Génie logiciel

Philippe Dugerdil

26.09.2019

Rappels

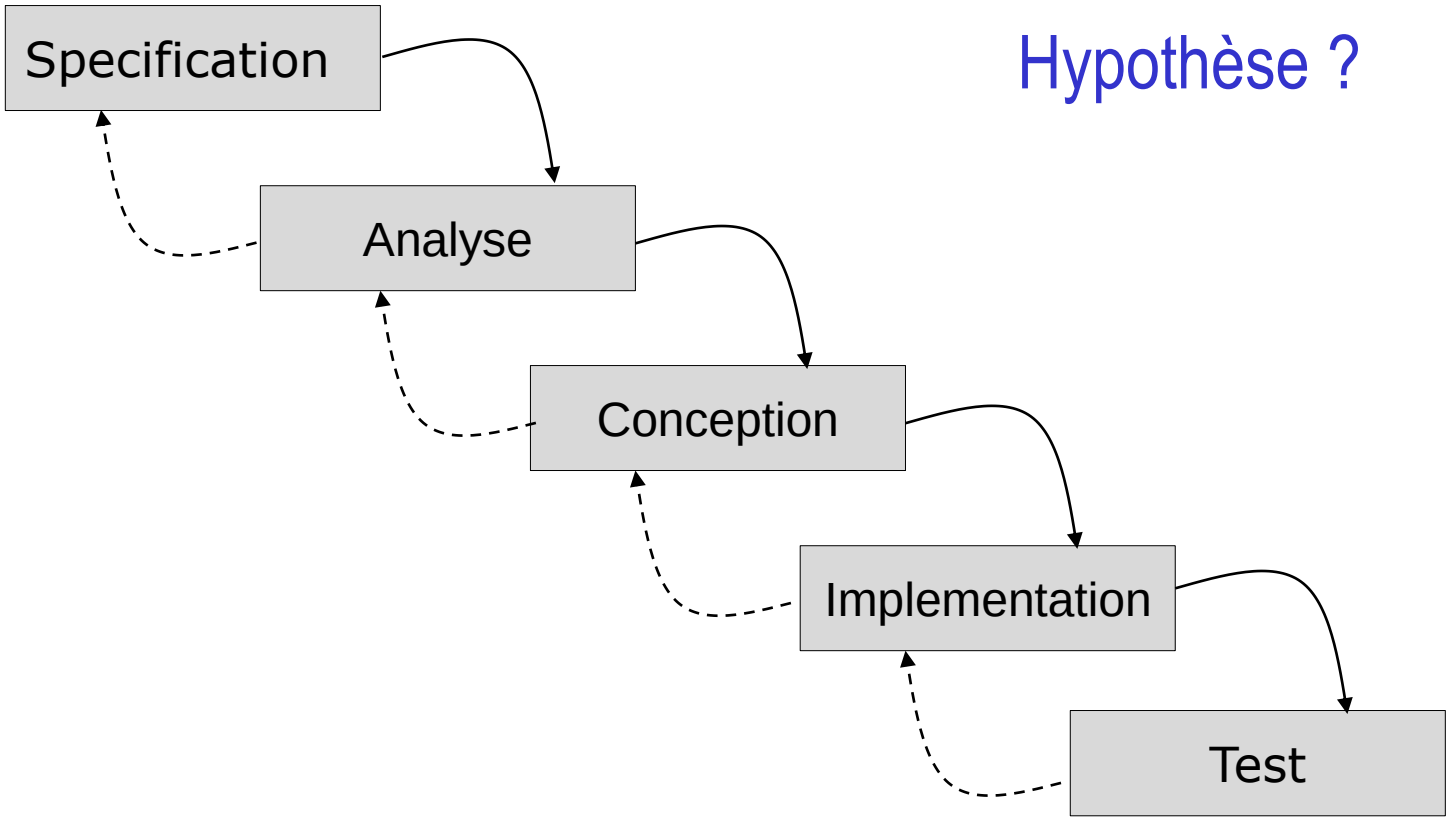
- Le cout du logiciel
- Lois de Lehmann
- Complexité selon Simon
- Couplage et cohesion
- Pyramide des specs et doc de vision
- Besoin d'un processus de developpement
- Problème des specifications

Modèles de processus

Temps

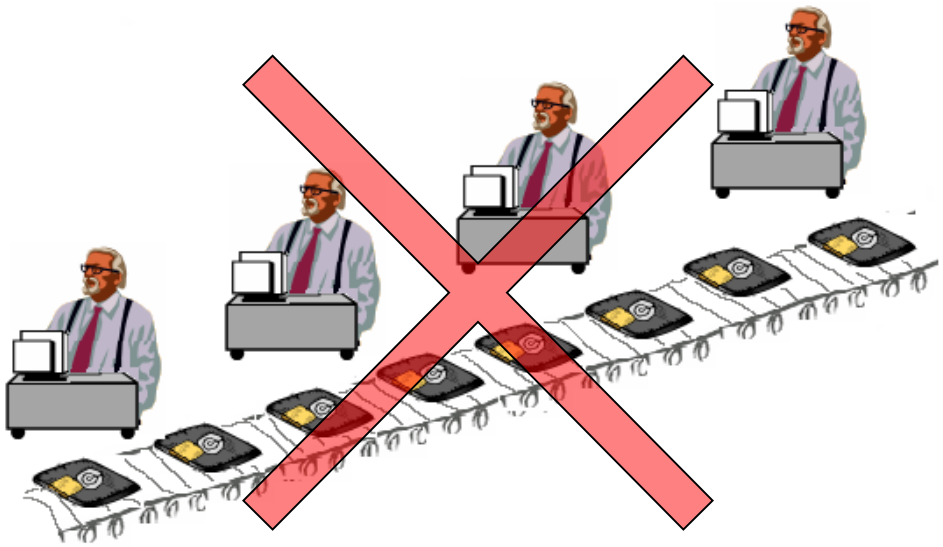
- Waterfall 70's
- Itératif et incrémental (**RUP**) 1998
- Agile (XP, **SCRUM**, AM,..) 1995-2000
- **DAD** 2012

Modèle Waterfall



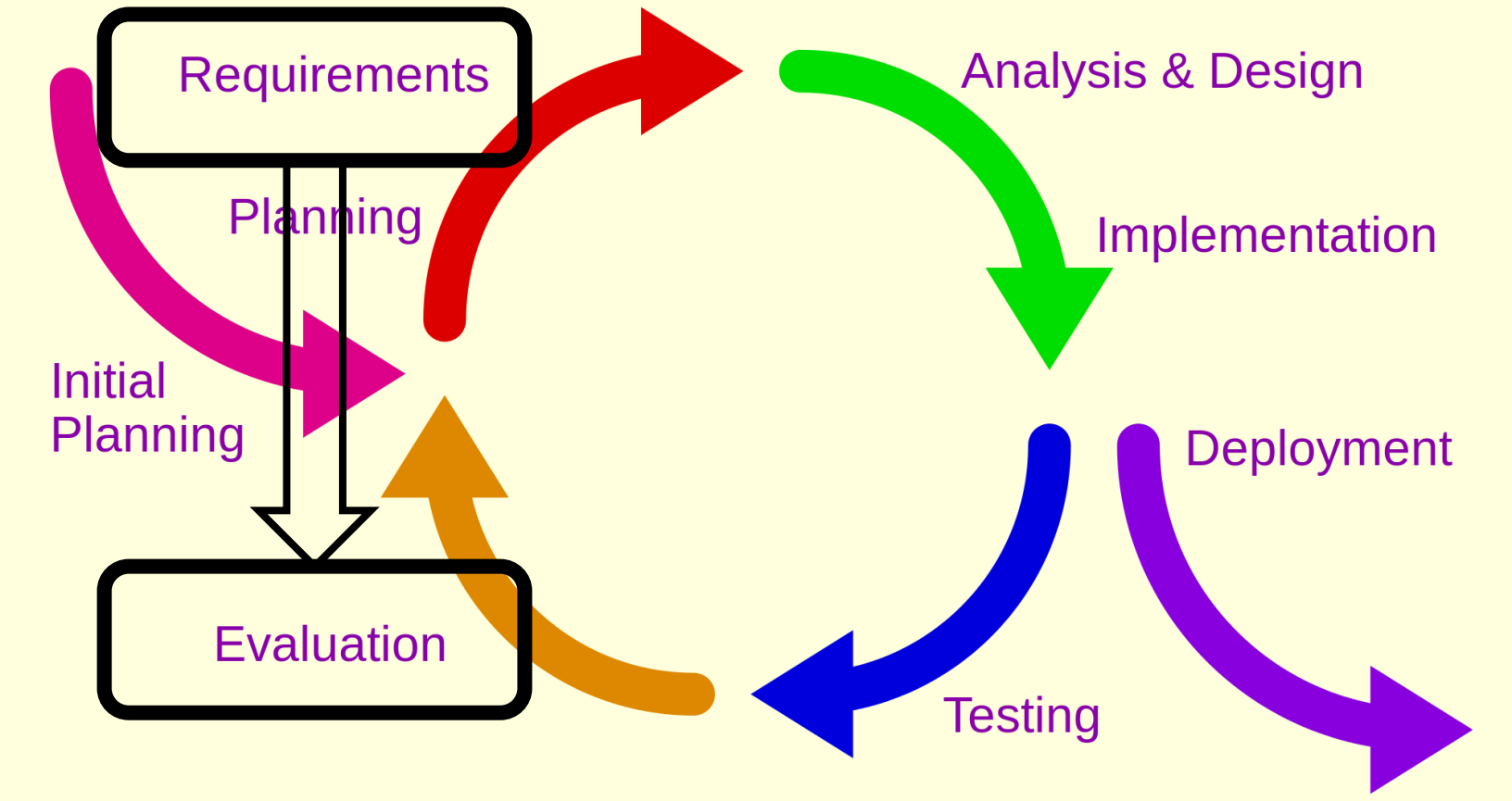
Hypothèse ?

Origine: processus industriels

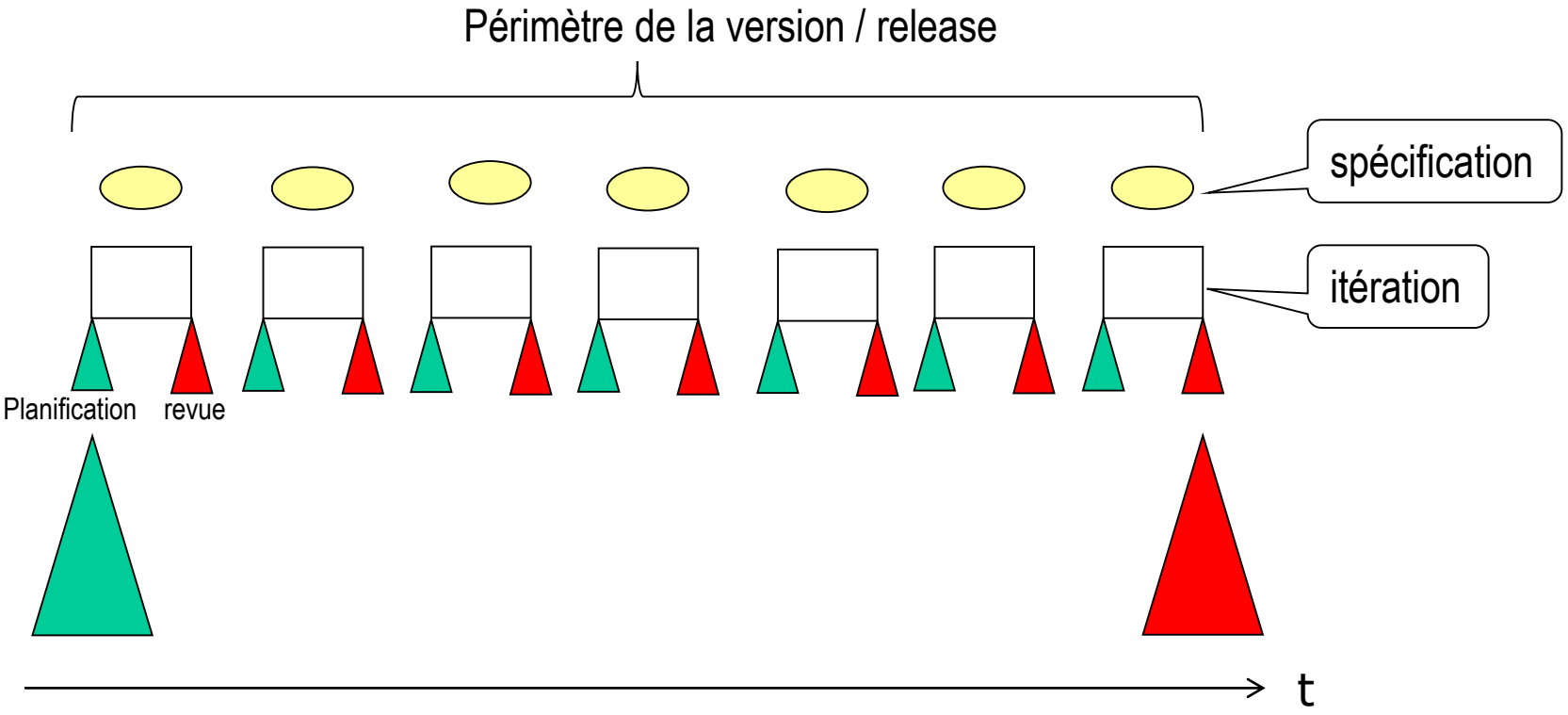


Mais cela ne marche pas (spéc. incrémentales)

Processus itératifs et incrémentaux



Impact sur la planification du projet

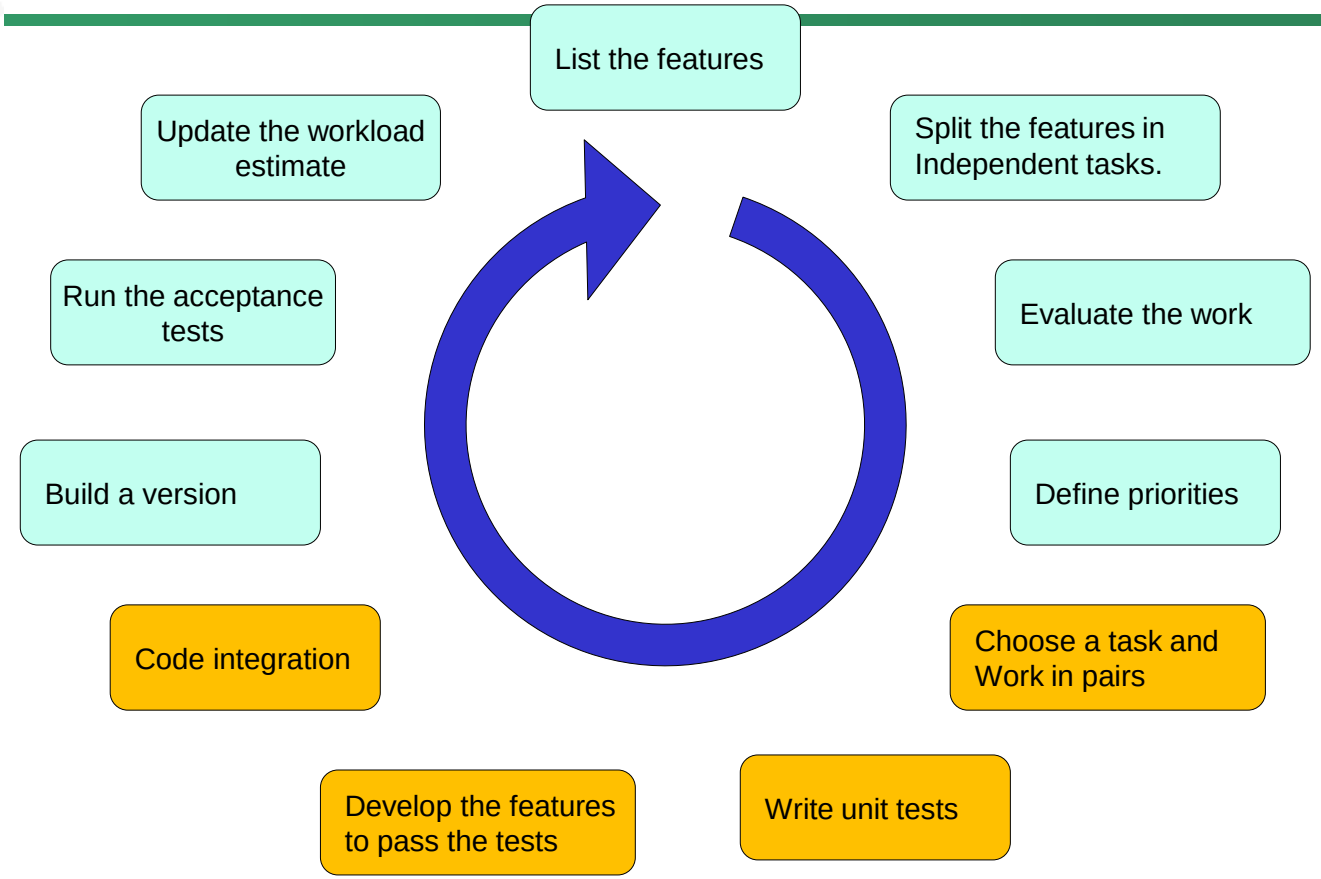


La séquence est réarrangée au fur et à mesure de l'avancement du projet

Agile processes

- Why ?
- February 2001 : Unification of agile principles
  - Agile values
  - Agile manifesto
- Deepening of agile methods
  - XP, SCRUM, AM, AUP, Crystal Clear, …

The forerunner: XP (Extreme Programming)



- Focus on the business value
- Eliminate waste (non code artifacts)

Agile values

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individual and interactions** over process and role
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

www.agilemanifesto.org

Agile principles

www.agilemanifesto.org

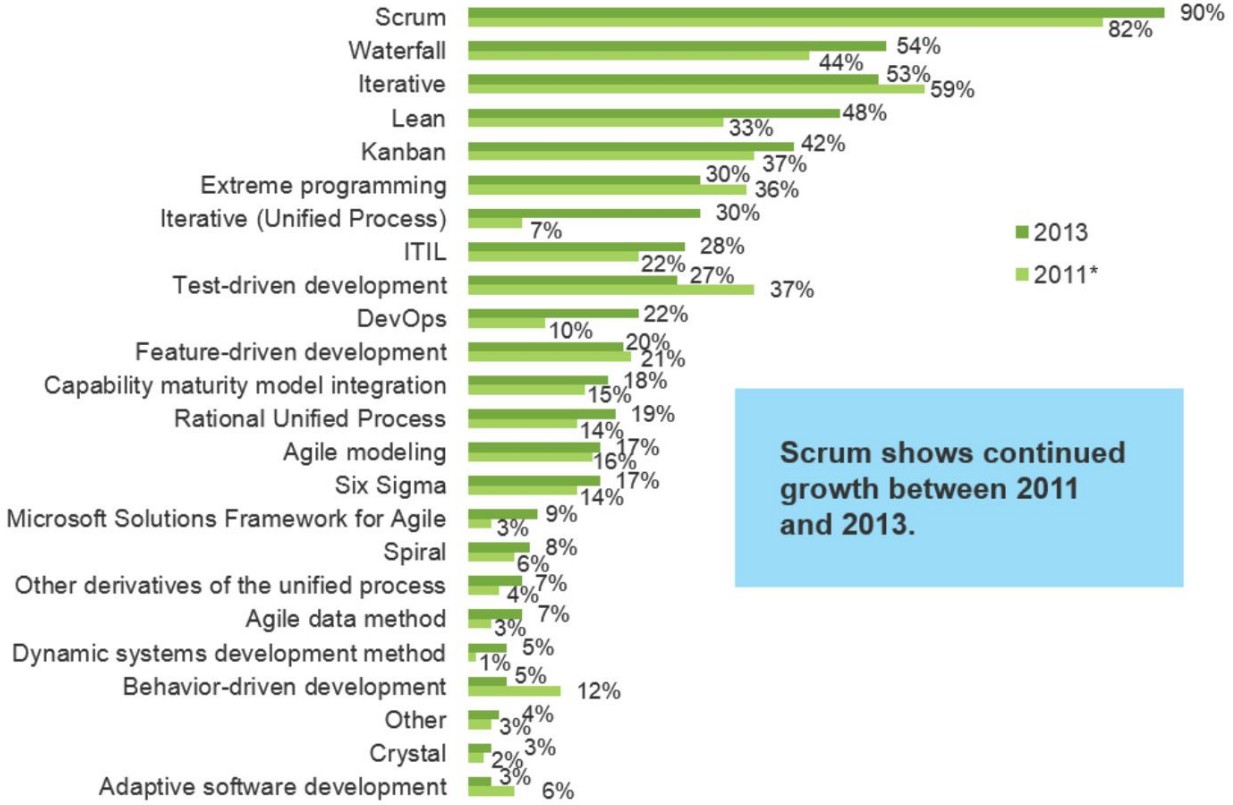
1-6

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
5. Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
6. La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.

7-12

7. Un logiciel opérationnel est la principale mesure d'avancement.
8. Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
9. Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité, c'est-à-dire l'art de minimiser la quantité de travail inutile, est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
12. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence

Scrum, iterative, and waterfall are the most common approaches (2013)

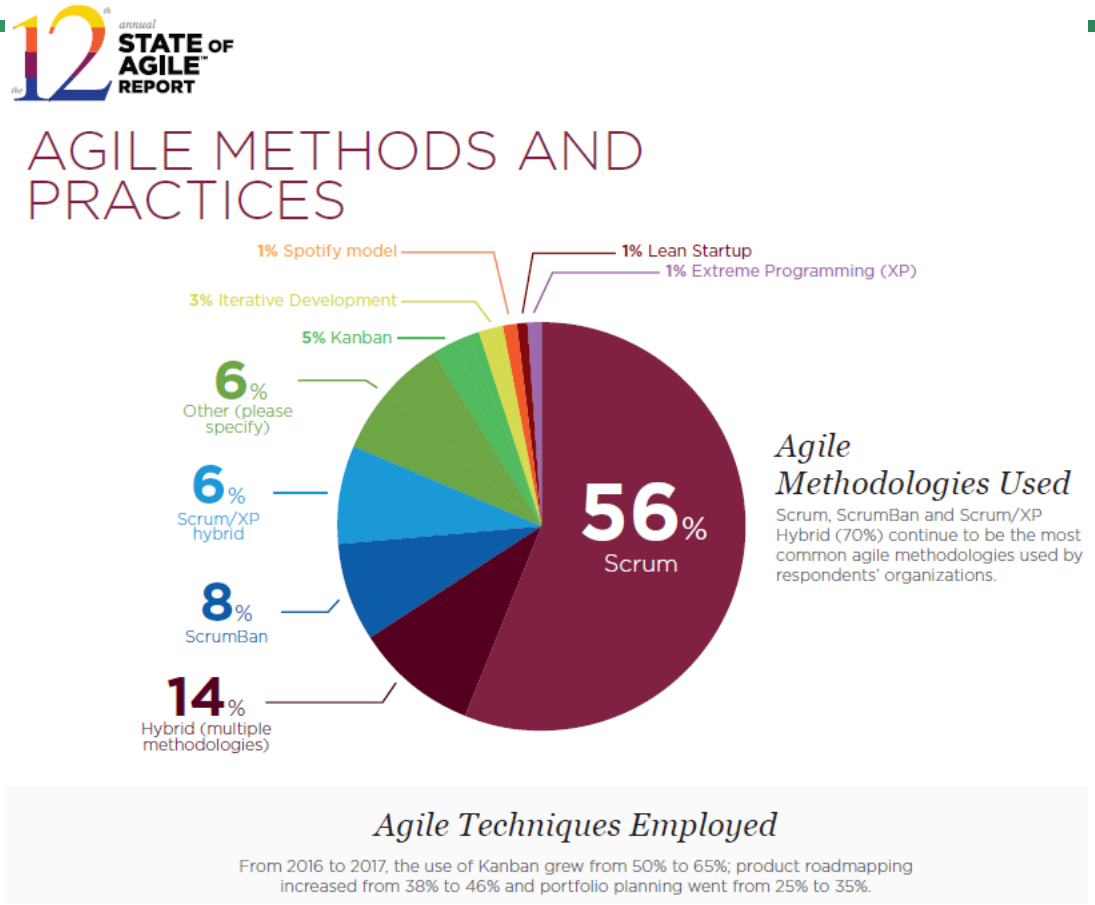


Base: 149 IT professionals from organizations that are planning to implement or have implemented Agile;
*205 IT professionals from organizations that are planning to implement or have implemented Agile
Source: Q3 2013 Global Agile Software Application Development Online Survey;
*November 2011 Global Agile Software Application Development Online Survey

Source: Diego Lo Giudice - What Are Companies Doing To Scale Agile Development?. Forrester Research. October 28, 2013

Private survey 2018: SCRUM !

(undocumented methodology)



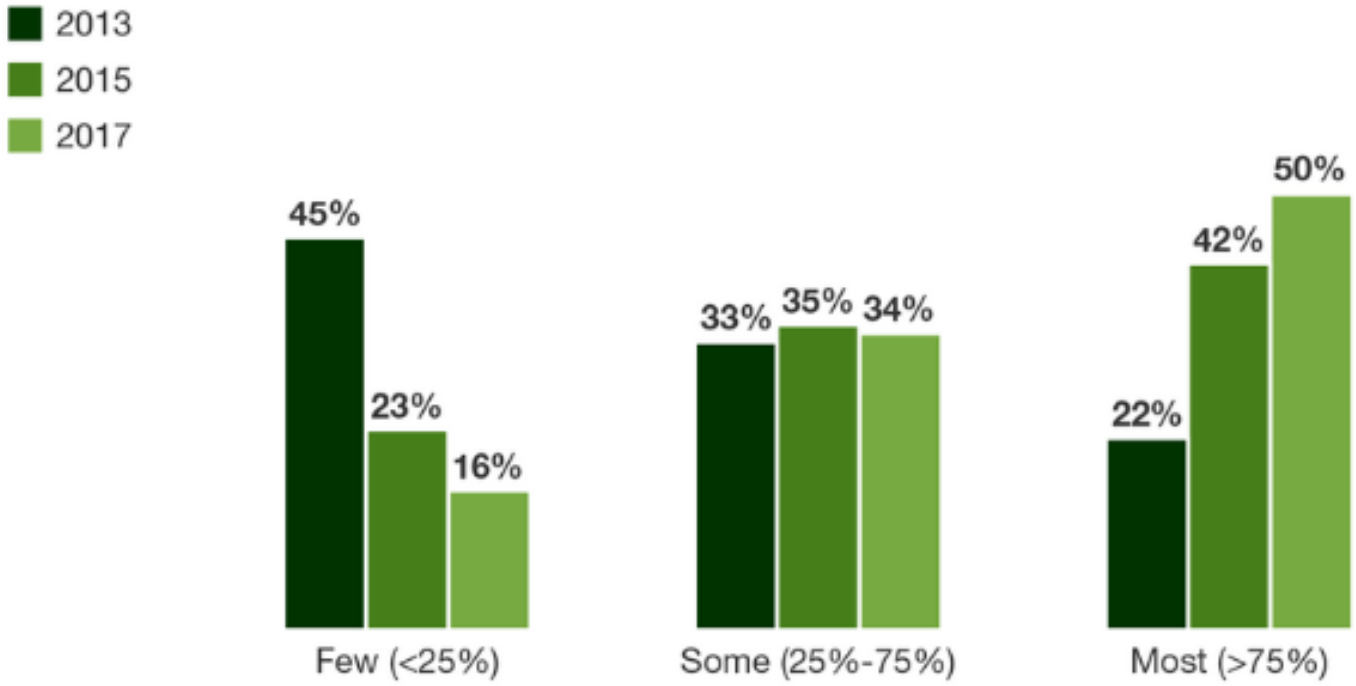
Source: https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report

Company: collab.net (solution provider) https://en.wikipedia.org/wiki/CollabNet

Agile Adoption Trends 2013 To 2017

The State Of Agile 2017: Agile At Scale

"Roughly what percent of development teams in your organization are using Agile practices?"

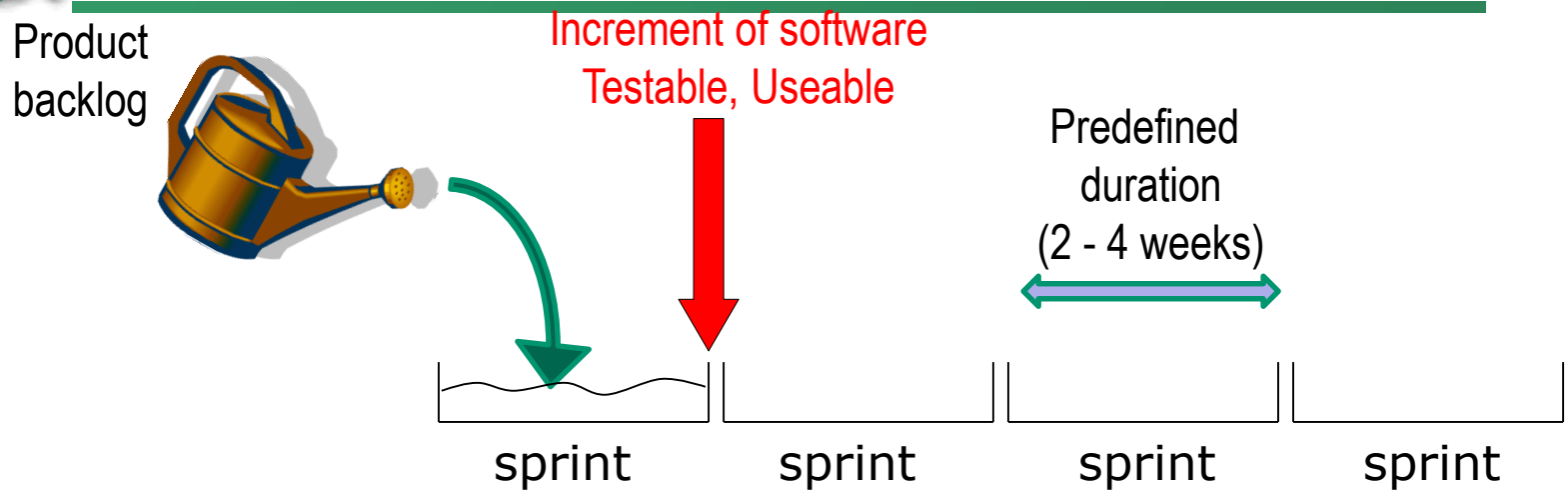


Base: 182 (2013), 149 (2015), and 232 (2017) professionals with knowledge of their firm's Agile techniques

Source: Forrester's Q3 2013, Q2 2015, and Q3 2017 Global Agile Software Application Development Online Surveys

SCRUM

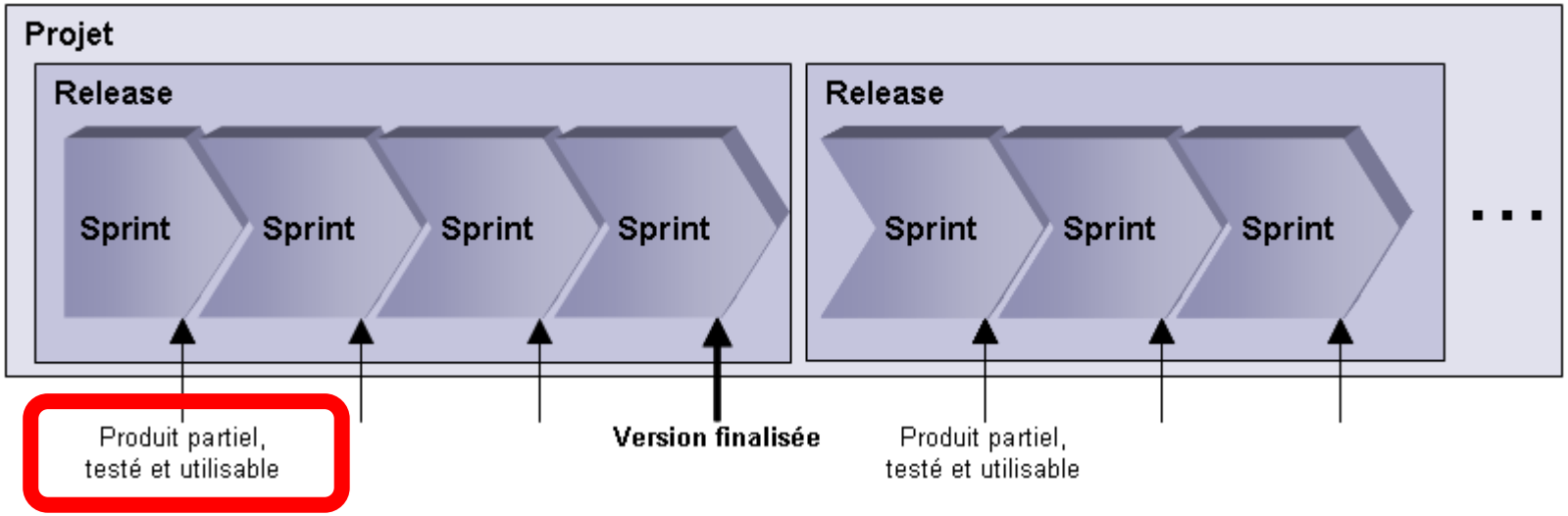
SCRUM



- 3 roles :
  - Product owner, team, Scrum master
- 3 documents:
  - Product backlog, Sprint backlog, Sprint result
- 3 meetings:
  - Sprint planning, daily meeting, Sprint review

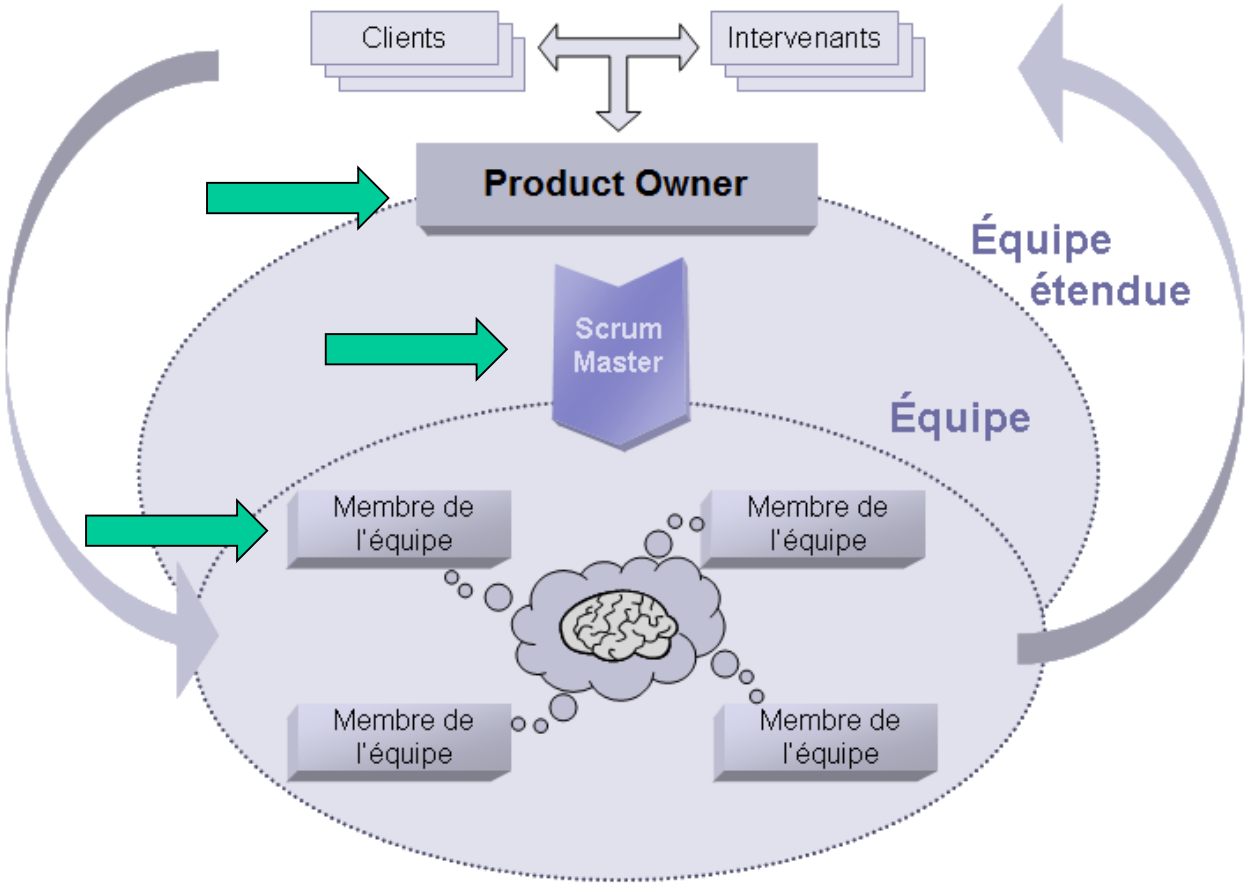
Plan de release

- Définit grossièrement le contenu des sprints d'une version
  - Bonne pratique mais plus obligatoire dans les nouvelles versions de SCRUM



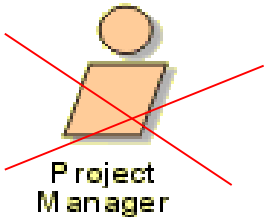
Source image: http://fr.wikipedia.org/wiki/Scrum

SCRUM: roles



Source image: http://fr.wikipedia.org/wiki/Scrum

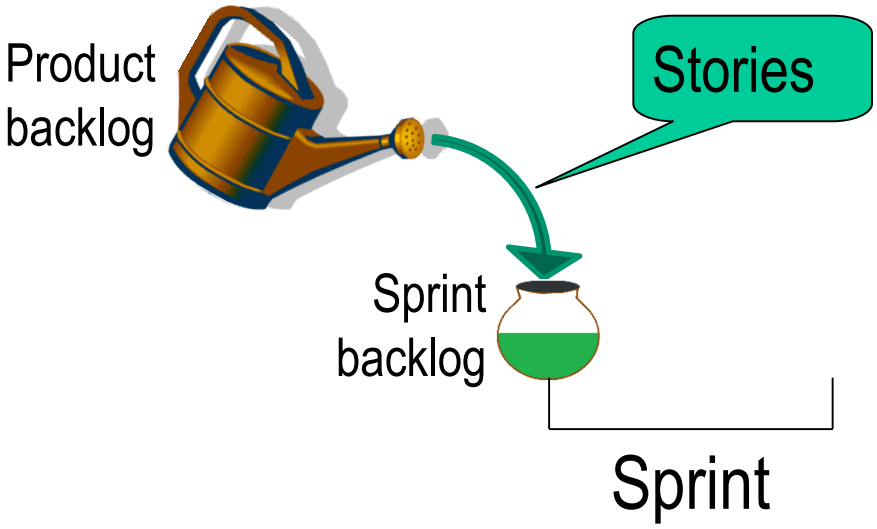
SCRUM: roles



- Pas de chef de projet
  - Les membres du team sont collectivement responsables du projet (auto-organisation)
- Product Owner (~ côté client)
  - Responsable de la vision du produit
  - Responsable de la priorisation des stories
  - Rôle augmente avec le déroulement du projet
- ScrumMaster (~ côté technique)
  - **Facilitateur, éliminateur d'obstacles**
    - Evite les interactions extérieures
    - Trouve une solution aux «obstacles» techniques (serveur, connexions,…)
    - Résout les problèmes d'équipe (absence, conflits,…)
    - Assure la transparence du fonctionnement / dysfonctionnement
  - Organise et anime les sprints
    - Mais n'organise pas le travail des membres du team dans un sprint
  - Mentor SCRUM (respect de la méthode)
  - Rôle diminue avec le déroulement du projet

Pas d'autorité hiérarchique sur le team

Backlogs



- Stories in the product backlog are ordered in decreasing order of the business value
- The stories to implement in a Sprint are recorded in the sprint backlog

Product backlog

- List of all the stories
  - Person in charge: product owner
  - Order of stories in list: decreasing order of business value
- Updated at the end of a Sprint
  - Stories that are **done** are removed from the list
  - Sprint review => new stories -> backlog
- Workload:
  - Sprint workload evaluated carefully at the beginning of the sprint
  - *If a release plan is used, whole workload grossly evaluated*

Business Value: MoSCoW

Simple way to express the relative priorities of the work items or stories according to the business.

- MoSCoW : Must be done (vital for the project)
- MoSCoW : Should be done (necessary for the projet)
- MoSCoW : Could be done (nice to have)
- MoSCoW : Won't be done (could be done in another release)

Requirements : User Stories

- Part of the backlogs (product & sprint)
- Each story must be implementable in one Sprint
- Each story must be testable at the end of the Sprint
  - Acceptance tests must be provided for each story
- Stories may be accompanied by additional documents

- Once a story is implemented, one gets an increment of product functionality

Story standard format (Mike Cohn)

As <type of user> **I want** < some goal> **so that** < some reason>

«**As** bank customer **I want** to change my PIN code **So that** I decrease the risk of my account be hacked»

- A story is only a reminder of a requirement to discuss with the product owner.
- This discussion will lead to the detailed requirements

Conditions of satisfaction

- Each story **must** come with its “condition of satisfaction” identified with the PO:
  - Boundary condition on input values
  - Constraints on the computations
  - Wrong input values it must detect
  - Situation it must identify and correct
  - …
- This will be the basis on which to write the **acceptance test** for the story

Scrum specifications : Stories + Acceptance tests

Backlog : Mike Cohn's format

ID	Theme	As a/an	I want to...	so that...	Notes	Priority	Status
2	Game	moderator	create a new game by entering a name and an optional description	I can start inviting estimators	If games cannot be saved and returned to, the description is unnecessary	Required	done
2	Game	moderator	invite estimators by giving them a url where they can access the game	we can start the game	The url should be formatted so that it's easy to give it by phone.		done
5	Game	estimator	join a game by entering my name on the page I received the url for	I can participate			done
6	Game	moderator	start a round by entering an item in a single multi-line text field	we can estimate it			done
8	Game	estimator	see the item we're estimating	I know what I'm giving an estimate for			done
~~40~~	~~Game~~	~~participant~~	~~always have the cards in the same order across multiple draws~~	~~it's easy to compare estimates~~	-	Replaced with A08 because I didn't want the story to talk about "the same order" as that might be a UI implementation detail	~~todo~~
35	Non-functional	user	have the application respond quickly to my actions	I don't get bored			done
36	Non-functional	user	have nice error pages when something goes wrong	I can trust the system and it's developers			done
A11	Non-functional	Researcher	results to be stored in a non-identifiable way	I can study the data to see things like whether estimates converged around the first opinion given by "estimator A" for example	No names or story text should be stored but we should store each card of each hand, know who played it, and know the final accepted estimate		
A05	Game	moderator	edit an item in the list of items to be estimated	so that I can make it better reflect the team's understanding of the item			
22	Archive	moderator	export a transcript of a game as a CSV file	I can further process the stories and estimates	Exported file should be directly importable back into the system.		done

http://blog.mountaingoatsoftware.com/a-sample-format-for-a-spreadsheet-based-product-backlog

Bad (but real) example

Story Name	As a... (Actor)	I can... (Objective)	So that…(Benefit)	When I... (Action)	Then... (Expected Result)
Print from table grid	Operator	Print from table grid	Have data printable	Am in tracking grid	Printed table grid information

Definition of Done

- Criteria to decide that a story is completed and can be removed from the backlog
  - Must be decided by the team at the inception of the project
  - Will be used throughout the project
- Example (Mike Cohn) : story **done** when
  - Code is well-written (no refactoring needed)
  - Acceptance tests are successful
  - Automated tests ready at all appropriate levels.
  - Code is inspected.
  - Code is documented as needed.

Sprint planning

Goal: to fill the sprint with just enough work to keep the team busy

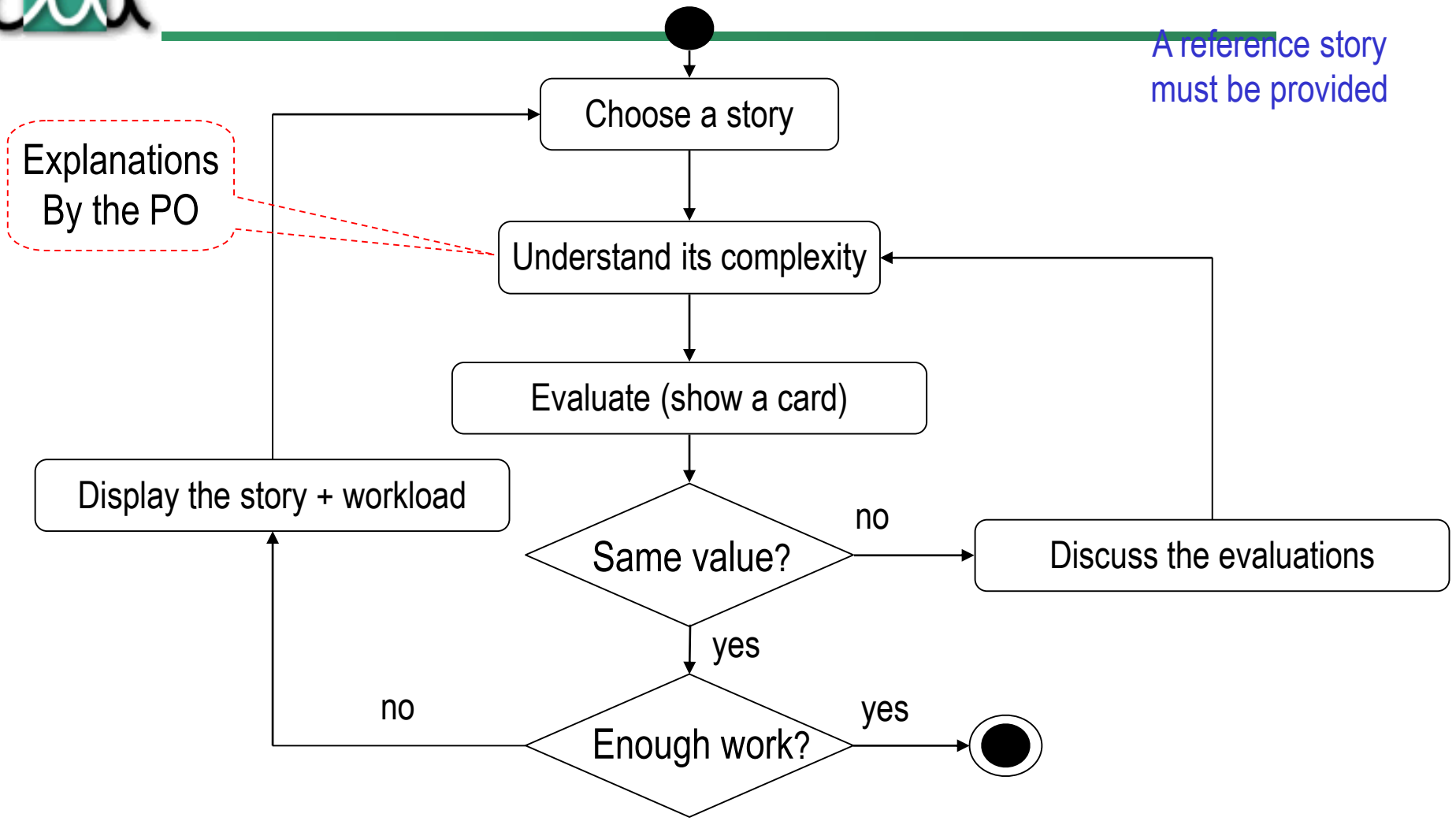
Roles:

- PO: gives extended explanations on the stories selected fro the sprint
- Team: evaluation of the workload of the stories + split the stories in small tasks
  - Workload of a task: max 1 day
  - Each member chooses the tasks he will work on

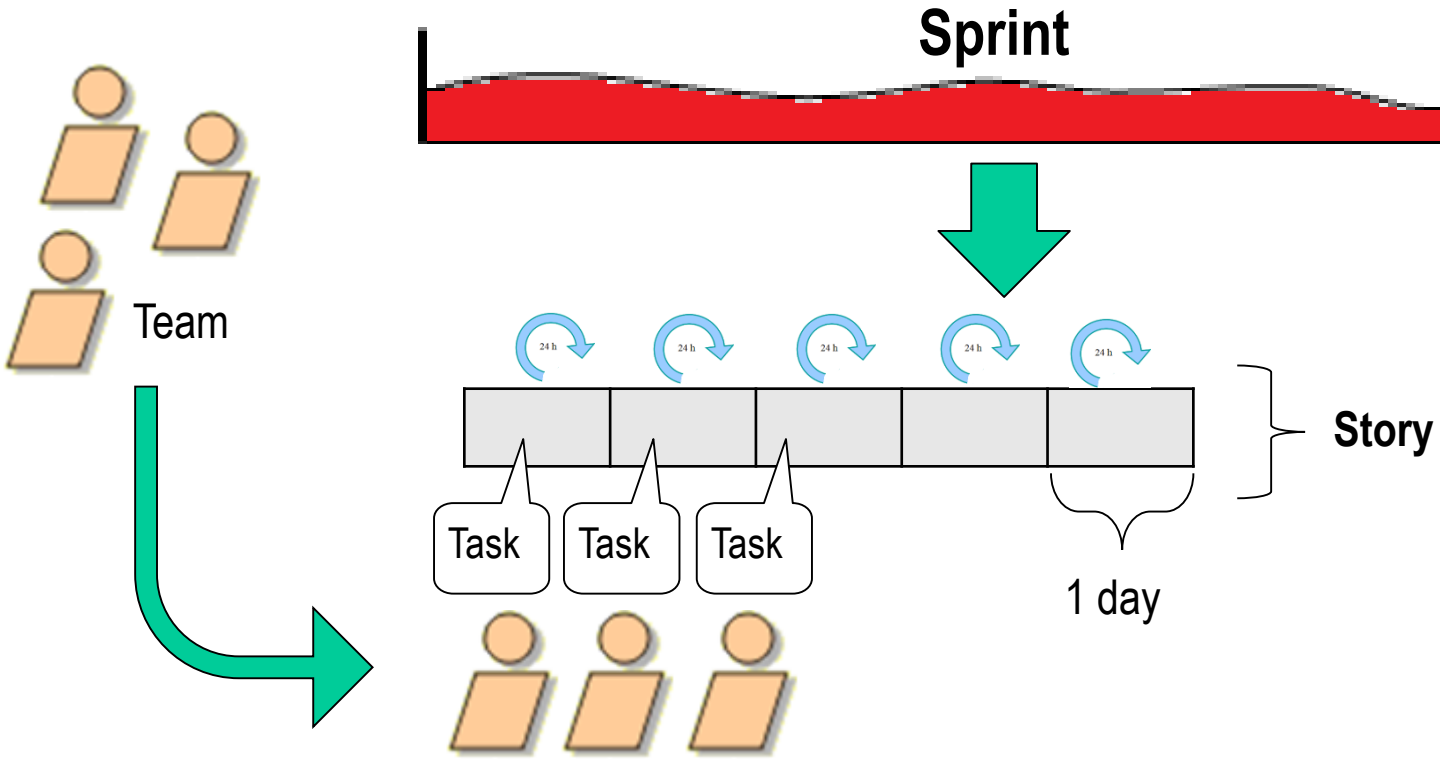
Story workload evaluation
Planning poker

- Done jointly by the team
- Each member is provided a set of card with non linear numbers
  - Typically following Fibonacci (0.5,1,2,3,5,8,13,20,40,…)
- A reference story must be chosen (known by all the team members)
- Steps
  1. Get the story on the top of the product backlog
  2. The story is explained by the PO
  3. The team discusses the problems associated with the story
  4. Each team member evaluates the story and assigns it a card (workload)
  5. The evaluation are compared. If not equal, back to 2
  6. If equal, place the story in Sprint backlog

Planning poker: process



Post workload evaluation: tasks



Tasks

Examples

- Analysis & Design of the story
- Implementation
- Testing
- Meeting with PO to clarify some points
- Documenting the implementation (maintenance)

If the work related to a task not clear enough, split the task in 2 subtasks

  - Investigation task to get more info on the work to be done
  - Development task

How many stories in a Sprint ?
Team availability

- Gather each members' availability during the Sprint
  - Subtract all tasks that are not related to the project
- Aggregate each member's availability to get the team's availability
  - The total workload on a Sprint must not exceed 80% of the team's availability
- The SCRUM Master makes sure that :
  - The team is not overcommitted
  - The individual are not overcommitted
  - All the available resources are utilized

The iteration goal must be stable

- Never add a new story to an on going iteration
  - The new story gets added to the backlog to be scheduled at the next iteration planning
- If a task is more complex than expected
  - Split it in subtasks
  - Evaluate the workload
  - Add the subtasks to the list of tasks to be done in the Sprint

Working on tasks: principles

Each task is developed individually or in collaboration

- The code is **collectively owned**. No star developer mentality!
- When one member's list of task is exhausted, he will picks up some of the other member's tasks
- The taskboard is updated as the tasks change status (daily meeting)
- All the iteration's tasks should be in the "done" state at the end of the Sprint

Daily coordination meeting

- Standup meeting 15 min max
- Question to ask to each team member:
  - What did you do yesterday?
  - What problem did you have?
  - What do you think you'll do today?

Release planning

If the project teams wants to plan a release, then a rough evaluation of the workload over the release must be performed

Sprint management tools

- Taskboard
- Burndown chart

Self-organization: taskboard

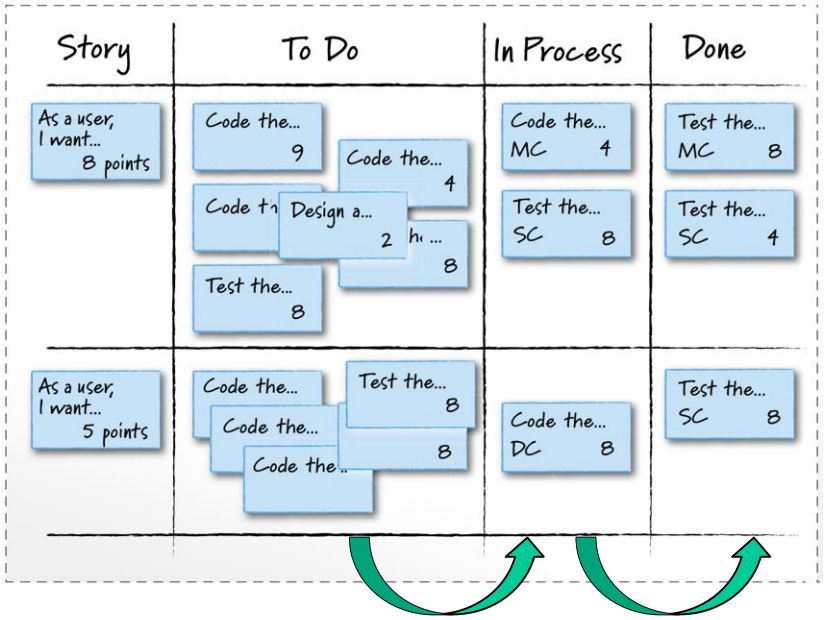


Image: Cohn M. – Succeeding with Agile. Addison-Wesley, 2010

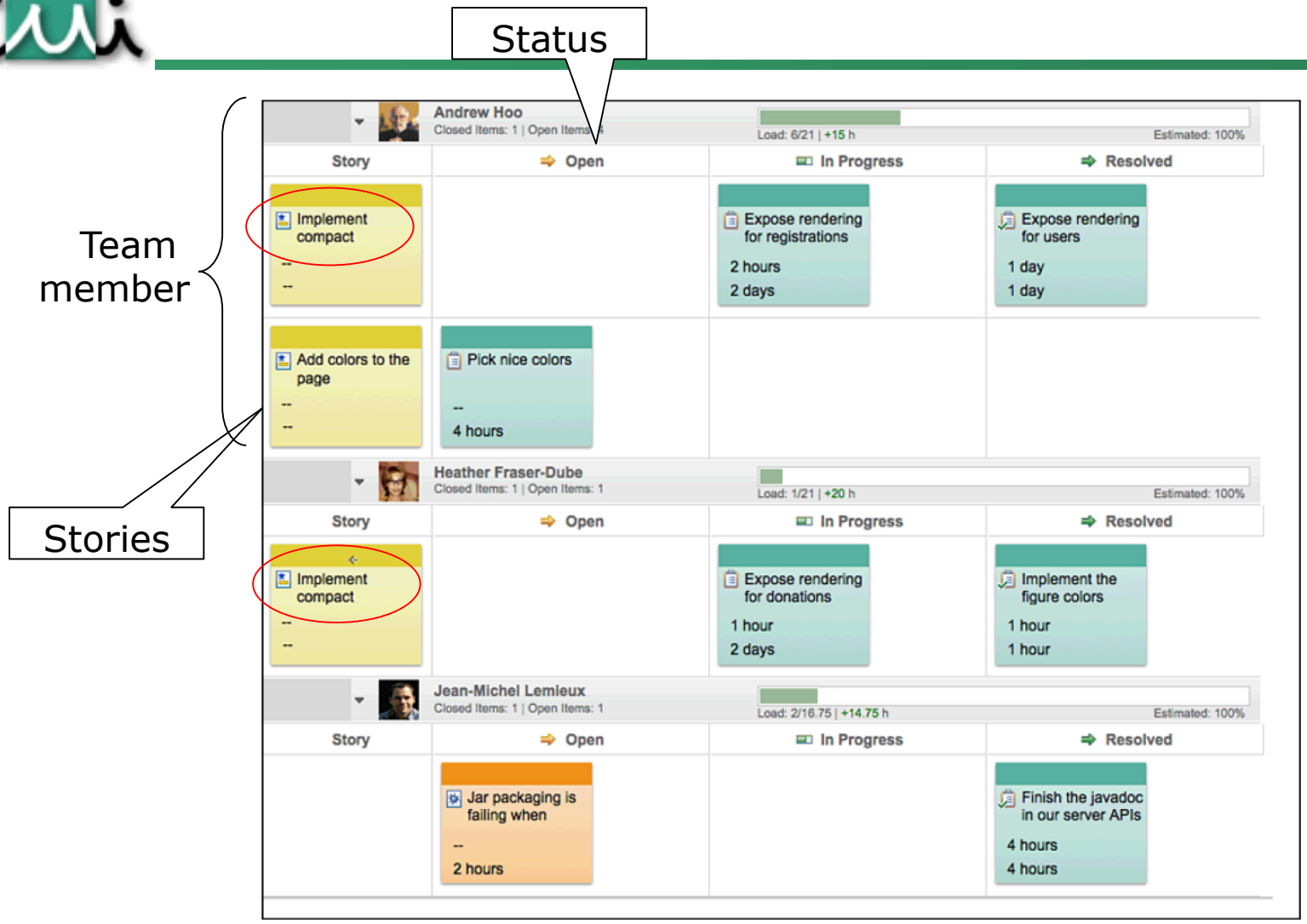
Team members

- Choose tasks on the taskboard to fill up their availability for the duration of the Sprint
- Show it on the task by:
  - Writing one's name on it
  - Sticking the task in one's area on the taskboard

Physical taskboard example

- Column name may vary: to do, started / in progress, blocked, waiting for validation / to verify, under test, completed / done / resolved
- Lines and colors could be used to identify the stories and the task assignment to people

Electronic version



When is a story completed?

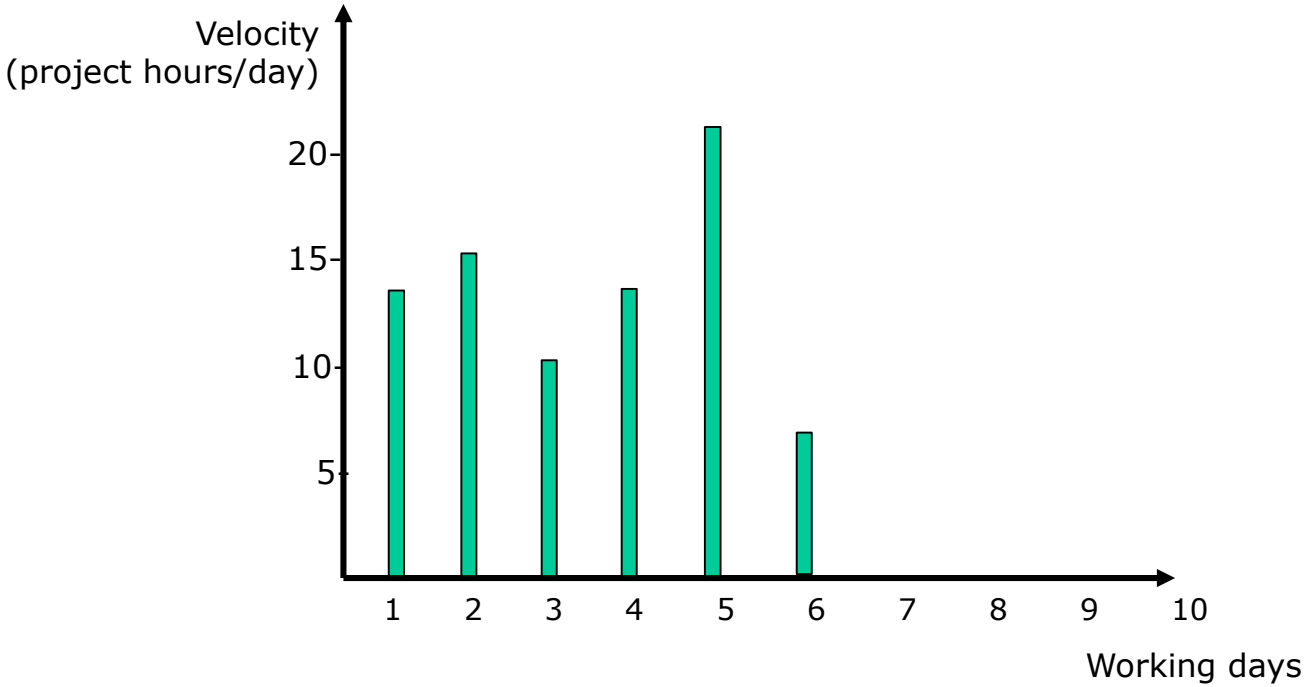
- When it is **DONE** (see definition of Done)
  - In particular: when its **acceptance tests** are successful
  - linked to the conditions of satisfaction

Velocity / burndown chart

Visualizing progress / estimating remaining work

- Velocity: nb of points or project hours completed per unit of time
- Burndown chart: displays the work progress toward the implementation of all the stories of the iteration (/ release)
- *Release duration:*
  - *(release workload / average iteration velocity) * iteration length.*

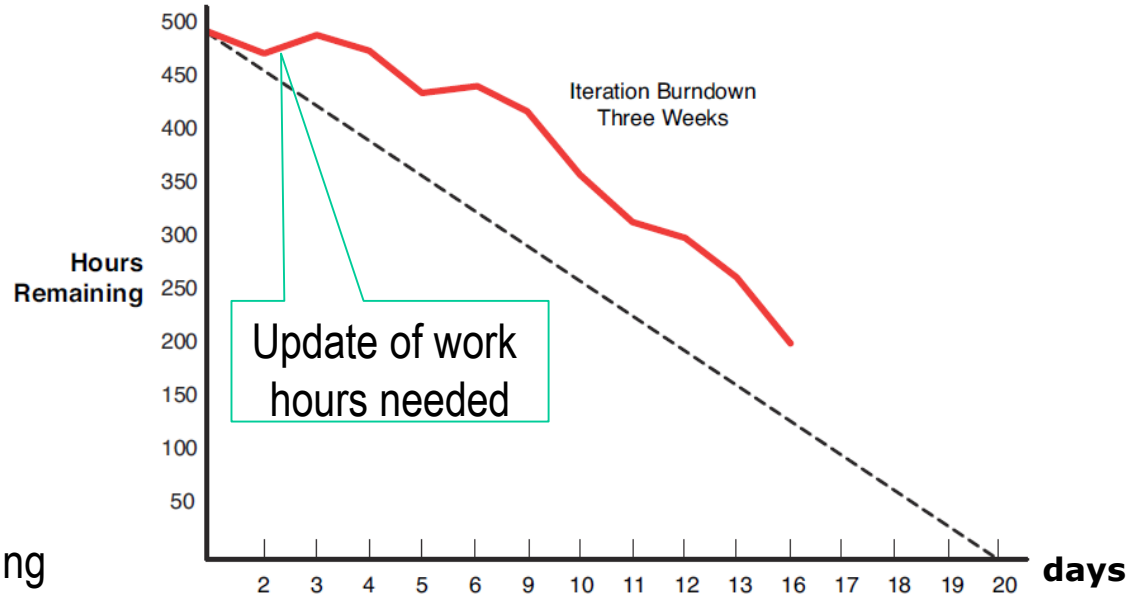
Velocity per day



Updated: daily meeting

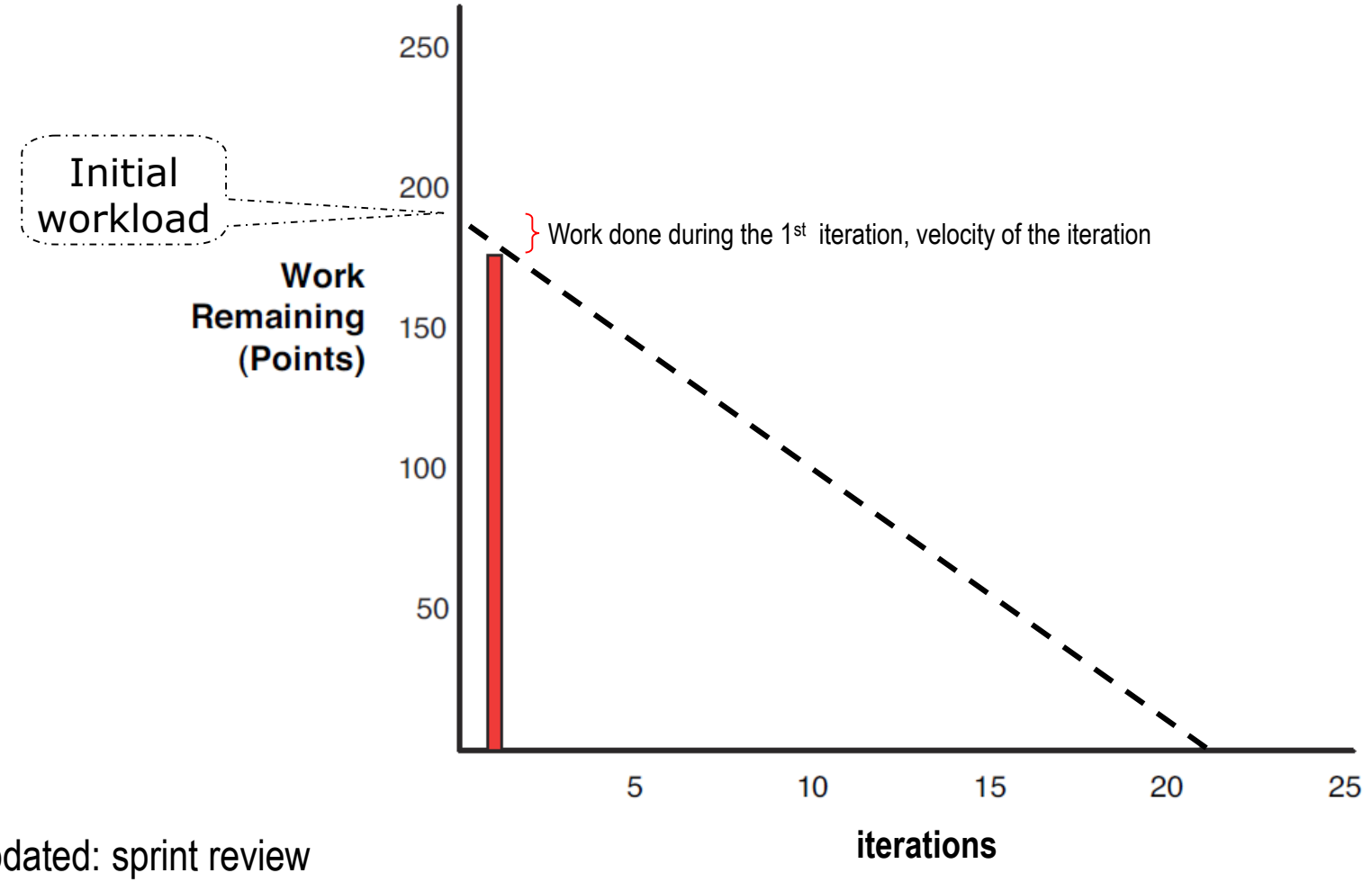
Iteration burndown chart

- Gather the work hours of all the tasks over the iteration
- Plot the straight line representing the theoretical velocity
- Each day, plot the actual remaining work hours (update the global workload if needed)



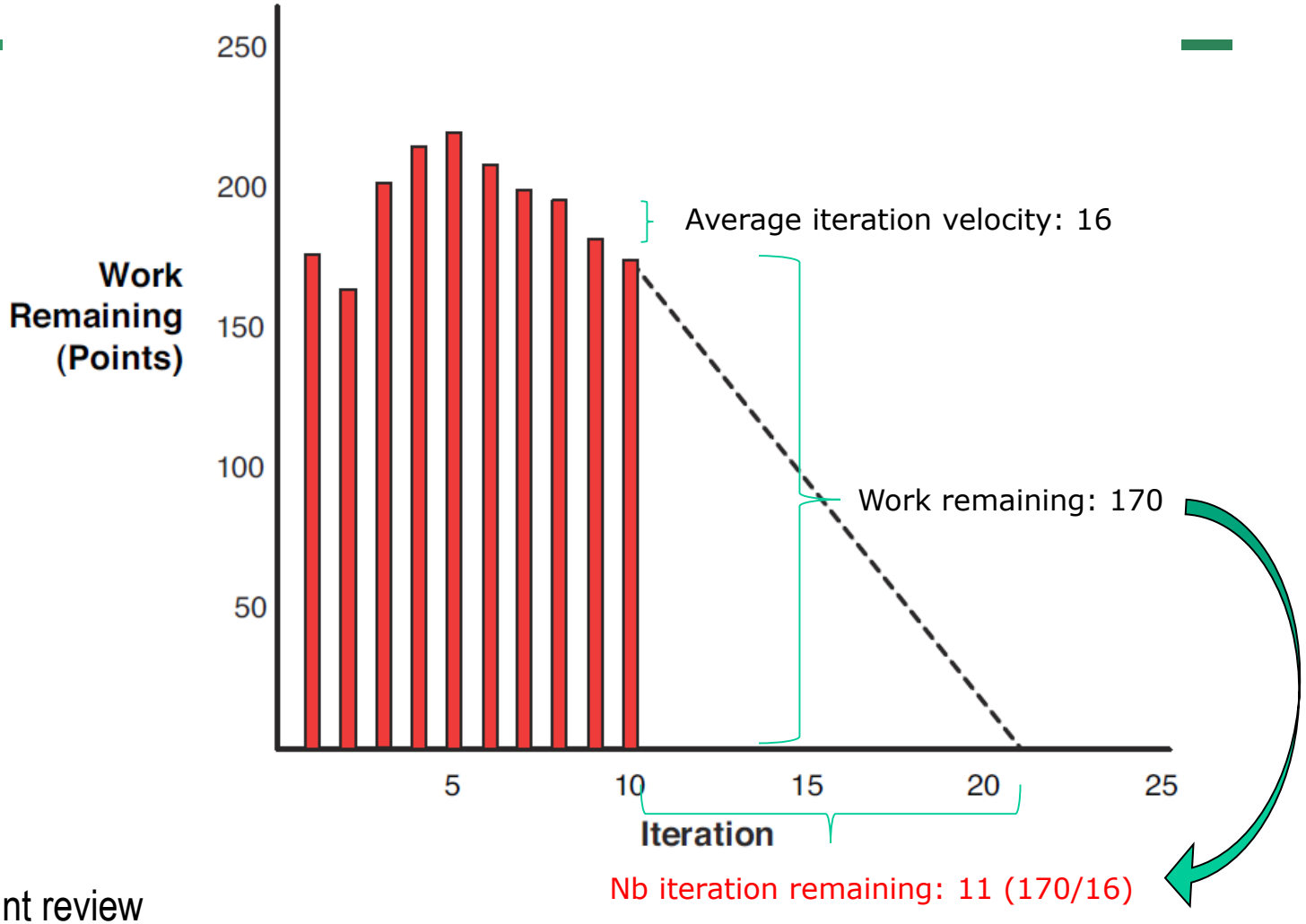
Updated: daily meeting

Release burndown chart
(if relevant)



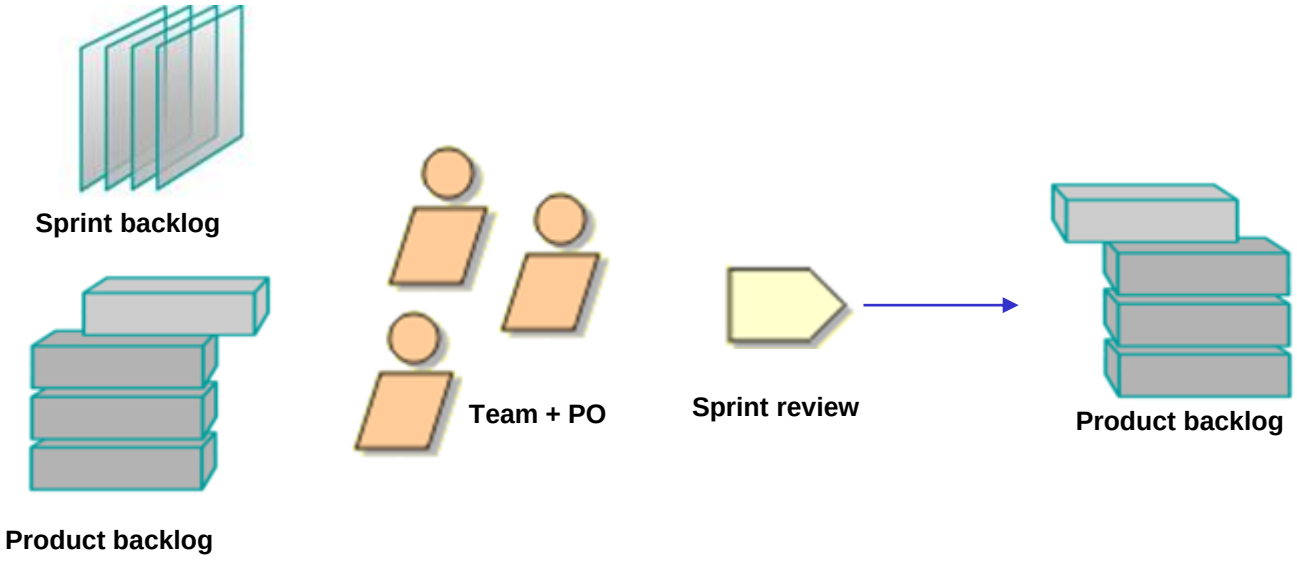
Updated: sprint review

Updating release burndown chart



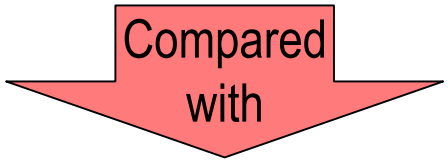
Updated: sprint review

Sprint review



Sprint review

Sprint result



Sprint planning

- Changes to the product backlog?
  - New stories as a result of the Sprint
  - Re-ordering of the backlog

Update: product backlog, (release burndown chart, release plan)

Technical stories

- Stories discovered during the Sprint review
  - Change of implementation
  - Improvement of a QA (NFR)
  - Bug correction
- Stories supporting other stories
  - Implementation of DB tables
  - Implementation of a logging mechanism
  - …

TS are inserted in the backlog

Spike

- When some specification requires investigations, one may create a Spike (borrowed from XP)
  - Story to investigate alternative solutions / to better understand some requirements
    - Technical spike: to investigate some technical problem
    - Functional spike: to investigate user-related solution
  - The output may not be executable code but : a piece of documentation, a revised story, new stories, implementation decision,…
  - Spikes should be exceptional, not the rule
    - In normal situation, the investigation of the design is part of the story development's work

Spikes are inserted in the backlog

Retrospective

"At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly"

- What went well?
- What could we improve on?
- What should we do differently next time?
- What still puzzles us?

- Junior team: at the end of each iteration.
- Senior team : whenever it makes sense to do so

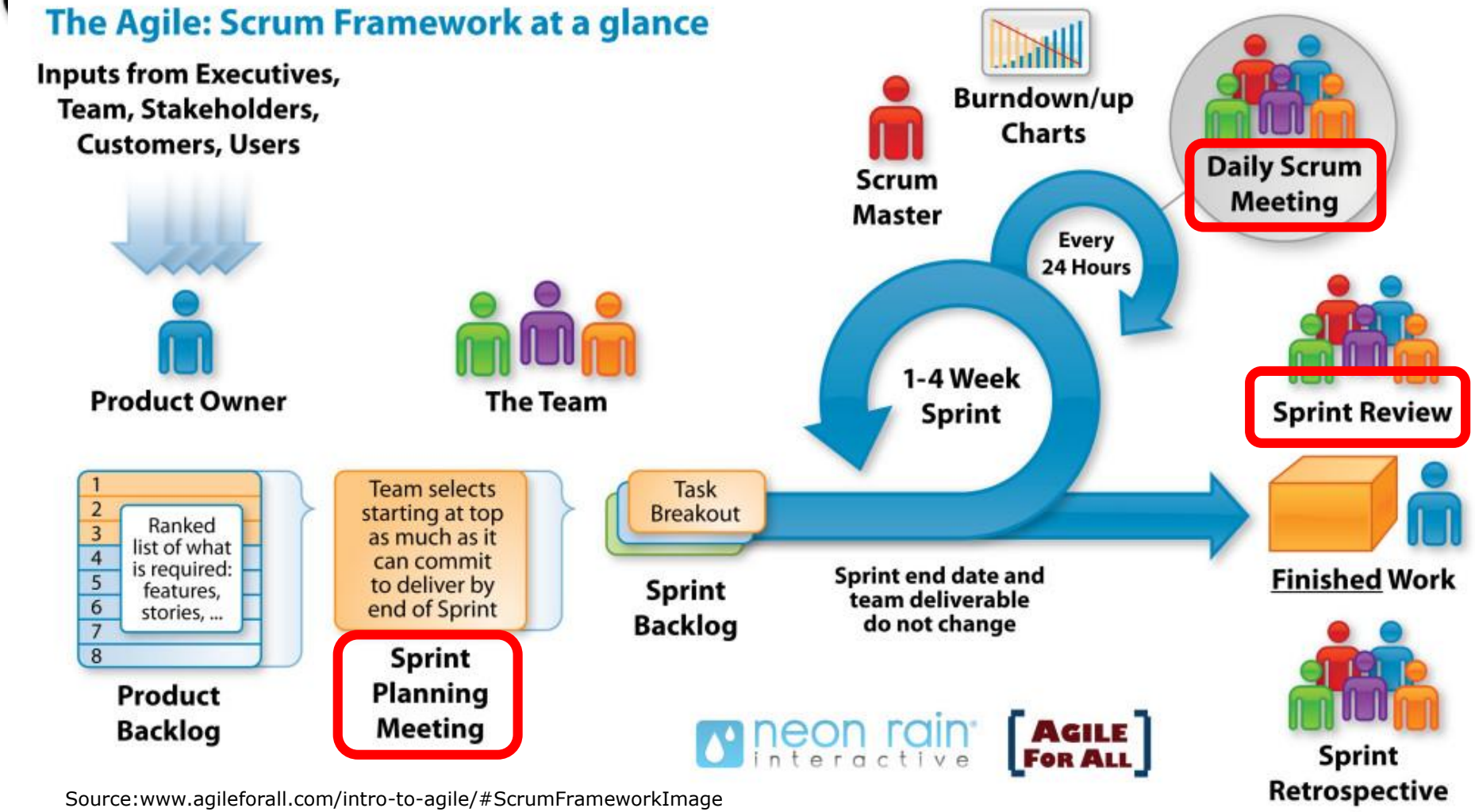
Retrospective

Improve the **Product**

Improve the **Team**

Source:www.agileforall.com

Summary: scrum life cycle



Source:www.agileforall.com/intro-to-agile/#ScrumFrameworkImage

Sprint 0

- Develop a **vision** document
- Initial backlog (initial release plan)
- Definition of **done** for stories
- Investigation of the problem (spikes)

Stories best practices

Stories: the INVEST Model

The stories should be:

I	Independent	The user story should be self-contained, in a way that there is no inherent dependency on another user story.
N	Negotiable	User stories, up until they are part of an iteration, can always be changed and rewritten.
V	Valuable	A user story must deliver value to the end user.
E	Estimable	You must always be able to estimate the size of a user story.
S	Small	User stories should not be so big as to become impossible to plan/task/prioritize with a certain level of certainty.
T	Testable	The user story or its related description must provide the necessary information to make test development possible.

Source: https://en.wikipedia.org

What is small

SCRUM Coach Richard Lawrence recommends to implement 6-10 stories per iteration

[https://agileforall.com/patterns-for-splitting-user-stories/]

So the workload of each story (number of points) together with the velocity of the team should comply with Lawrence's recommendation

Example: iteration duration: 10 working days, 4 full time people in the team, velocity of the team : 23 hours / day. This leads to 23*10 = 230 hours of work per iteration.

If wish to implement 6 stories per iteration, on average each story must be implemented in 230/6 = 38 hours. If 1 story point is equivalent to 4 hours, the average number of point of each story must be 9.5. Then we may split the stories whose number of points is bigger than 8.

Spliting stories

When…

1. They are not **E**stimable because they are not **S**mall
2. They are too big to be implemented in a single sprint
3. It is hardly understood as a whole
4. It has many variants
5. It has several error conditions
6. There is room in the next iteration but not enough to implement a full story

How to split ?

Separate out..

1. The variants in data processed, data outputted, algorithms,…
2. Processing parameters
3. Business rule variations / alternatives use cases

Include one or more variants in each story depending on the resulting size of the story

Example

As a researcher **I want** to record my publications in the open archive **so that** they become visible by my colleagues and peers

- Variants
  - Single publication, several publications
  - Journal article, conference paper, book chapter, book
  - Business rules to check the validity of the input
  - Input format (mark21, plein text, GUI)
  - Visibility rules (non visible, visible from the institute, visible to all people)
  - Enter, update, delete the record
  - Error conditions / error correction techniques

A possible split…

As a researcher **I want** to record my publications in the open archive **so that** they become visible by my colleagues and peers

1. **As a** researcher **I want** to record my journal articles in the open archive with all the quality checks **so that** the article is standardized and become visible by my colleagues and peers
2. **As a** researcher **I want** to record my conference papers in the open archive with all the quality checks **so that** the paper is standardized and become visible by my colleagues and peers
3. **As a** researcher **I want** to select the visibility of my records in the open archive **so that** I could control who could read what article or conference proceedings
4. …

Workflow stories

- When implementing a workflow, it is not a good idea to work on each of the steps individually through stories
  - Each of the story will not be **V**aluable until the full workflow is implemented

- Solution : implement the workflow end to end in one story but
  1. Find the minimal workflow, with all the default values for the steps (most frequent situation)

  Or

  2. Bypass some of the steps and only keep the smallest required steps

So that we can still implement a valuable, full workflow. All the variants will be implemented in later stories.

Last word…

Make sure that the resulting stories are:

- As **I**ndependent as possible
  - However when implementing variants of some workflow, the main workflow must generally be implemented first
- **V**aluable
  - However the variant of the workflow only gets its value when the basic workflow is implemented

Advices from a SCRUM Coach

Choose the split that lets you deprioritize or throw away a story

- The 80/20 principle says that most of the value of a user story comes from a small share of the functionality. When one split reveals low-value functionality and another doesn't, it suggests that the latter split hides waste inside each of the small stories. Go with the split that lets you throw away the low-value stuff.

Choose the split that gets you more equally sized small stories

- The split that turns an 8 point story into four 2 point stories is more useful than the one that produces a 5 and a 3. It gives the Product Owner more freedom to prioritize parts of the functionality separately.

[Richard Lawrence on https://agileforall.com/patterns-for-splitting-user-stories/]

Resources

https://agileforall.com/patterns-for-splitting-user-stories/

https://agileforall.com/wp-content/uploads/2009/10/Story-Splitting-Cheat-Sheet.pdf

https://agileforall.com/2012/01/new-story-splitting-resource/

Epic

- A large story that comprises (may be split into) smaller stories but whose business value is realized only when all stories are implemented.

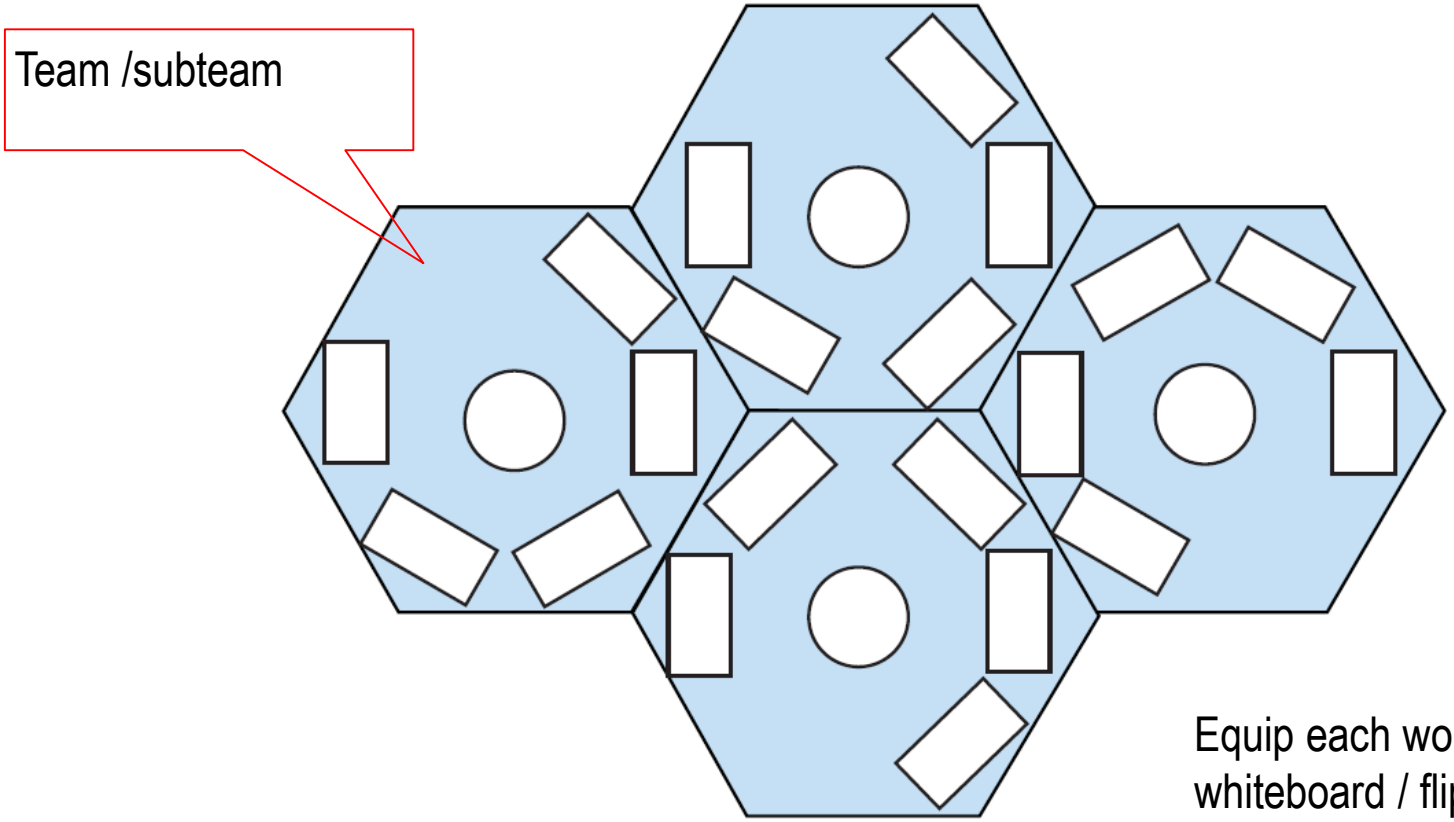
- This resemble use cases with their many flows

SCRUM best practices

Agile team

- Collocated
  - Small group in the same room ($\leq$ 8 people)
- Each member is respected
  - Can share problems and issues openly
  - Does not fear to be blamed for mistakes
- Close collaboration
  - Designing
  - Coding
  - Testing

Example: work area



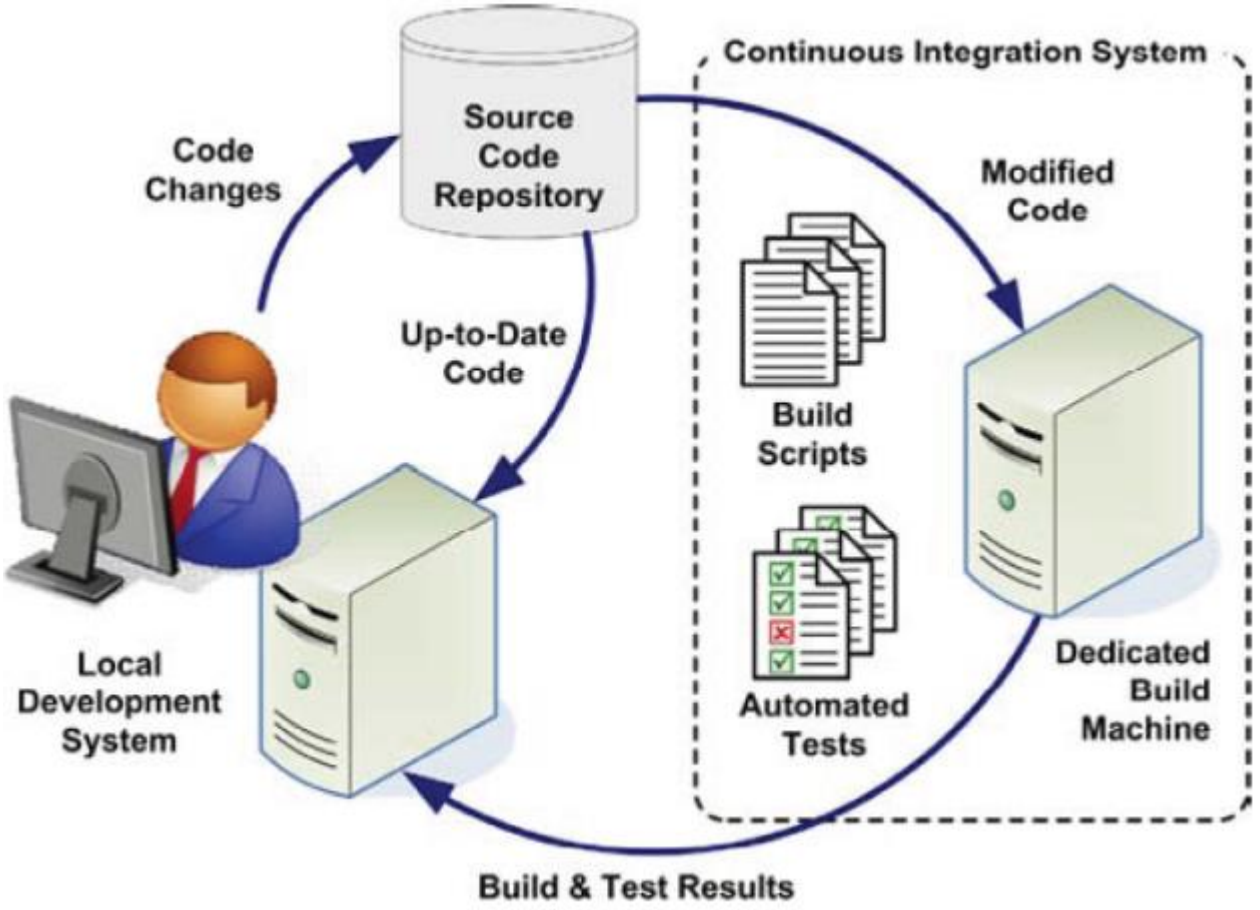
Equip each work areas with whiteboard / flip chart and wall areas for sticky notes

Source: Ambler S.W., Lines M. - Disciplined Agile Delivery. IBM Press 2012

Test-first programming et Integration continue

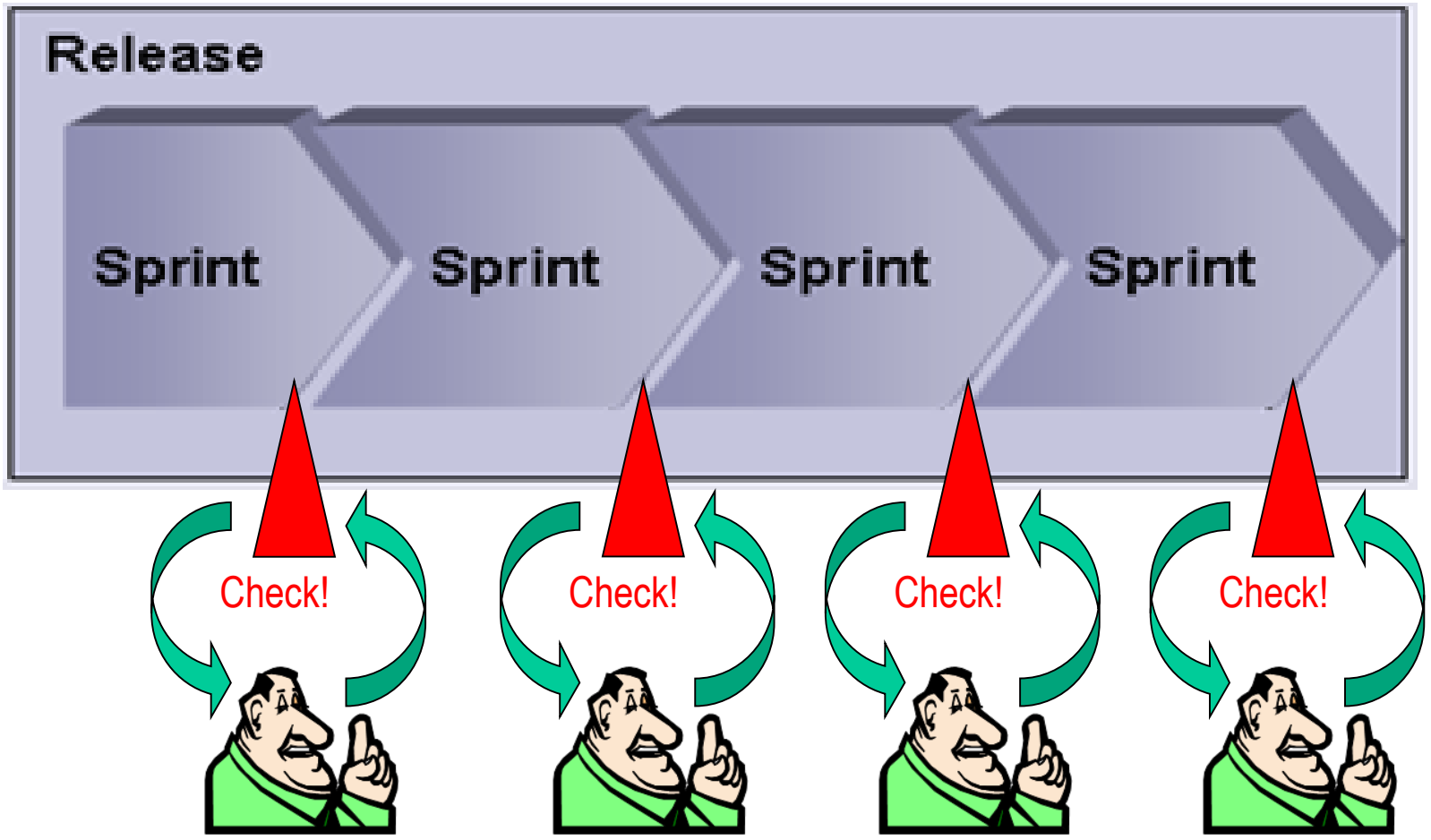
- Test first programming est une discipline amenée par XP
  - Elle consiste a écrire les tests avant d'écrire le code des composants
  - Elle oblige a définir les interfaces (protocole) en premier
  - Elle assure qu'on dispose de tests pour tout les composants
- Constat: elle ne s'impose pas dans l'industrie
- L'intégration continue consiste à intégrer tous les soirs les exécutables produits par l'équipe pour évaluer la qualité du tout
  - Les tests définis pour chaque composant sont lancés
  - Un compte rendu de l'exécution des tests est produit chaque matin

Continuous integration



http://www.fil.univ-lille1.fr/portail/stagesLicence/2010/resumes/guette/resume.php

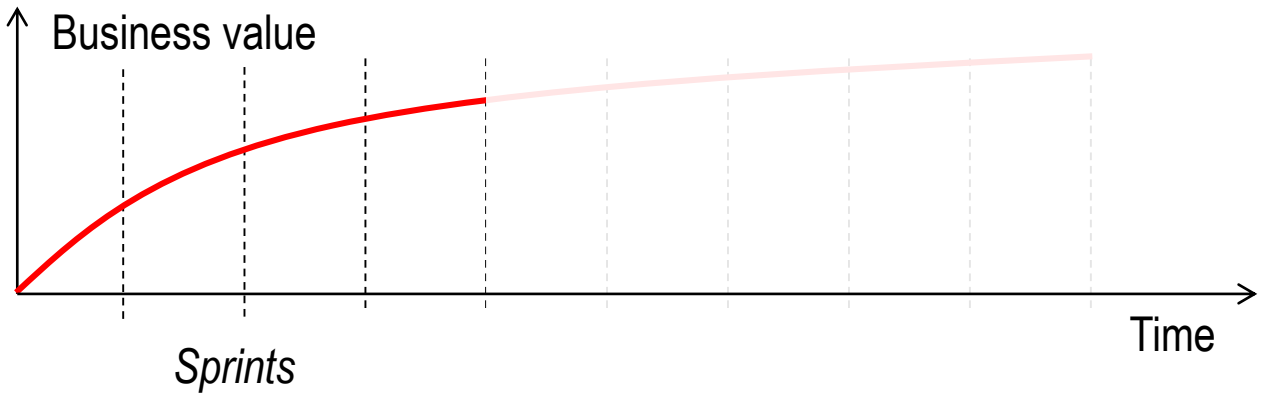
How to monitor the process ?



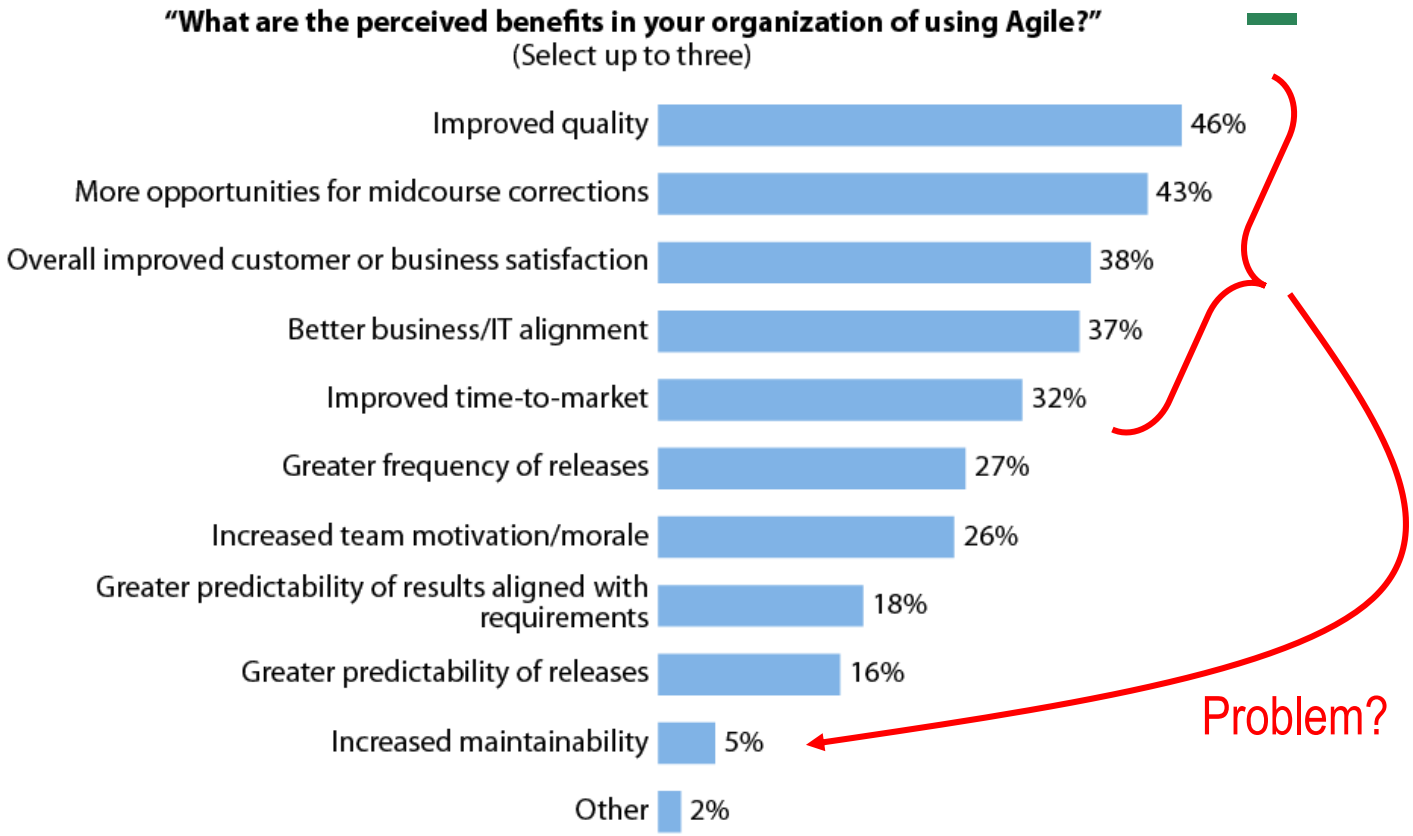
Benefit of the BV ordering

One can stop after any iteration and have :

- A stable and useable version
- Implemented the most useful function
- The customer will always get the best value for money



What about agility?



Base: 205 IT professionals who are implementing or have implemented Agile

Source: November 2011 Global Agile Software Application Development Online Survey

Tom Grant - Navigate The Future Of Agile And Lean. Forrester Research, Jan 10, 2012