Agile Methods

Matthieu Moy

UCBL

2018

Outline

Problems and Solutions in Project Mgmt (level 0)

Level 0: no project management ("code and fix"). Maybe OK when working alone, but ...

- Planning impossible
- One developer leaving the team ⇒ failure of the whole project?
- Not reproducible: may work today, but tomorrow?
- ...

Problems and Solutions in Project Mgmt (traditional)

Traditional project: rationalize the process, document as much as possible (V cycle, “say what you do, do what you say”)

- Planning easy: all specifications are there, metrics from previous projects too, ...
- One developer leaving the team $\Rightarrow$ the next ones will read the docs
- Reproducible: just follow the procedures

But ...

- Metrics: 9 pregnant women’s problem, metrics management vs human being management
- Is following the plan the most clever option if the client changes their mind?
- Is the documentation up to date?
- What if procedures are sub-optimal? How to change them?

Problems and Solutions in Project Mgmt (agile)

Agile: keep in mind that working software is the main goal

- If specifications are broken, change them.
- Doing specification is hard. Discussing a prototype is easier $\Rightarrow$ Release early. Release often (Eric S. Raymond, The Cathedral and the Bazaar)
- Many traditional project management techniques turn out to have bigger overhead than benefit. Abandon them. (“eliminate waste”, lean principle)
- Developers are important. Give them power.

Manifesto for Agile Software Development

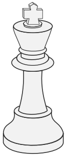
We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

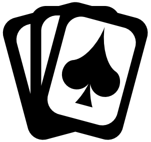
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

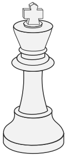
`http://agilemanifesto.org/`, 2001.

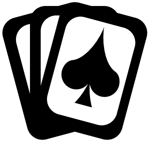
Time in Agile Methods

Traditional = 

Agile = 

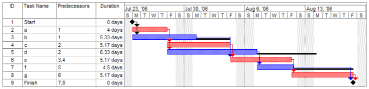
Time in Agile Methods

Traditional = 

Agile = 

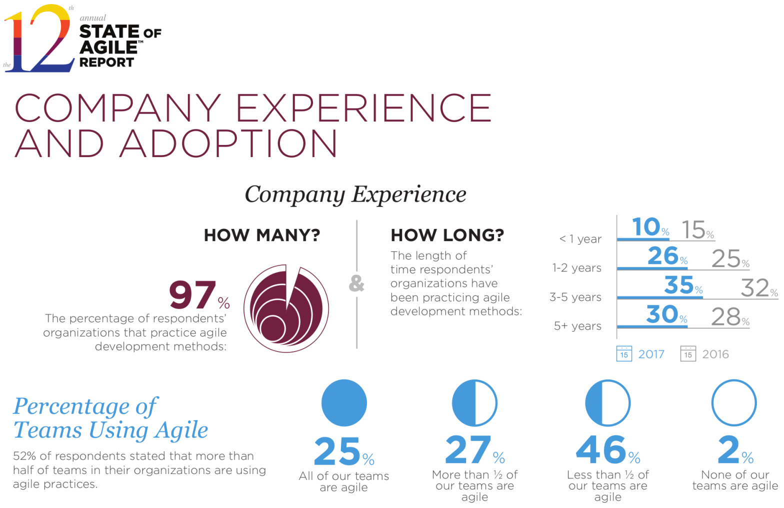
$$\text{Start} \xrightarrow{\text{Projet}} \text{End}$$

$$\text{Start} \rightarrow \cdots \xrightarrow{\text{Iterations}} \cdots \rightarrow \cdots$$



ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days

Who care?



the 12th annual STATE OF AGILE™ REPORT

COMPANY EXPERIENCE AND ADOPTION

Company Experience

HOW MANY?

97%

The percentage of respondents' organizations that practice agile development methods:

&

HOW LONG?

The length of time respondents' organizations have been practicing agile development methods:

	2017	2016
< 1 year	10%	15%
1-2 years	26%	25%
3-5 years	35%	32%
5+ years	30%	28%

Percentage of Teams Using Agile

52% of respondents stated that more than half of teams in their organizations are using agile practices.

25%	All of our teams are agile
27%	More than ½ of our teams are agile
46%	Less than ½ of our teams are agile
2%	None of our teams are agile

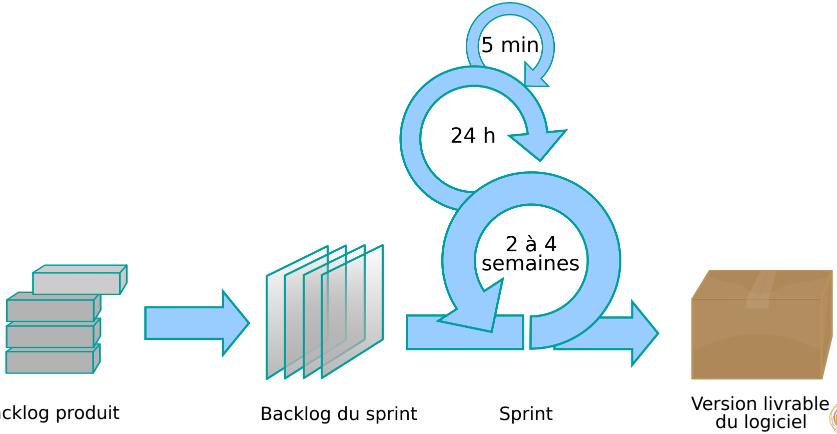
https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report

Scrum, Lean, Kanban, DevOps, XP, ...?

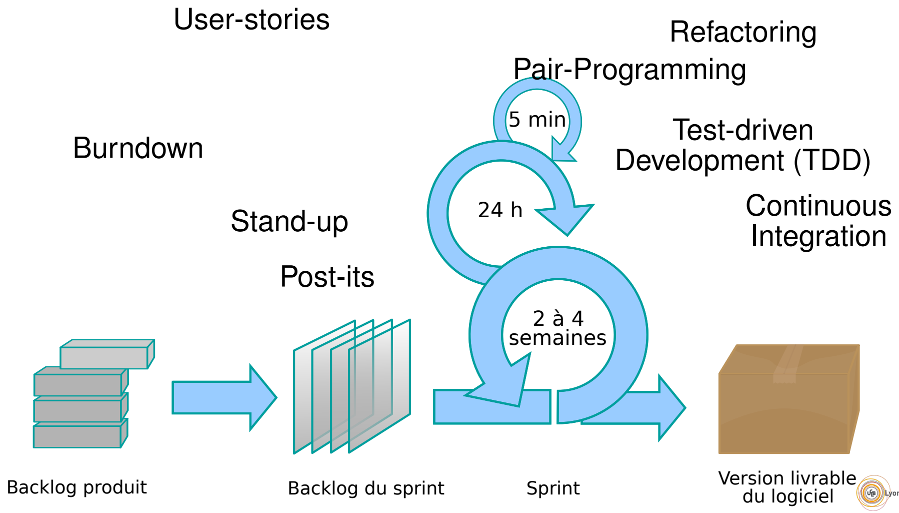
- Global trend, many complementary tools or variants:

eXtrem Programing:	focused on coding practices (code review, tests, ...)
Scrum:	divide a project into iterations, plan each iteration at once
Kanban:	$\approx$ lightweight variant of Scrum
Lean:	Global approach (company-wide) to eliminate waste
DevOps:	Automate what can be (from development to deployment)

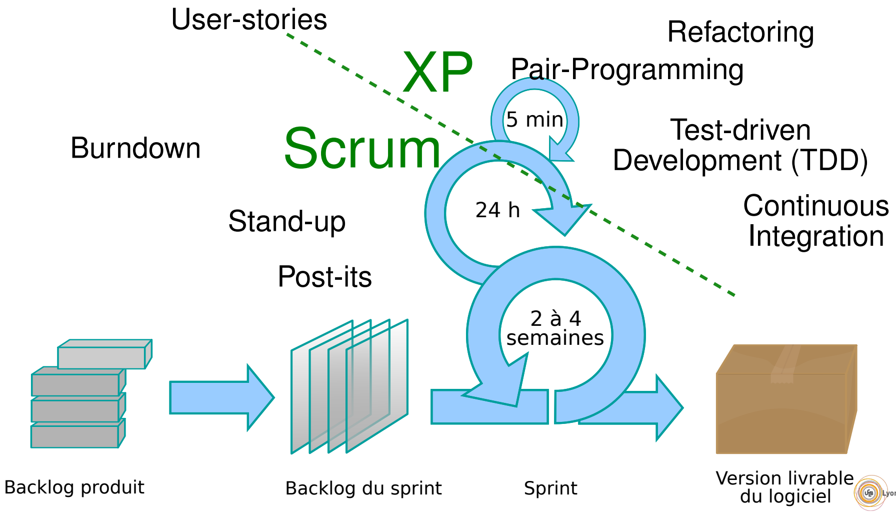
Iterations, Scrum, eXtrem Programming (XP)



Iterations, Scrum, eXtrem Programming (XP)



Iterations, Scrum, eXtrem Programming (XP)



Outline

Outline of this section

Scrum

- Set of practices to organize a team ($\approx$ 7 +/- 2 developers)
- Focused on human interactions (inside and outside the team)[1]
- Short iterations (called “sprint”) & delivery cycle: 1 sprint = 1 week to 1 month
- The most popular in companies today(?)
- Scrum Roles:

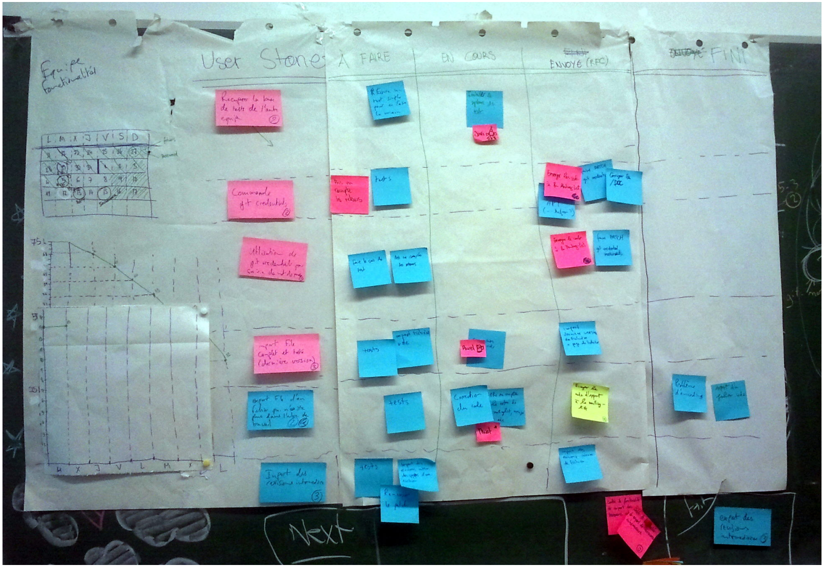
  Product Owner (PO): discussion with the client
  Scrum Master: facilitator ($\neq$ boss), helps the team follow Scrum (or not)
  Developers: Write code, take decisions

[1] The fact that the vocabulary of team sports is used is not a coincidence

Journey of a feature in Scrum

1. Client wants “something”
2. Discussion with the PO (Product Owner)
3. Agreement on a (set of) user stories (“as a ... I want to ... in order to ...”
4. User stories not started = product backlog (PO fills-in the backlog, developers empty it)
5. Start of iteration: decide on the sprint backlog
6. User stories split into technical tasks
7. Each task goes from TODO $\rightarrow$ ongoing $\rightarrow$ Done.
8. Demo at the end of iteration
9. Release (or not)
10. Retrospective and celebrate!

Scrum Board


Equipe fonctionnalité
User Stories
À FAIRE
EN COURS
~~Fini~~ ENVOYÉ (RFC)
~~Envoyé~~ FINI
L M X J V S D
Récupérer la base de tests de l'autre équipe ②
Commande git credentials ②
Utilisation de git credential pour saisie du login
Import File complet et testé (dernière version) ③
Export File d'un fichier qui n'existe plus dans l'arbre de travail ② ③
Import des révisions intermédiaires ③
tests
Next

Outline of this section

Scrum Board

- Can be physical (post-its) or virtual (GitLab or GitHub's issues, Trello, ...)
- Virtual: more traceability, multiple geographical sites, ...
- Physical also has benefits:
  - 1 post-it = 1 unit of information. Text doesn't fit on post-it $\Rightarrow$ split into smaller items.
  - (Use a felt tip pen, not a thin pen)
  - Always visible on the wall $\Rightarrow$ you can't claim you forgot!
  - No remote access $\Rightarrow$ if your boss wants to see the board, she must come in the room.
  - Flexible (take a pen and draw a line Vs ask the admin of the project to create a column)
  - Satisfaction of moving post-its to DONE :-)

Role: Product Owner

- Role: discuss product specifications. Must understand the client needs (not necessarily a computer-scientist)
- Preserve developers from direct customer interactions (distraction), but provides as much information as possible to the client.
- Does not take technical decisions.
  - Example: “Replace technology X with technology Y” is not decided by the PO
  - “Improve the scalability of the system” can be turned into a user-story and asked by the PO
  - The team may decide that technology Y is needed to accomplish it.
- Decides on the priority of story (= which one to do first)
- Discusses/negotiates the sprint backlog with developers
- Does not change the sprint backlog during the sprint

Role: Team Member (= developer)

- Role: develop the product.
- Includes development, debugging, testing
- Scrum's ideal: collective code ownership. Each individual may have special skills, but no overspecialization ("this is not my code, I can't modify it").
- Estimate the amount of work for each user-stories (e.g. planning pocker):
  - Use arbitrary time unit: "story points" ($\neq$ man.day), each team may have a different notion of story point.
  - Too large story $\Rightarrow$ ask the PO to split it
- Split user stories into technical tasks (e.g. "write the HTML", "write CSS", "add entry in DB", ...)

Role: Scrum Master

- Role: facilitator, protect the team against distraction
- Experimented in team management and/or Scrum
- May be a developer, or not
- Make sure everybody work in good condition
- Works for the team, not the other way around

Outline of this section

Steps of a sprint

1. Sprint planning: discuss/agree on the sprint backlog
2. Development, continuous testing & integration. Daily scrum meetings.
3. Demo
4. Retrospective: discuss and improve
5. goto 1 (forever?)

Sprint planning (1 meeting, a few hours)

- Prepared by the PO: product backlog = set of user-stories, sorted by priority
- Evaluation (story points) of first stories
- By experience, 1 sprint = X points $\Rightarrow$ stop when $\sum$(story points) $= X$
- Example discussion:
  - Team: we evaluate this story to 40 points.
  - PO: that's too high!
  - Team: that's not your business ...
  - PO: OK, this is high priority but it's too long for now, I'm changing its priority.
- Or:
  - PO: can we reduce the scope of the story to make it fit in 20 points?
  - Team: yes, for example we can make a rough UI and finish the business logic for 20 points.
  - PO: OK, I'm splitting the story, we'll make a nice UI in the next sprint.

End of sprint planning

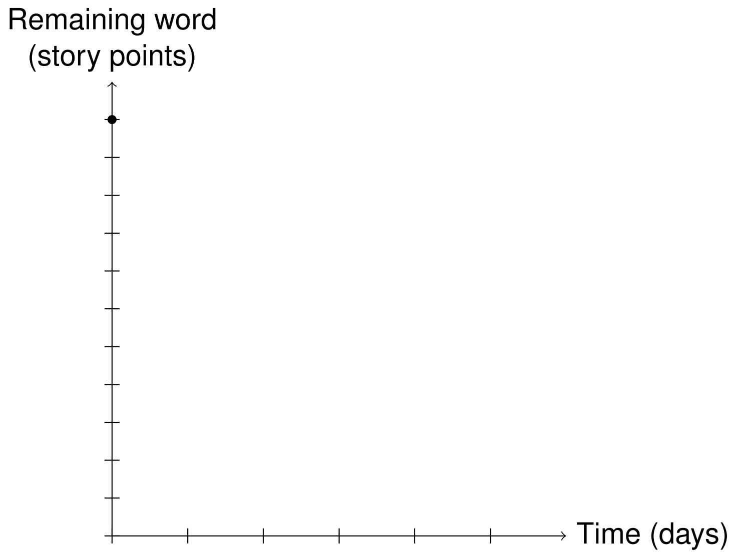
- Team split user-stories (user-spec) into technical tasks
- Some teams evaluate technical tasks in hours of work. Some just split the story points of the story.
- Make a nice scrum board!
- Initialize the burndown

End of sprint planning

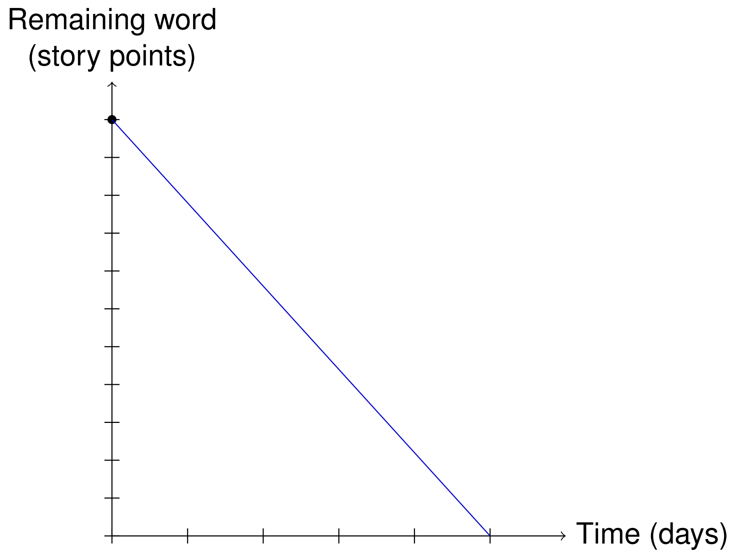
- Team split user-stories (user-spec) into technical tasks
- Some teams evaluate technical tasks in hours of work. Some just split the story points of the story.
- Make a nice scrum board!
- Initialize the burndown

err, what's that?

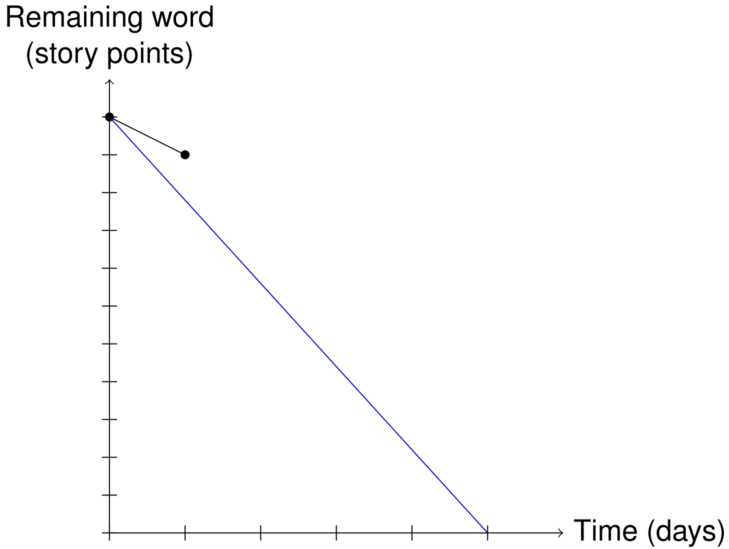
The Burndown Chart



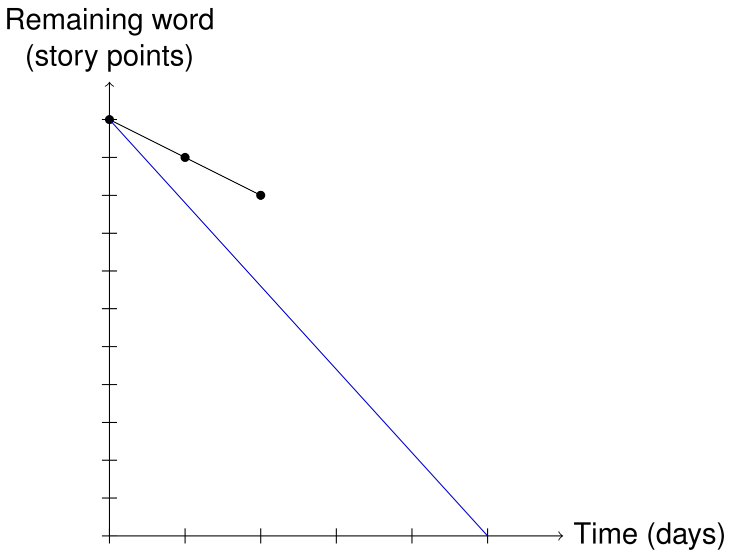
The Burndown Chart



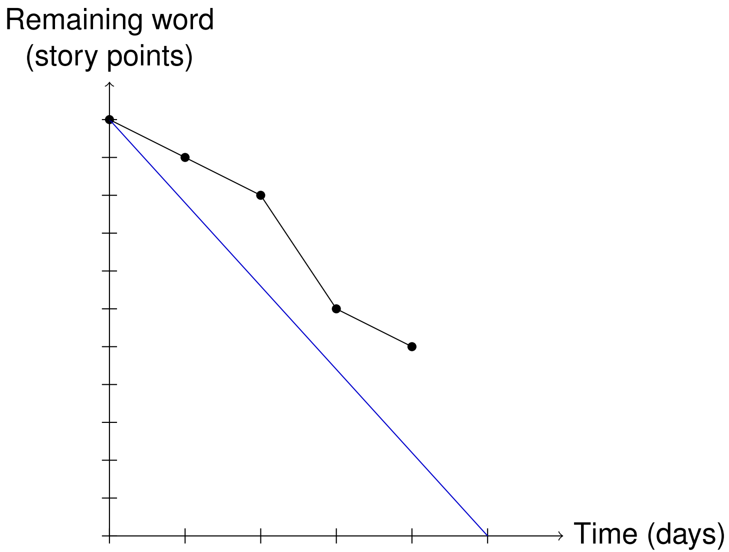
The Burndown Chart



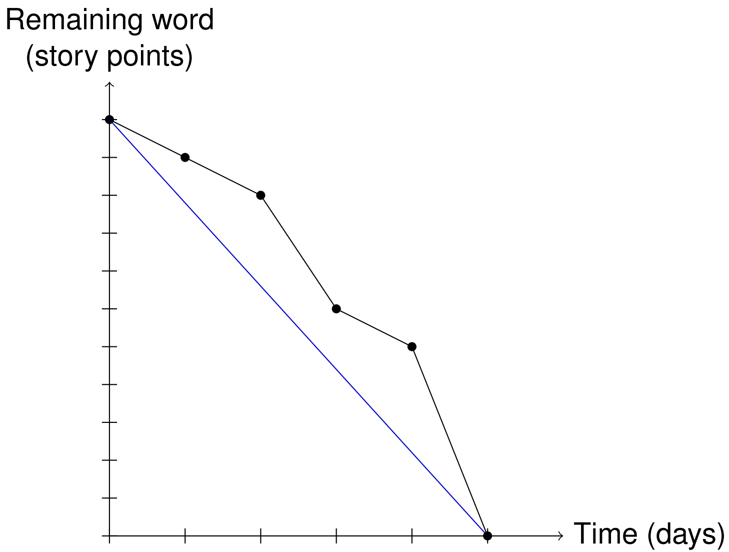
The Burndown Chart



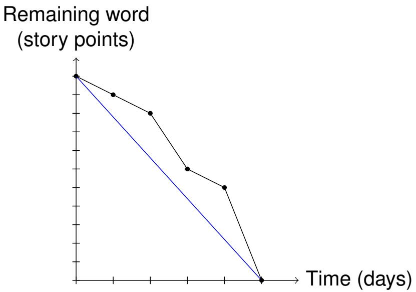
The Burndown Chart



The Burndown Chart



The Burndown Chart



- 1 point every day
- Helps medium-term planning (are we late?)
- ⇒ helps having constant pressure all along the sprint (≠ “Cool, we’re on time” followed by “sleepless night before release”)
- Count remaining work, not work done. Why:
  - Task estimated to 7
  - 5 points done
  - 4 point remaining (sorry, we underestimated the task!)
- Hopefully a decreasing function

Sprint Planning

- Checklist:
  - Sprint planning ready
  - Stories evaluated and divided into technical tasks
  - Burndown initialized (first point)
  - All this is clearly visible (displayed on the wall)
- On your marks, set, ... Go!

Sprint (development)

- Write “good” code. Not specified by Scrum, but usually:
    - Pair programming
    - TDD
    - Continuous integration
- No compromise on quality (cf. technical debt)
- Daily Scrum (short meeting) every day
- Keep a sustainable but sustained pressure. 40h/week max.

Daily Scrum, aka “daily stand up”

- Short meeting ($\approx$ 15 min)
- Stand up meeting ($\neq$ everybody reading mail on laptop while one guy talks)
- In front of the Scrum board
- Every developer answers 3 questions:
  - What did I complete last day?
  - What will I complete next?
  - What’s blocking me? ($\rightsquigarrow$ do I need help?)
- Time-boxed (responsibility of Scrum master). Examples:
  - Scrum master: “Interesting point, but we’re getting technical. Can we continue offline?”
  - Scrum master: “Time is out, we’re stopping. Tomorrow, let’s be quicker so that everybody gets time to talk”
- Move post-its to “ongoing” and “done”
- Update Burndown

“Moving post-its”

- Move from “todo” to “ongoing”:
  - Be careful, your problems are in the “ongoing” column.
  - Some methods (e.g. Kanban) limit the number of items in the “todo” column ($\rightsquigarrow$ “I can’t start this, we need to finish another task before I do”).
- Move from “ongoing” to “done”:
  - Some team attach a “definition of done” to each story, some have a project-wide definition.
  - “done” is “done”: implemented, tested, nothing left to do. 95% of done is not “done”.
- Some team add other columns (e.g. “to review by PO”)

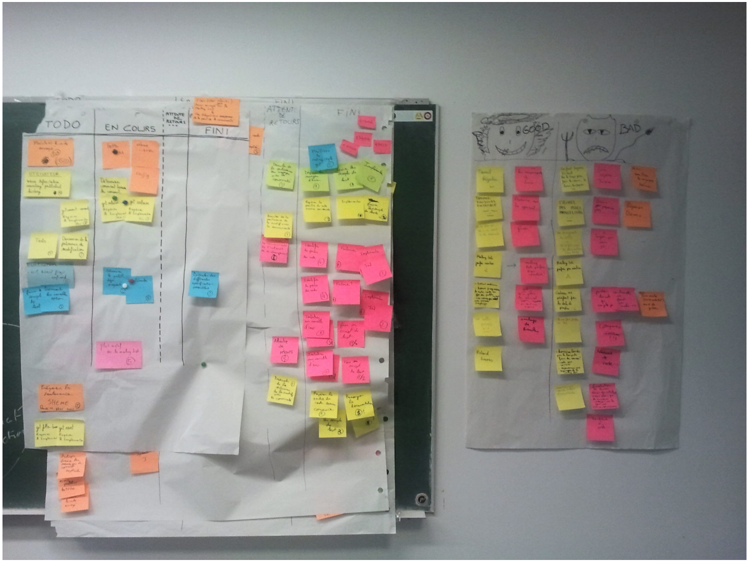
End of Sprint Demo/Review

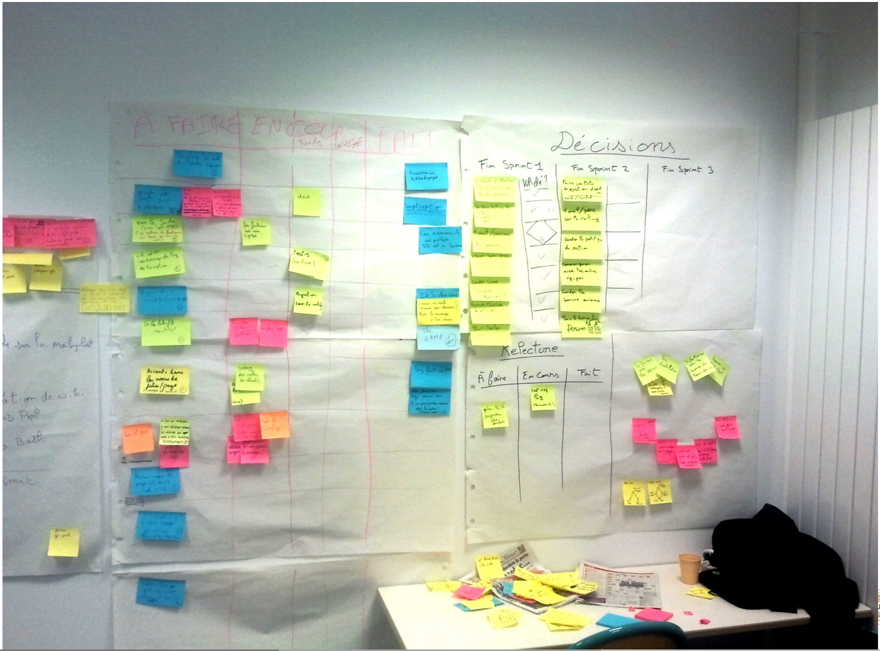
- Demo of working software to the client and/or PO
- “Hey, look how clever my code is” is not a demo: show the value for the client.
- Encourages end-to-end implementations (e.g. business logic without UI $\rightsquigarrow$ no demo)
- “Demo effect” does not exist: test enough before, not during demo.
- Demo in front of other team: inform other team of what you’re doing, be proud of your work.
- Get feedback (helps for next sprint planning)
- Stories are validated by PO (or not)
- Measure velocity (number of story points validated) $\Rightarrow$ gives an idea for next sprint

Retrospective

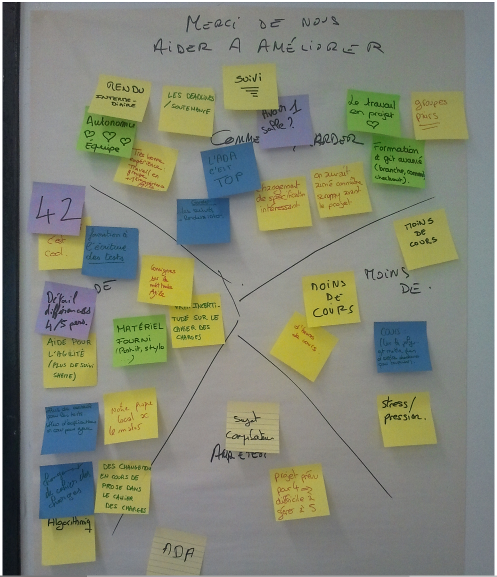
- One meeting: how the sprint went? how to do better?
- Usually in several phases:
  - Information gathering: everybody says/writes what comes to mind
  - Organize ideas (e.g. “good/bad”, “helped us/handicaped us”, ...)
  - Decisions for next sprint (concrete items: checklist if possible)
- Fundamental notion of Scrum and Agile methods: continuous improvement

Retrospective




À FAIRE
EN COURS
FAIT
Décisions
Fin Sprint 1
Validé ?
Fin Sprint 2
Fin Sprint 3
Refactore
À faire
En Cours
Fait

Continuous Improvement


Merci de nous aider à améliorer
Rendu intermédiaire
Les deadlines / soutenance
Suivi
Avoir 1 salle ?
Le travail en projet ♡
groupes pairs
Autonomie ♡ ♡ ♡ Équipe
Très bonne expérience : Travail en groupe
Comme
garder
L'ADA c'est TOP
Formation à git avancé (branche, commit, checkout).
42
c'est cool.
formation à l'écriture des tests
Garder les sujets
changement de spécification intéressant
On aurait aimé connaître scrum avant le projet
Moins de cours
De
Consignes sur la méthode Agile
Défaut différence 4/5 pers.
Incertitude sur le cahier des charges
Matériel fourni (Post-it, stylo ...)
Aide pour l'agilité (plus de suivi SM/ETE)
Moins de cours
Moins de .
d'heures de cours
Cours
Notre propre local
Sujet Compilateur
Arrêter
stress / pression.
changement de cahier des charges
Des changements en cours de prose dans le cahier des charges
Projet prévu pour 4 → difficile à gérer à 5
Algorithmique
ADA

Celebrate

!

Sprint Planning

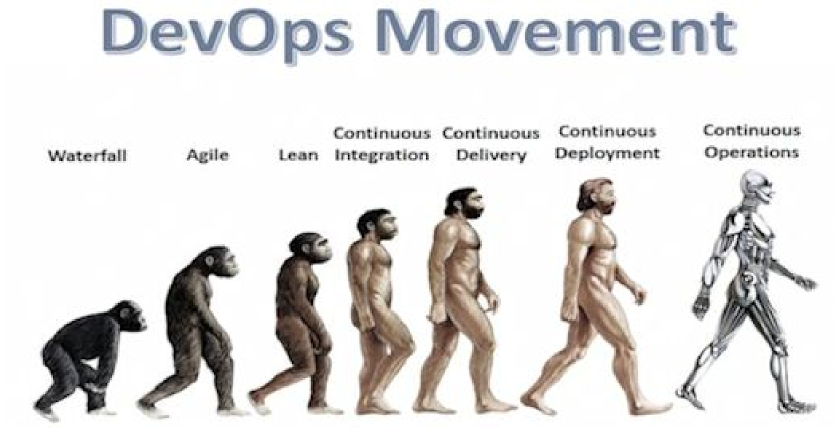
Same for next sprint

Steps of a sprint: summary

1. Sprint planning
2. Development and daily meetings
3. Demo
4. Retrospective
5. goto 1 (forever?)

Outline

Scrum Already Has-Been?



https://devops.com/devops-killed-developer-star/

Trend: Short Cycles

- Traditional project: release every few years
- Scrum: release every few weeks
- DevOps: release every few hours/minutes?

Eliminate Waste

- Scrum is a good <u>tool</u> to improve efficiency
- Still some project management overhead: what's useful? what's waste?
- What's next?

No Silver Bullet

- Consider each method as a tool: use the appropriate tool for the task

No Silver Bullet

- Consider each method as a tool: use the appropriate tool for the task

(But remember, when you have a hammer,
everything looks like a nail)

No Silver Bullet

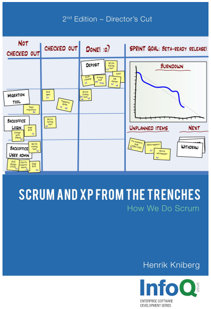
- Consider each method as a tool: use the appropriate tool for the task

  (But remember, when you have a hammer, everything looks like a nail)

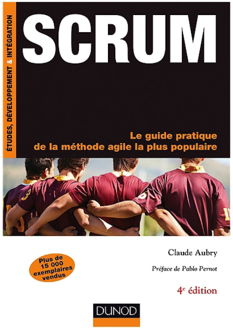
- Don't apply without understanding
  - ~~The book says "meeting every day", I'm the boss, so do one meeting everyday~~
- Don't over-interpret
  - ~~I don't write docs because I'm agile~~
  - ~~I can't give you any price because it's Scrum~~
  - ~~It's Scrum, there's no boss, it's anarchy~~
- Adapt
  - Not too often: it may take time to get the benefits
  - Often enough: don't keep doing the same mistakes

Outline

Recommended Reading



- Pleasant to read
- Straight to the point (written in 1 week-end)
- Pragmatic
- French or English


ÉTUDES, DÉVELOPPEMENT & INTÉGRATION
SCRUM
Le guide pratique de la méthode agile la plus populaire
Plus de 15 000 exemplaires vendus
Claude Aubry
Préface de Pablo Pernot
4e édition
DUNOD

- Another classic
- More complete
- (I didn't read)

- `agilemanifesto.org` a few principles
- `extremeprogramming.org` old site, but still a reference/source of inspiration on many aspects
- "Getting Real" par 37 signals (lean)
- "Lean Software Development: An Agile Toolkit" (Marie & Tom Poppendieck)
- And also:

`programming-motherfucker.com`

5 minutes must see

Scrum Master - Funny movie about The Power of Scrum

The power of Scrum,
`https://www.youtube.com/watch?v=P6v-I9VvTq4`

Club Agile Rhône Alpes

- `https://www.clubagilerhonealpes.org/`
- Coding Dojo
- Conferences ("Agile Grenoble" = ♥, "Agile Lyon" = probably good too ;-))
- Agile Play Ground

Outline

TD: ~~Lego4Scrum~~Paper4Scrum

- Idée : construire une ville en papier en applicant Scrum
- Backlog :
  - 1 lotissement de 5 bâtiments sans étage
  - 10 bâtiments à un étage
  - Un magasin
  - Une école
  - Un hôpital
  - Une maternelle
  - Un arrêt de bus
  - Un jardin public
  - Un pont
  - Un statue de Douglas Engelbart
  - Un aéroport
  - Une mairie
  - Une station essence

Déroulement

- Équipes de 6-8 personnes
- Préparation du backlog
- 3 mêlées (time-box)
- Ranger ;-)

Préparation des équipes

3 min

- Organiser les équipes (6-8 personnes)
- Qui est scrum master?
- Un étudiant peut être PO, sinon c'est votre enseignant
- Préparer la salle (3 tables/équipe)

Estimation, backlog produit

7 min

- Estimer et prioriser chaque story
- Définition de “fait” ?

3 mêlées

- Sprint planning : 3 min
- Sprint : 7 min
- Demo/revue : 3 min
- Retrospective : 2 min

La fin ...

- On nettoie autour de chaque groupe
- On range la salle
- Retour sur l'activité et l'application de SCRUM:
  - Plus de ...
  - Moins de ...
  - Arrêter ...
  - Commencer ...
  - Garder ...