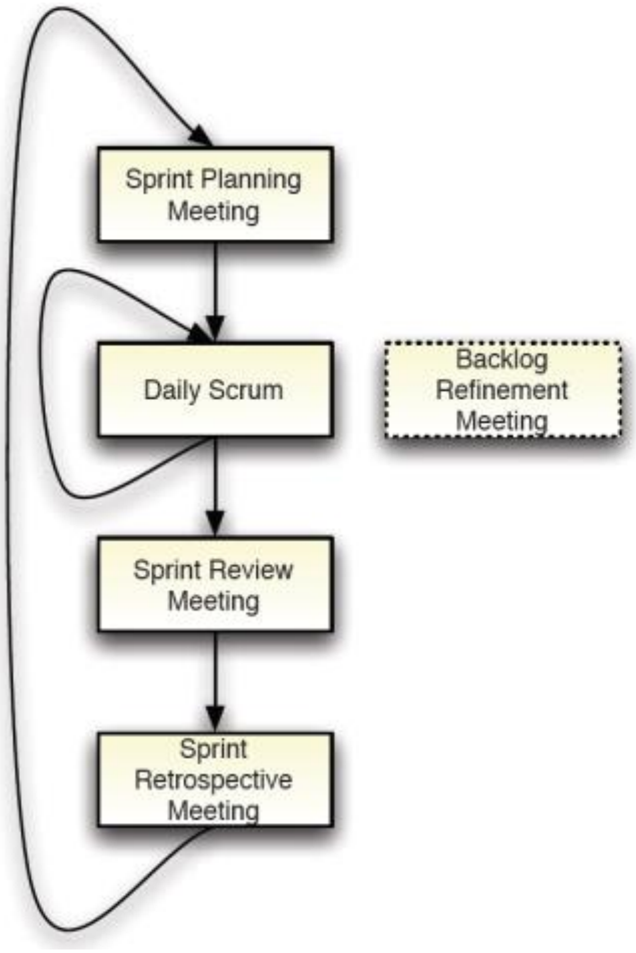
>The heart of Scrum is a Sprint, a time-box of one month or less during which a "Done", useable, and potentially releasable product Increment is created.

>Sprints best have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint.

>During the Sprint:

- » No changes are made that would endanger the Sprint Goal;
- » Quality goals do not decrease; and,
- » Scope may be clarified and re-negotiated between the Product Owner and Development Team as more is learned.

Scrum Meetings



Sprint Cancelling

A Sprint can be cancelled before the Sprint time-box is over. Only the Product Owner has the authority to cancel the Sprint, although he or she may do so under influence from the stakeholders, the Development Team, or the Scrum Master.

A Sprint would be cancelled if the Sprint Goal becomes obsolete or if it no longer makes sense given the circumstances.

When a Sprint is cancelled, any completed and “Done” Product Backlog items are reviewed. If part of the work is potentially releasable, the Product Owner typically accepts it. All incomplete Product Backlog Items are re-estimated and put back on the Product Backlog. The work done on them depreciates quickly and must be frequently re-estimated.

Sprint cancellations consume resources, since everyone has to regroup in another Sprint Planning to start another Sprint. Sprint cancellations are often traumatic to the Scrum Team, and are very uncommon.

Sprint Planning meeting

Definition of: What can be delivered in the Increment resulting from the upcoming Sprint?

>Development Team works to forecast the functionality that will be developed during the Sprint. The Product Owner discusses the objective that the Sprint should achieve and the Product Backlog items that, if completed in the Sprint, would achieve the Sprint Goal. The entire Scrum Team collaborates on understanding the work of the Sprint.

>Input to this meeting is the Product Backlog, the latest product Increment, projected capacity of the Development Team during the Sprint, and past performance of the Development Team. The number of items selected from the Product Backlog for the Sprint is solely up to the Development Team. Only the Development Team can assess what it can accomplish over the upcoming Sprint.

Sprint Planning meeting

Definition of: Who will do the chosen work?

>The Development Team self-organizes to undertake the work in the Sprint Backlog, both during Sprint Planning and as needed throughout the Sprint.

>The Development Team may also invite other people to attend in order to provide technical or domain advice.

>At the start of each sprint

>Duration of 1day for a 30 days sprint

>Participants: Development Team, Product Owner and Scrum Master

Sprint review meeting

- Should feature a live demonstration, not a report.
- After the demonstration, the Product Owner reviews the commitments made at the Sprint Planning Meeting and declares which items he now considers *done*. (For example, a software item that is merely “code complete” is considered *not done*, because untested software isn’t shippable).
- Incomplete items are returned to the Product Backlog and ranked according to the Product Owner’s revised priorities as candidates for future Sprints. The Scrum Master helps the Product Owner and stakeholders convert their feedback to new Product Backlog Items for prioritization by the Product Owner.
- The Sprint Review Meeting is the appropriate meeting for external stakeholders (even end users) to attend. It is the opportunity to inspect and adapt the product as it emerges, and iteratively refine everyone’s understanding of the requirements.

Daily Scrum Meeting

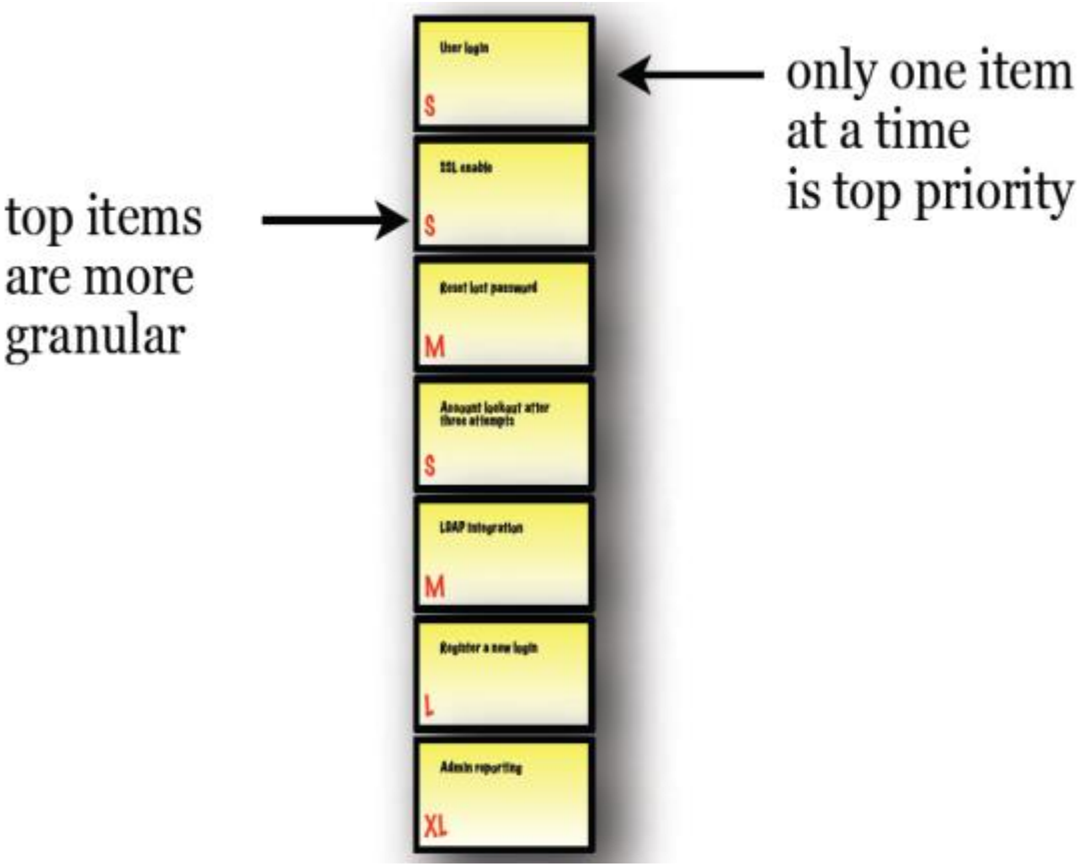
- Participants: Development Team, Scrum Master, interested participants to listen.
- Questions:
  - What did we achieve yesterday?
  - What are going to do today?
  - What's keeping me from good performance and reaching my goals?
  - What do I need to be able to work without being disturbed?
- The team may find it useful to maintain a current Sprint Task List, a Sprint Burndown Chart, a KANBAN chart and a difficulties list.

Sprint retrospective meeting

> Each Sprint ends with a retrospective. At this meeting, the team reflects on its own process. They inspect their behavior and take action to adapt it for future Sprints. Dedicated Scrum Masters will find alternatives to the stale, fearful meetings everyone has come to expect. An in-depth retrospective requires an environment of psychological safety not found in most organizations. Without safety, the retrospective discussion will either avoid the uncomfortable issues or deteriorate into blaming and hostility.

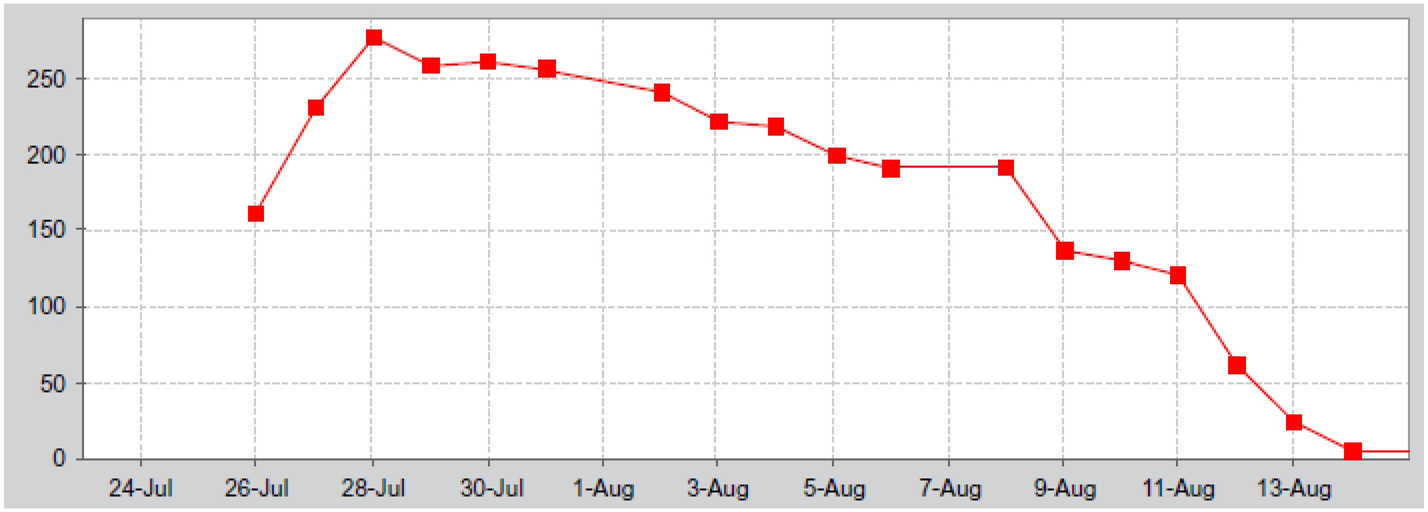
Backlog refinement meeting

In the Backlog Refinement Meeting, the team estimates the amount of effort they would expend to complete items in the Product Backlog and provides other technical information to help the Product Owner prioritize them. Large vague items are split and clarified, considering both business and technical concerns. Sometimes a subset of the team, in conjunction with the Product Owner and other stakeholders, will compose and split Product Backlog Items before involving the entire team in estimation.


User login
S
SSL enable
S
Reset lost password
M
Account lockout after three attempts
S
LDAP integration
M
Register a new login
L
Admin reporting
XL
top items are more granular
only one item at a time is top priority

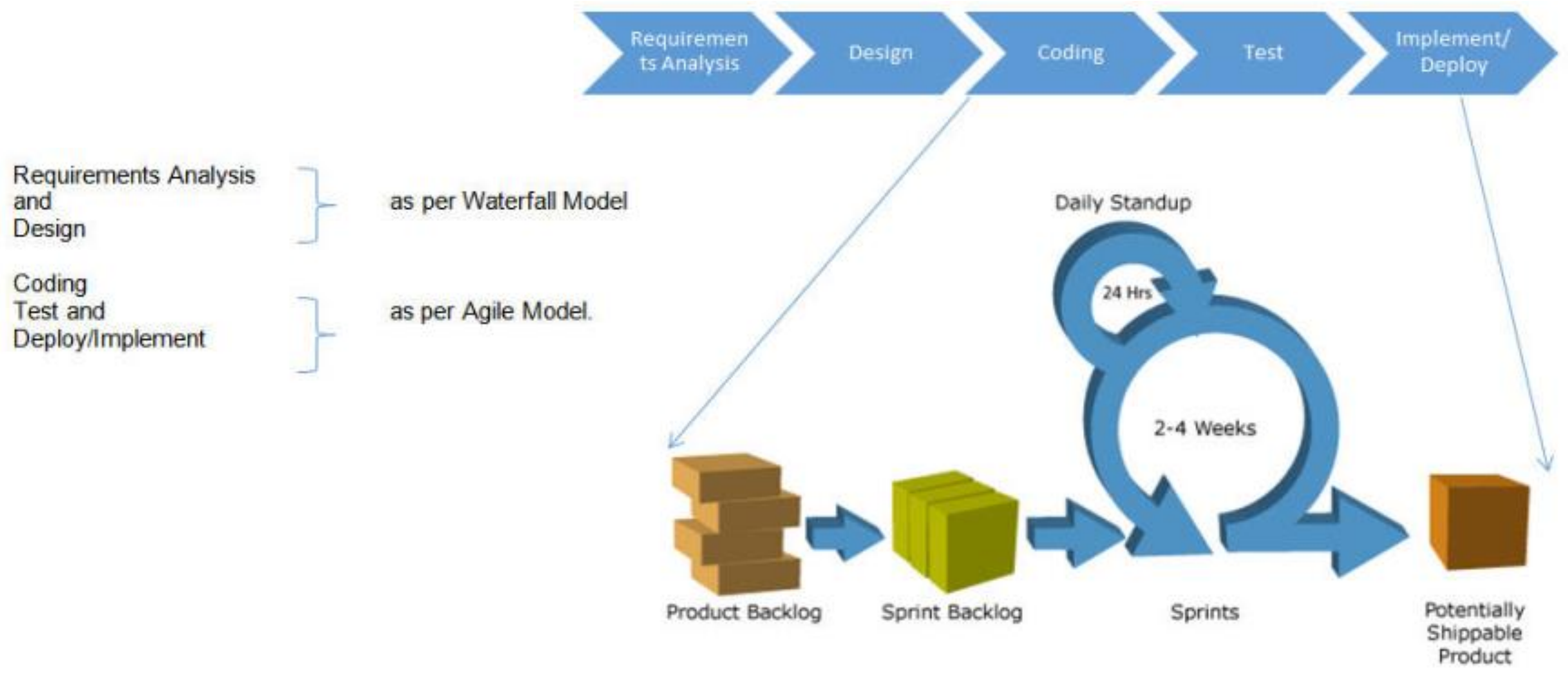
Sprint Burndown Chart

> Indicates total remaining team task hours within one Sprint Re-estimated daily, thus may go up before going down
> Intended to facilitate team self-organizationten
> The ScrumMaster should discontinue use of this chart if it becomes an impediment to team self-organization.



Hybrid/collaborative model

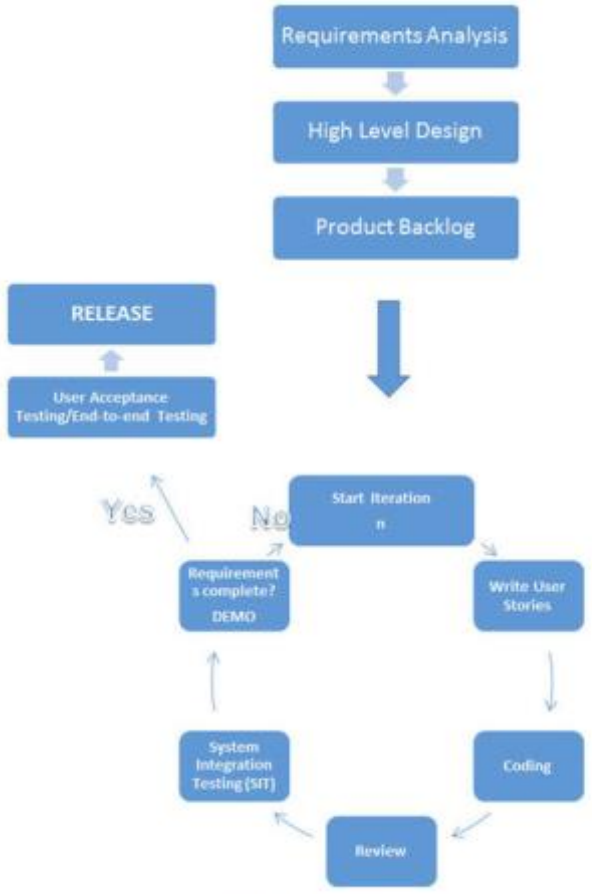
Quellle: http://www.softwaretestinghelp.com/agile-waterfall-hybrid-model/



Collaborative (Hybrid) Model

Hybrid/collaborative model

Quellle: http://www.softwaretestinghelp.com/agile-waterfall-hybrid-model/



The hybrid model

Example: Being an agile project manager as link between Scrum and classical project environment

(Source: Projektmagazin 23/2014 – Claus Kolb)

Introduction

Question for discussion:

Which rolls would you like to take over after SCRUM development has been introduced? In SCRUM, no PM is needed, but suggests that the PM changes to Scrum Master or Product Owner.

That's the ideal SCRUM world. But what can the reality in companies look like?

Challenge:

> SCRUM approach is not introduced in the whole company but only in the development department.
> Project stakeholder are also "non SCRUM" departments

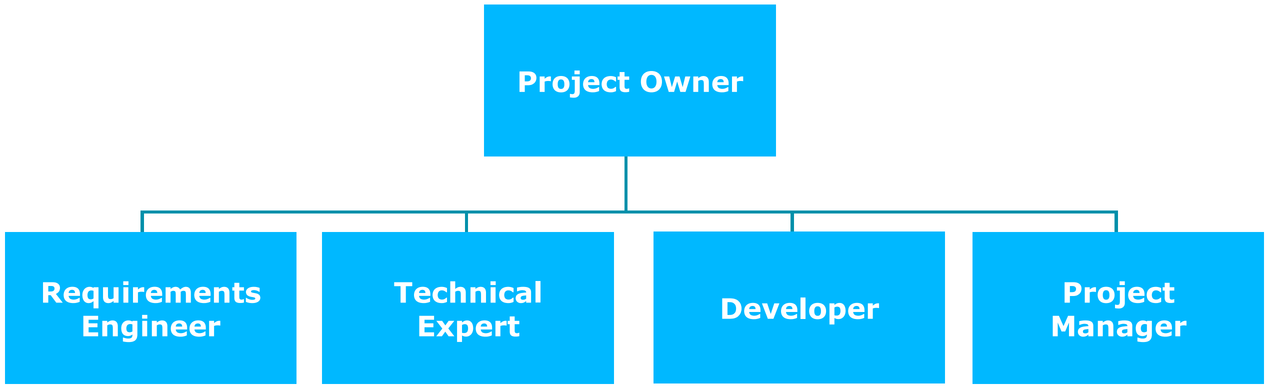
Possible solution:

> Acting as AGILE PROJECT MANAGER

Definition

Classic project managing:

- Sequential approach (Concept, Realization, Test, Rollout)
- Additional Tasks
  - Development of concept documentation
  - Management reporting
  - Budget overview
  - Expectations of specific industry know how and development know how



Definition

“Agile” Project Manager:

An agile project manager, leads a project, in which a part of the team is using an agile approach. In this part of the project, most of the over all project effort is located.

According SCRUM a PM is not needed anymore. In reality, a lot of needed task are still there to be handled by a project manager.

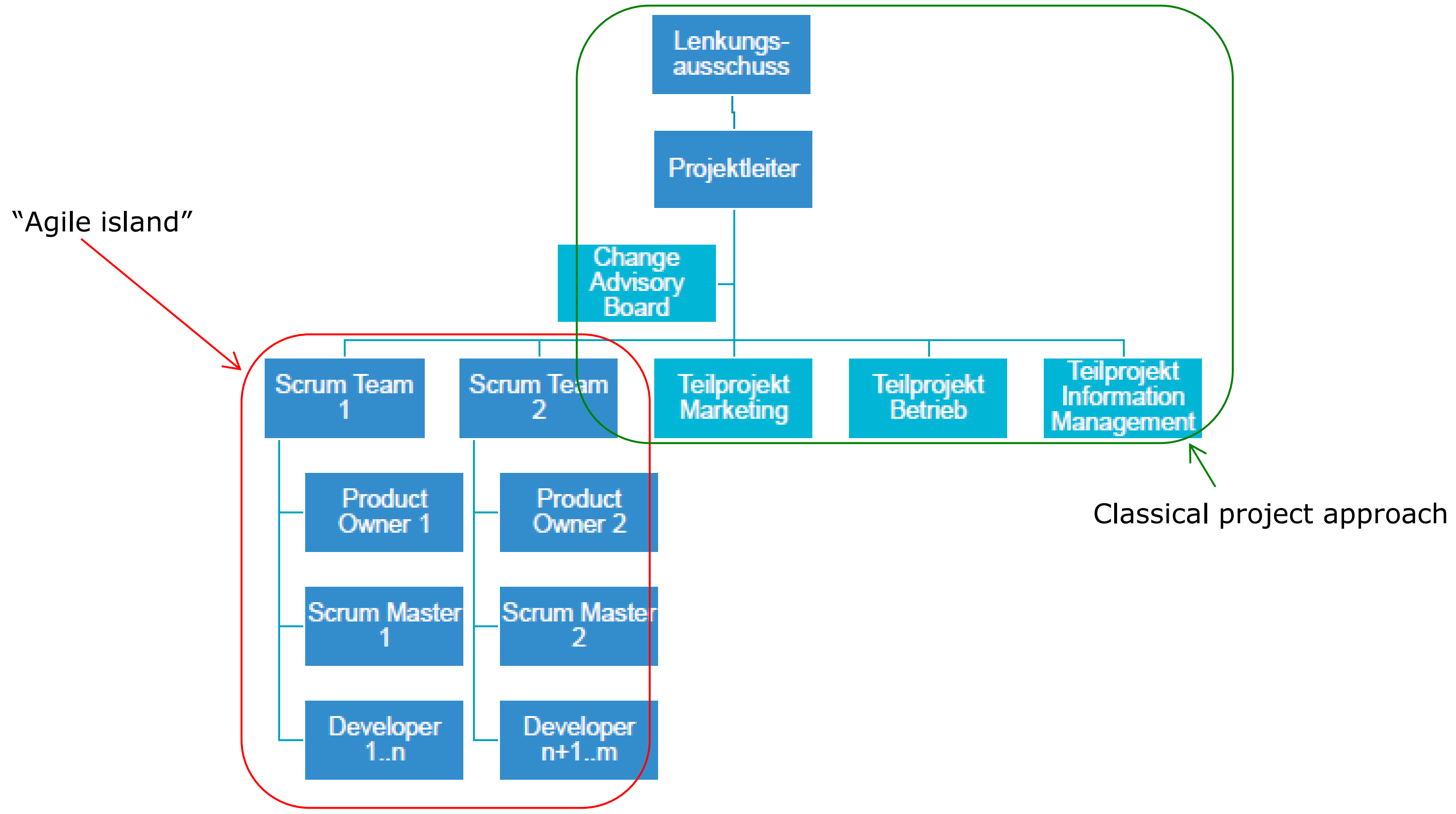
Project example: Enhancement of an online transaction platform

Project goals:

- Additional function of the technical platform (70% of the effort)
  - Development according SCRUM
  - In stable teams – Responsible for different functionalities
- Adaptation of processes and systems other departments (not working with SCRUM)
  - Establishment of a new data warehouse for new clients
  - Enhancement of call center processes and infrastructure

Project example: Enhancement of an online transaction platform

Typical project organization:



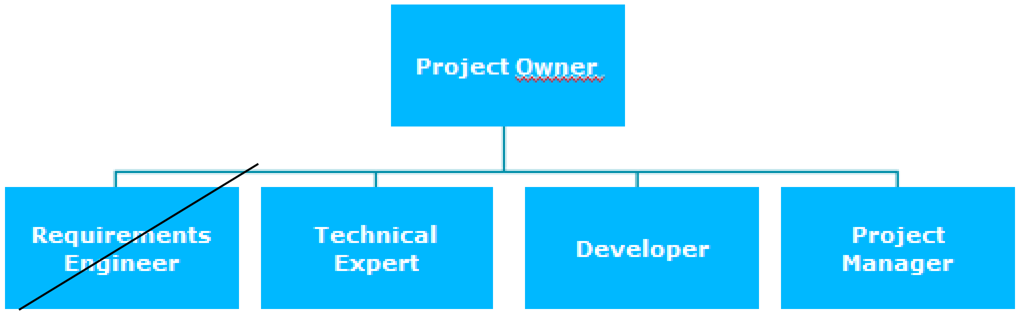
Possible role of a classical PM in SCRUM

Requirements Engineer/ Business Analyst = Product Owner

- > Focus on content expertise
- > Documentation of requirements in epics and user stories
- > Maintenance and prioritization of back log

Result:

By having a dedicated product owner, the PM does not need to get into content expertise. He is needed for moderation in technical and subject specific discussion.



Possible role of a classical PM in SCRUM

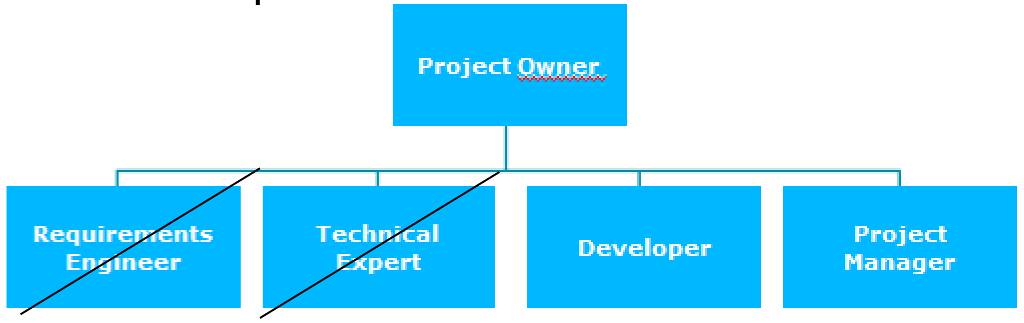
"Technical expert"

Classical approach: Technical know how at software products, programming and infrastructure to be able to define work packages, assign tasks and define dependencies and priorities.

In "agile teams"

Not necessary anymore: developers are responsible for architecture prioritization and implementation.

Work package definition is done by product owner from functionality point of view and done by developers themselves from technical point of view. Involvement of PM in this process is counter-productive.

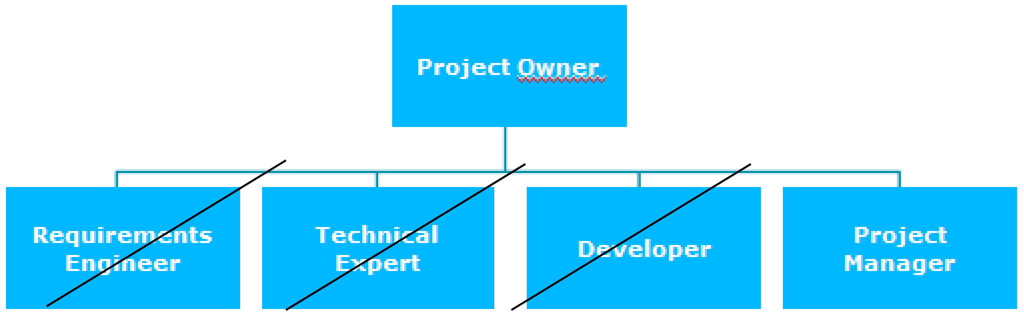


Possible role of a classical PM in SCRUM

"Software Developer"
In SCRUM software is only produced by development team.

"Project Lead" – Who is assigning tasks between SCRUM Master, Product Owner, Teams and agile PM?

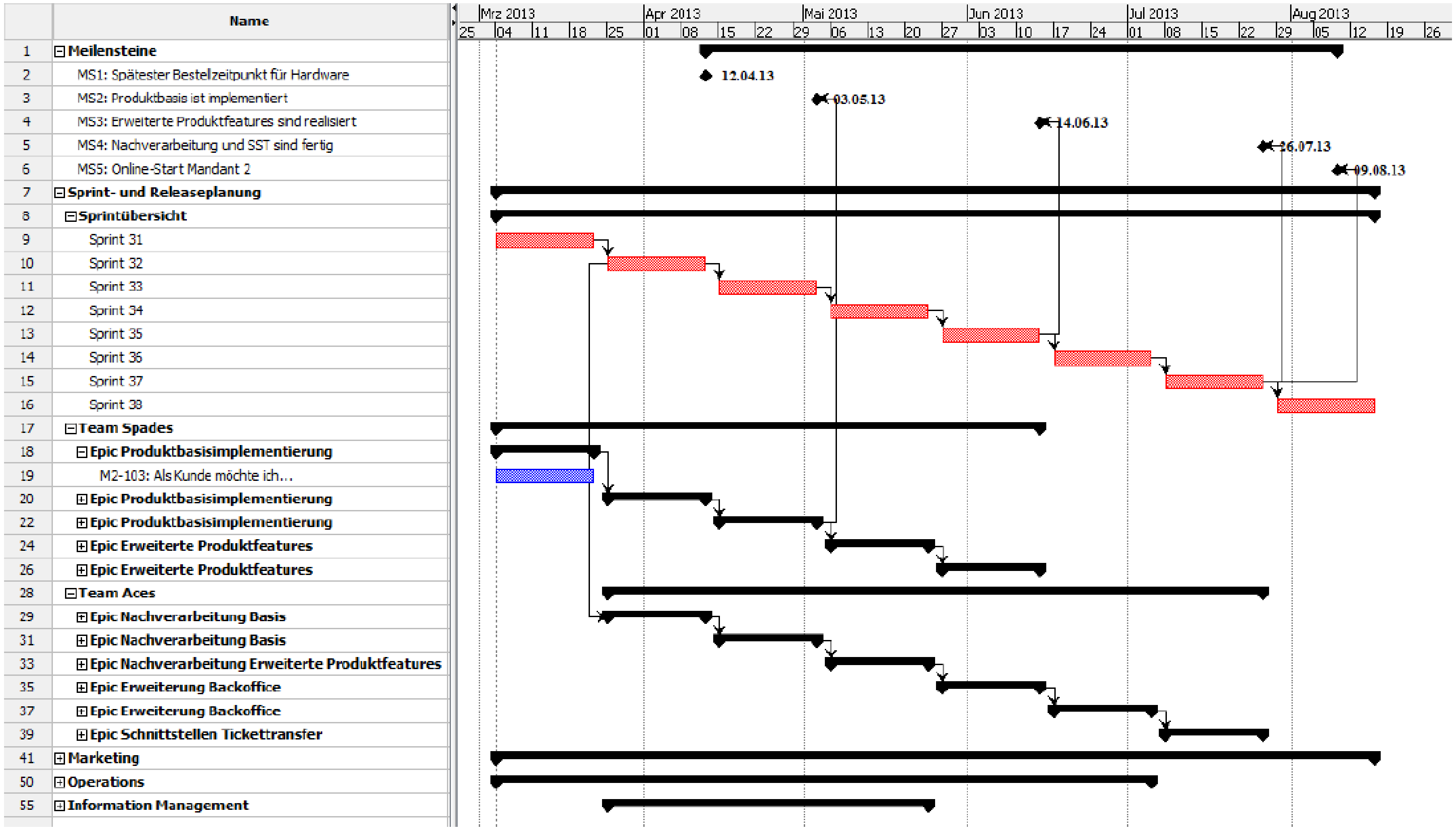
> Initialization
> Method/approach and organization
> Planning
> Coordination agile und "classic islands"
> Controlling
> Change Mgmt.
> Communication/Reporting
> Project Marketing
> Risk Management
> Leadership / Motivation
> Close Out and Review



Cost and effort estimation

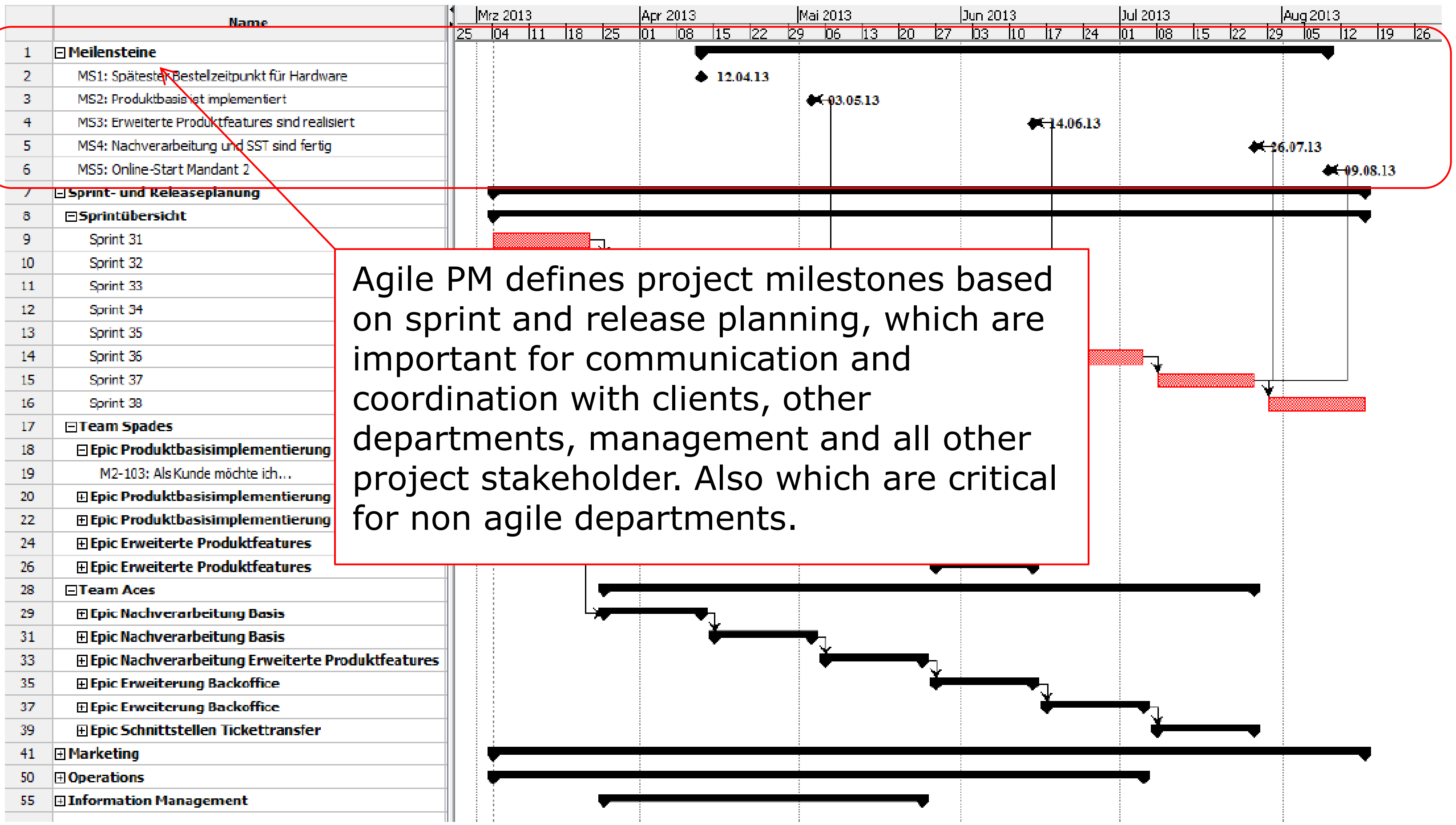
- Outside development– T-Shirt Size
  - XS <= 5PT
  - S <= 10PT
  - M <= 25PT
  - L <= 50PT
  - XL <= 100PT
  - XXL > 100PT
- Estimation based on epics and UserStories. Based on velocity of teams product owner calculate story points into person days.
- Planning Poker in Sprint Planning
- Minimal Viable Product definition
- Needed for feasibility analysis and basis for decision making and for prioritization in portfolio management.

Example sprint-planning



	Name
1	⊟ **Meilensteine**
2	MS1: Spätester Bestelzeitpunkt für Hardware
3	MS2: Produktbasis ist implementiert
4	MS3: Erweiterte Produktfeatures sind realisiert
5	MS4: Nachverarbeitung und SST sind fertig
6	MS5: Online-Start Mandant 2
7	⊟ **Sprint- und Releaseplanung**
8	⊟ **Sprintübersicht**
9	Sprint 31
10	Sprint 32
11	Sprint 33
12	Sprint 34
13	Sprint 35
14	Sprint 36
15	Sprint 37
16	Sprint 38
17	⊟ **Team Spades**
18	⊟ **Epic Produktbasisimplementierung**
19	M2-103: Als Kunde möchte ich...
20	⊞ **Epic Produktbasisimplementierung**
22	⊞ **Epic Produktbasisimplementierung**
24	⊞ **Epic Erweiterte Produktfeatures**
26	⊞ **Epic Erweiterte Produktfeatures**
28	⊟ **Team Aces**
29	⊞ **Epic Nachverarbeitung Basis**
31	⊞ **Epic Nachverarbeitung Basis**
33	⊞ **Epic Nachverarbeitung Erweiterte Produktfeatures**
35	⊞ **Epic Erweiterung Backoffice**
37	⊞ **Epic Erweiterung Backoffice**
39	⊞ **Epic Schnittstellen Tickettransfer**
41	⊞ **Marketing**
50	⊞ **Operations**
55	⊞ **Information Management**

Example sprint-planning

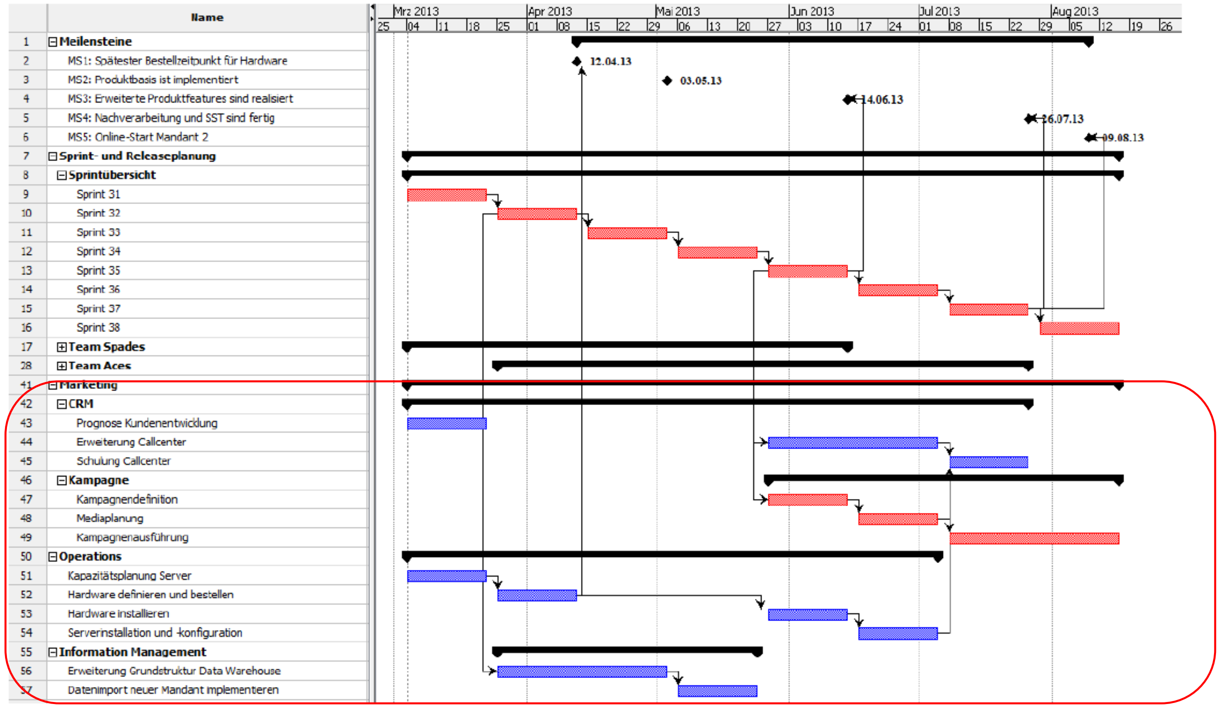


	Name
1	⊟ **Meilensteine**
2	MS1: Spätester Bestelzeitpunkt für Hardware
3	MS2: Produktbasis ist implementiert
4	MS3: Erweiterte Produktfeatures sind realisiert
5	MS4: Nachverarbeitung und SST sind fertig
6	MS5: Online-Start Mandant 2
7	⊟ **Sprint- und Releaseplanung**
8	⊟ **Sprintübersicht**
9	Sprint 31
10	Sprint 32
11	Sprint 33
12	Sprint 34
13	Sprint 35
14	Sprint 36
15	Sprint 37
16	Sprint 38
17	⊟ **Team Spades**
18	⊟ **Epic Produktbasisimplementierung**
19	M2-103: Als Kunde möchte ich...
20	⊞ **Epic Produktbasisimplementierung**
22	⊞ **Epic Produktbasisimplementierung**
24	⊞ **Epic Erweiterte Produktfeatures**
26	⊞ **Epic Erweiterte Produktfeatures**
28	⊟ **Team Aces**
29	⊞ **Epic Nachverarbeitung Basis**
31	⊞ **Epic Nachverarbeitung Basis**
33	⊞ **Epic Nachverarbeitung Erweiterte Produktfeatures**
35	⊞ **Epic Erweiterung Backoffice**
37	⊞ **Epic Erweiterung Backoffice**
39	⊞ **Epic Schnittstellen Tickettransfer**
41	⊞ **Marketing**
50	⊞ **Operations**
55	⊞ **Information Management**

Agile PM defines project milestones based on sprint and release planning, which are important for communication and coordination with clients, other departments, management and all other project stakeholder. Also which are critical for non agile departments.

Planning

> Outside the agile islands, where the PO does the sprint and release planning, the project runs like a classical project. The work packages are defined by the PM.



	Name
1	⊟ **Meilensteine**
2	MS1: Spätester Bestellzeitpunkt für Hardware
3	MS2: Produktbasis ist implementiert
4	MS3: Erweiterte Produktfeatures sind realisiert
5	MS4: Nachverarbeitung und SST sind fertig
6	MS5: Online-Start Mandant 2
7	⊟ **Sprint- und Releaseplanung**
8	⊟ **Sprintübersicht**
9	Sprint 31
10	Sprint 32
11	Sprint 33
12	Sprint 34
13	Sprint 35
14	Sprint 36
15	Sprint 37
16	Sprint 38
17	⊞ **Team Spades**
28	⊞ **Team Aces**
41	⊟ **Marketing**
42	⊟ **CRM**
43	Prognose Kundenentwicklung
44	Erweiterung Callcenter
45	Schulung Callcenter
46	⊟ **Kampagne**
47	Kampagnendefinition
48	Mediaplanung
49	Kampagnenausführung
50	⊟ **Operations**
51	Kapazitätsplanung Server
52	Hardware definieren und bestellen
53	Hardware installieren
54	Serverinstallation und -konfiguration
55	⊟ **Information Management**
56	Erweiterung Grundstruktur Data Warehouse
57	Datenimport neuer Mandant implementieren

Planning – Best Practices

> Adoption of work package timelines to rhythm of sprints and releases (2-3 weeks

> Copying of sprint and release plan into the project plan and assigning dependencies with classical work packages.

> Detailed coordination within a sprint is task of SCRUM team members. PM acts as coordinator and moderator

> Working with fixed SCRUM team members and if possible constant capacity.

> PM asks department heads for capacity at non SCRUM departments according sprint needs.

Coordination of agile and non-agile departments

- Sub projects managed by sub project PMs
- Coordination between agile PM and department heads about needed capacity.
- Progress controlling in work packages by agile PM
- Identification and tracking of dependencies between agile and non agile departments.

Project Controlling - Costs

In SCRUM scope of services is not fixed. Time and budget is.

The cost per story point develop based on velocity and cost of personal in the development team.

Product Owner has the responsibility of assigning the budget smart to user stories.

In case of budget issues and still open and needed story points the agile PM manages a change request.

Project Controlling – Performance and Quality

- Classical project parts– Project manager
- Agile islands – Development team with Product Owner in Sprint Review Meetings.

Best Practice:

- Sprint Review at morning time. Little issues can be solved before the sprint and the whole US does not needed to be taken to the next sprint.
- When there are bigger issues, the solving of those can be done until the next milestone. A fixed capacity for bug fixing makes sense.

Project Controlling - Time

> Covered by roles:
  - PM = project phases and milestones
  - Product Owner = Sprint and Release planning
  - SCRUM Master and Dev = Keep sprints in time.

Change management

In agile part of the project:

No classical change requests. Because changes are integral part of the agile process. Product Owner needs to keep the big picture and overview.

If the PO realizes problems, he can request a change request from the agile PM. Done by a steering committee.

In our example there is a „Change Advisory Board" to consult the agile PM technically and functionally and show consequences of changes.

Members: Development, PO and PM

Communication and reporting

> SCRUM Master → Progress of SCRUM teams
> Agile PM → Puts together sprint reporting, release planning and status of non agile project parts for management overview.

Project marketing

Product Owner keeps product vision. Agile PM is responsible for project and project marketing.

Risk management

Agile Project manager keeps big picture and is responsible for risk

Leadership

Assigning work packages:

Not needed in SCRUM . Developers take their own user stories from the back log. Requirements only result from the sprint planning and release planning.

In classical part, the PM assigns work packages. Or if there is a weak matrix organisation, the department head of classical departments

Conflict- and Motivation management

In agile part only by the SCRUM Master

In classical part by the agile PM . Also improving the project approach itself.

Project close out and review

In agile part done by sprint retrospectives. Organised and moderated by SCRUM Master.

In classical part by agile PM.

Recommended a project-wide retrospective meeting every 3 sprints organised by agile PM. (agile and non agile participants)

Other best practice for agile methods

Combination between agile method and the KANBAN production structure (KANBAN 1947 – Toyota Motor Corporation)

What is KANBAN?

Source: http://kanbanblog.com/explained/

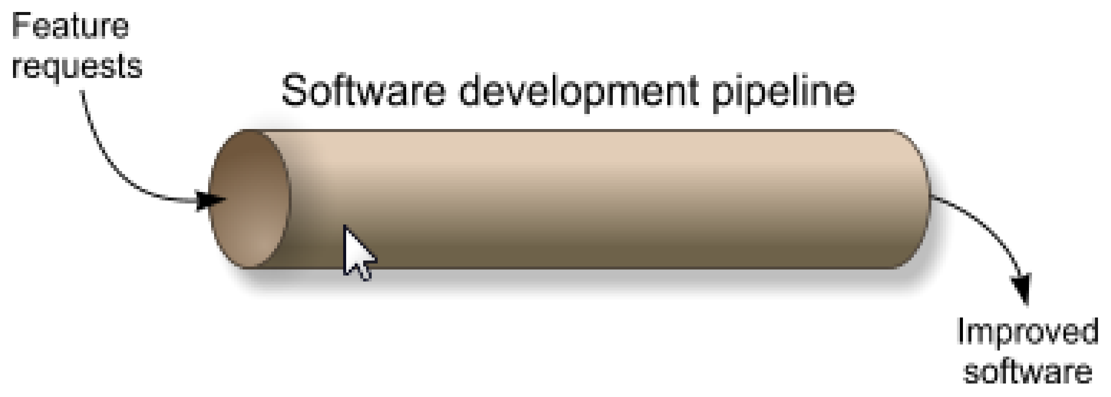
>Kanban is a new technique for managing a software development process in a highly efficient way. Kanban underpins Toyota's "just-in-time" (JIT) production system. Although producing software is a creative activity and therefore different to mass-producing cars, the underlying mechanism for managing the production line can still be applied.

>A software development process can be thought of as a pipeline with feature requests entering one end and improved software emerging from the other end.

>Inside the pipeline, there will be some kind of process which could range from an informal ad hoc process to a highly formal phased process. In this article, we'll assume a simple phased process of: (1) analyse the requirements, (2) develop the code, and (3) test it works.

What is KANBAN?

Source: http://kanbanblog.com/explained/



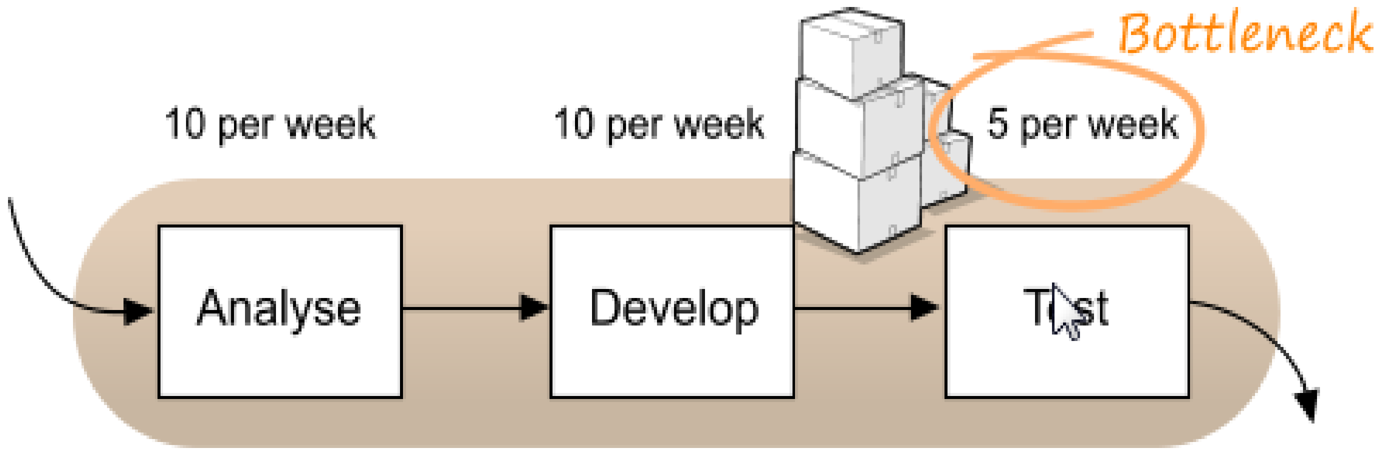
Whats the problem about pipes? And things rushing through?

Bottlenecks…

A bottleneck in a pipeline restricts flow. The throughput of the pipeline as a whole is limited to the throughput of the bottleneck.

What is KANBAN?

Source: http://kanbanblog.com/explained/

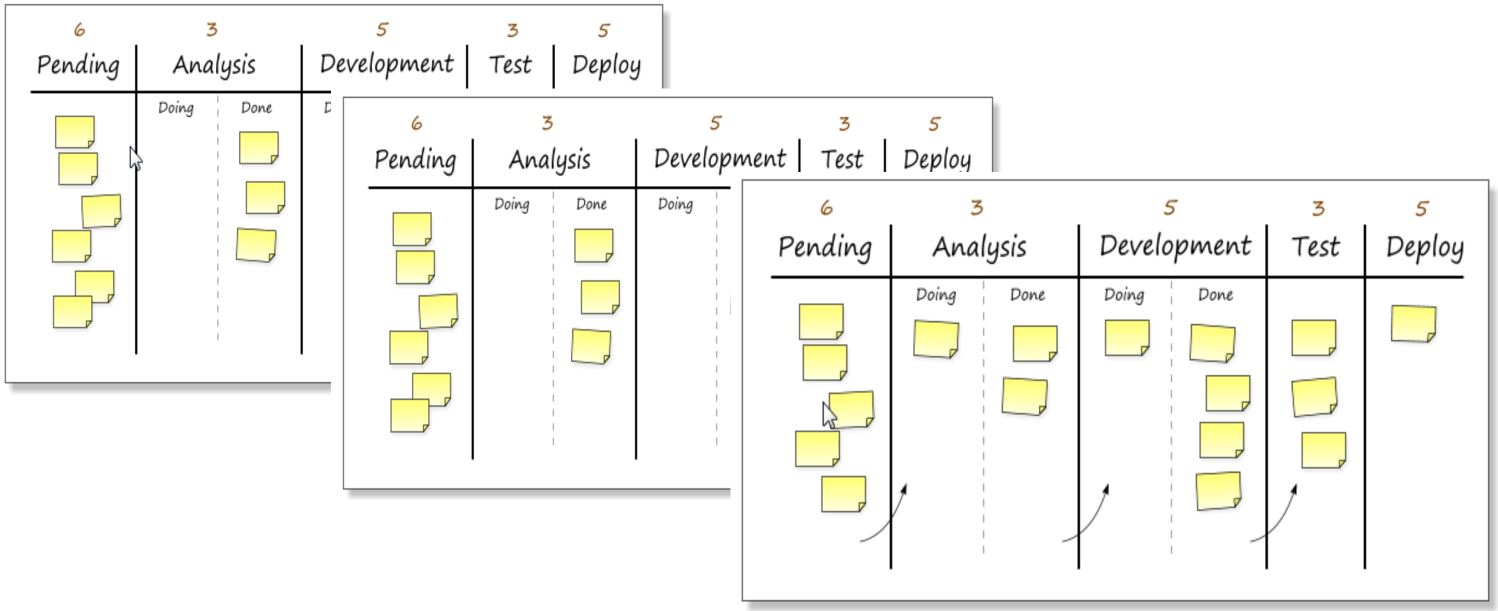


Inevitably, quality suffers. To keep up, the testers start to cut corners. The resulting bugs released into production cause problems for the users and waste future pipeline capacity.

Kanban reveals bottlenecks

Source: http://kanbanblog.com/explained/

The board below shows a situation where the developers and analysts are being prevented from taking on any more work until the testers free up a slot and pull in the next work item. At this point the developers and analysts should be looking at ways they can help relieve the burden on the testers.



Kanban Production system principles

Source: http://leankit.com/kanban/what-is-kanban/

1. Visualize Work

By creating a visual model of your work and workflow, you can observe the flow of work moving through your Kanban system. Making the work visible—along with blockers, bottlenecks and queues—instantly leads to increased communication and collaboration.

2. Limit Work in Process

By limiting how much unfinished work is in process, you can reduce the time it takes an item to travel through the Kanban system.

3. Focus on Flow

By using work-in-process (WIP) limits and developing team-driven policies, you can optimize your Kanban system to improve the smooth flow of work

4. Continuous Improvement

Once a Kanban system is in place, it becomes the cornerstone for a culture of continuous improvement. Teams measure their effectiveness by tracking flow, quality, throughput, lead times and more.

How to get started with KANBAN?

1.Map your value stream (your development process).

Where do feature ideas come from? What are all the steps that the idea goes through until it's sitting in the hands of the end-user?

2.Define the start and end points for the Kanban system.

These should preferably be where you have political control. Don't worry too much about starting with a narrow focus, as people outside the span will soon ask to join in.

3.Agree:

> Initial WIP limits and policies for changing or temporarily breaking them
> Process for prioritizing and selecting features
> Policies for different classes of service (e.g. "standard", "expedite", "fixed delivery date"). Are estimates needed? When choosing work, which will be selected first?
> Frequency of reviews

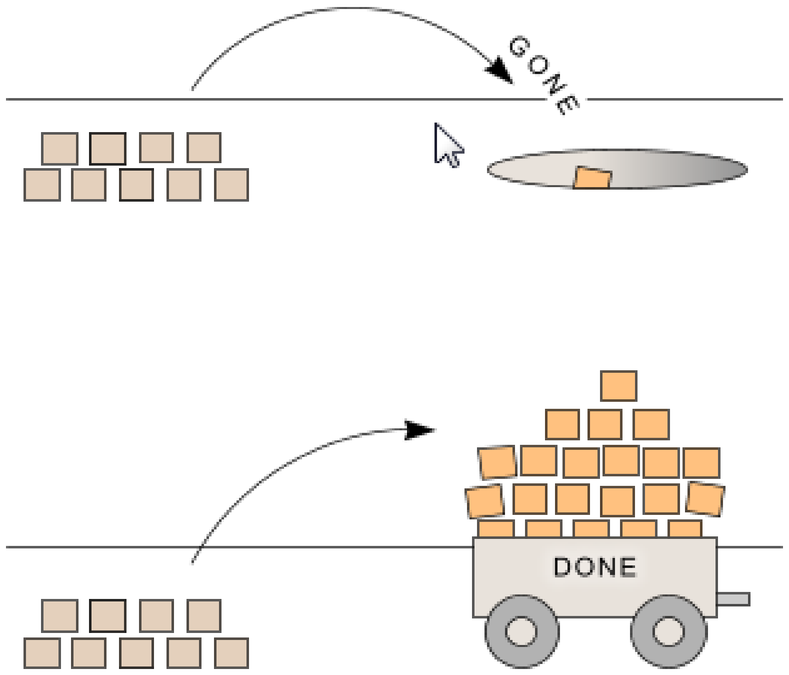
4.Draw up a Kanban board.

All you need is a whiteboard and some Post-It™ notes. Don't spend too much time making it look beautiful because it will almost certainly evolve.

5.Start using it.

Kanban Tips

Seeing progress is motivating



Kanban Tips

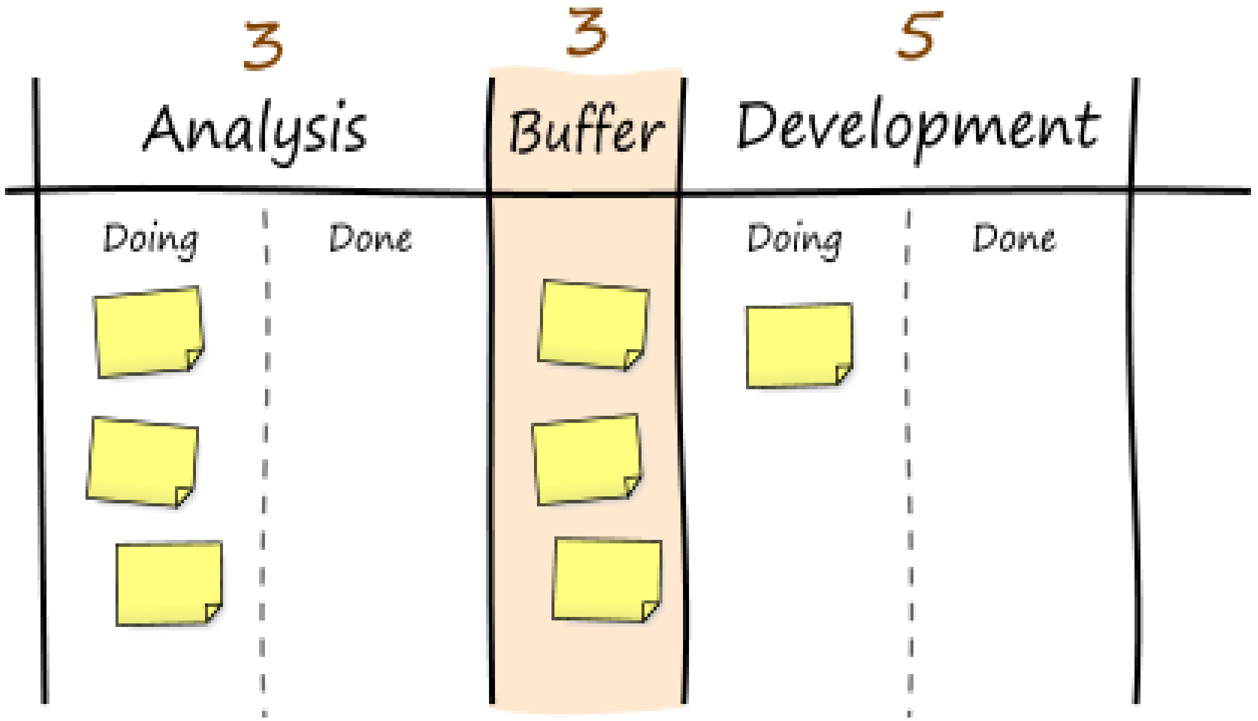
Buffer your bottlenecks

Since the bottleneck is so critical, one of the things we want to do is make sure that it rarely (preferably never) runs out of work. You might wonder how a bottleneck can run dry, when, by definition, it is fed by a wider pipe? The thing to remember is that in our kanban system we're purposely limiting work-in-progress (WIP), and work items vary in size; so a couple of large items being processed upstream could end up temporarily starving the bottleneck.

The solution is to place a buffer in front of the bottleneck, as per the diagram below. In this example, the development team is the bottleneck so a buffer with a limit of 3 work items has been inserted immediately before it (the numbers at the top of the columns are the limits).

Kanban Tips

Buffer your bottlenecks



Kanban Tips

Limit your WIP (work in progress)

Without specific policies to limit WIP, people will naturally tend to multi-task. Non-bottlenecks will start new tasks whenever they're waiting for a bottleneck to complete. The effect is a continually growing pile of partially-completed work. Since the bottleneck limits the throughput of the system, the extra WIP doesn't actually result in more throughput; it just pushes up lead-times.

Kanban Tips

<u>Time at the bottleneck is what counts</u>

Which of these feature ideas should we select?

Feature	Estimated Value	Estimated Production Time
A	$100,000	40 hrs
B	$80,000	40 hrs
C	$60,000	40 hrs

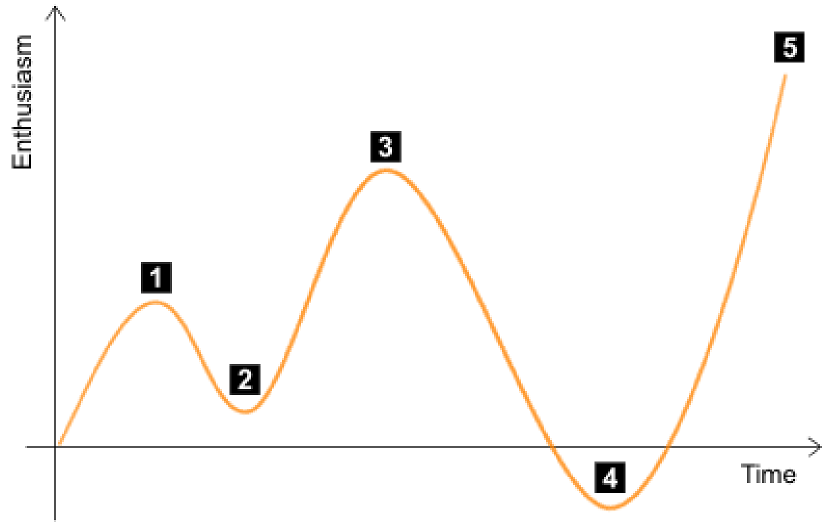
Feature A? It's a no-brainer, right?

Feature	Estimated Value	At the Bottleneck	Away from the Bottleneck	Total Production Time
A	$100,000	**20 hrs**	20 hrs	40 hrs
B	$80,000	**10 hrs**	30 hrs	40 hrs
C	$60,000	**10 hrs**	30 hrs	40 hrs

In this case, for the same 20 hours at the bottleneck we can produce both B and C for a total of $140,000, compared with A at $100,000.

Kanban Tips

Curve your Enthusiasm (by Ray Immelman)



1 "OK, let's give it a whirl!"

2 "It's harder than it looks."

3 "It seems to be working!"

4 "We're reaching the point of no return. Do we really want to do this forever?"

5 "Hey, we're starting to get really good at this."

KANBAN for Software Development Projects?

Why could be KANBAN a bad choice for software development projects?

David Anderson, the creator of The Kanban Method: "Kanban is NOT a software development life cycle or project management methodology! It is not a way of making software or running projects that make software. It is actually not possible to develop with only Kanban. The Kanban Method by itself does not contain practices sufficient to do product development.

KANBAN for Software Development Projects?

Kanban is modeled more after the assembly line and manufacturing. Scrum is modeled more after creative product design.

Which do you think more closely resembles software development? Franz and Herbert on the assembly line at the Magna? Or the group of NASA engineers on the ground who saved the lives of the Apollo 13 astronauts by coming up with a creative solution to a problem within a time-box? We shouldn’t think about software rolls off of an assembly line

KANBAN for Software Development Projects?

Software Development is about empirical processes

Think of yourself making a pot of soup from scratch, without a recipe. Think about all of the "taste-tweak ingredients-taste" experiments(feedback loops) you would need to get a pot of soup that tastes good.

Scrum has the frequent feedback loops built in, for a variety of audiences(Dev Team, Product Owner, Stakeholders, Users) , and for a variety of topics(process-Sprint Retro, product-Ordered Product Backlog, product-Sprint Review, product-Valuable/Releasable Increments). Kanban has no such built in loops, but again, that's because it wasn't designed for software development!

KANBAN for Software Development Projects?

From a Complexity Science view, Kanban is for ‘complicated’ work while Scrum is for “complex” work.

‘Complicated’ work is best solved via ‘good practice’ and ‘experts’ who can find ’cause and effect’ fairly easily. think of an the IT support person who sets up your computer or trouble shoots it. Yes, you need an expert to solve these problems, and the vast majority of the time, the steps to solve these kinds of problems are fairly consistent and repeatable. They are not exactly repeatable, just mostly repeatable.   If the steps were exactly repeatable then they would fall into the ‘Simple’.

‘Complex’ work is best solved via ‘safe to fail experiments’ and one can only ascertain cause and effect after the fact. Each Sprint in Scrum is a ‘safe to fail’ experiment because, while the Sprint increment is always releasable, the business side of the house makes the decision on whether it is safe/valuable to release it or not. In the case of an increment that is un-safe, the team course corrects and comes back with an increment the next sprint that is hopefully safe or more-safe. These safe to fail experiments can be repeated over and over again until it’s time to release the increment.

Conclusion: When to use KANBAN?

Proposal: software team is doing XP, Scrum, Crystal, Waterfall, RUP, DSDM, FDD, etc, then you can layer Kanban on top of it to help find bottlenecks and waste.

Or continues improvement of software (new versions), Bugfixing,…