Integrated Software Project
Fundamentals in project management

Jean-Louis Binot

Objectives of this session

- Explain why project management is important for Information Technology.
- Provide exposure to professional expectations concerning IT applications & projects.
- Present the role of project manager and the fundamentals of project management.
- Explain the project management deliverables required for the Integrated Project, taking into account the selected methodology.

Agenda

1. IT Applications and IT Projects
2. Challenges of IT Projects
3. Organizing an IT project
4. Project Management & deliverables

Why Project Management ?

In a company, IT supports the business through ***IT applications*** and ***IT services.***

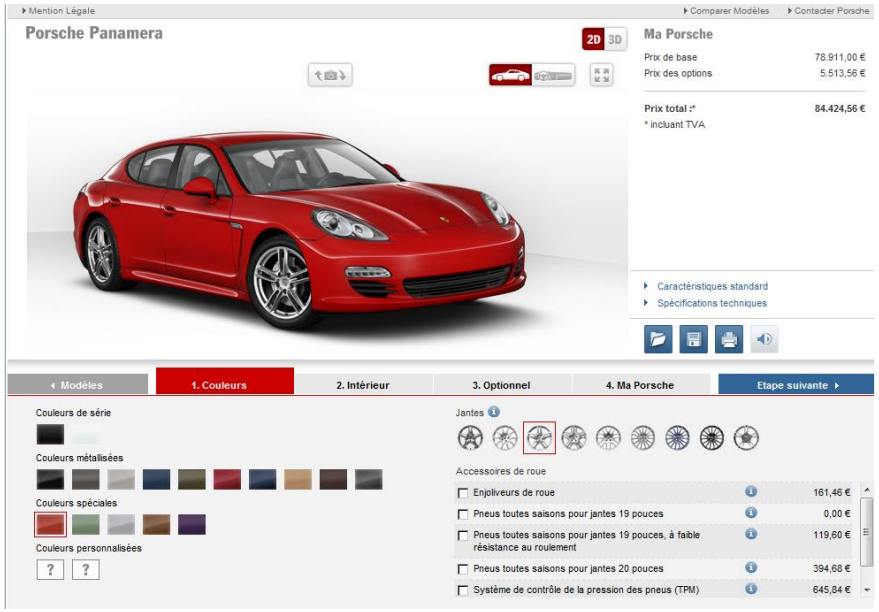
- Implementation of a new IT application, or rolling out of IT services, is done through *IT* ***projects.***
- IT projects are difficult to execute, and require a specific discipline to succeed: ***Project Management.***
- Project management is a management activity, which can involve large teams and budget. ***The role of project manager*** may be part of a typical IT career path.

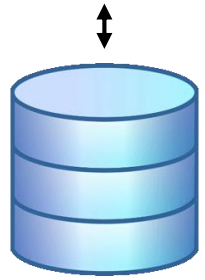
An IT Application

- Is a computer software designed to help the user to perform specific tasks.
- Automates business processes in order to produce business benefits, in four possible categories:
  - Increase sales and hence revenues;
  - Reduce costs and hence increase profits;
  - Increase quality of product / service & customer satisfaction;
  - Ensure compliance to regulations.
- Needs to store, retrieve, and process data in order to achieve that automation.

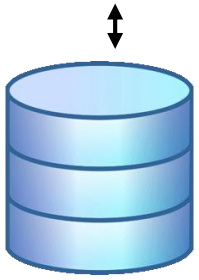
which ones ?

Example 1: a car configurator





Product DB



Customer DB

Benefits: increase sales

Car configuration
Quote
Test drive request

Test drive appointment

Challenges
- Project size (>5M€; many concurrent users)
- Data accuracy
- Performance and availability
- Graphical interface (3D, accuracy)
- Multi country / language (30 countries)
- Maintenance (25 people)

Example 2: a nationwide service architecture for health insurance

External PC user

Mobile devices

Web site (Mymut)

SOA Gateway

ESB (Enterprise Service Bus)

Data isolation layer (cache database)

AS/400
Legacy RPG systems

Linux

AIX

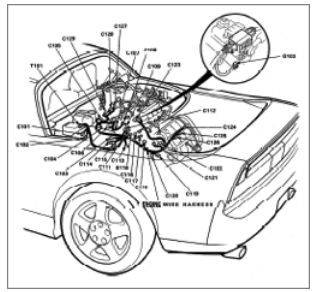
Identity and access management
Monitoring and logging
Tracking and tracing
Statistics
Error mapping
Transactional state management

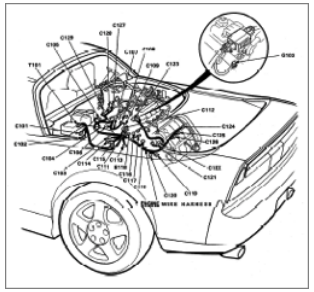
Benefits:
- increase time to market (revenues)
- reduce costs (IT)

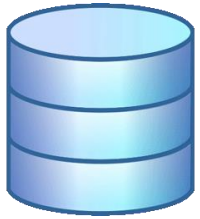
Challenges
- Project size (>5M€; 5 “Muts”, several 100K users)
- Data accuracy and confidentiality
- Performance and availability (8000 t/m peak)
  - at front end (caching, optimization)
  - at backend (DB, ESB, network, server …)
- Data consistency between cache and backend

Example 3: technical document mgt

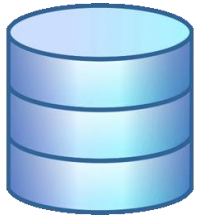
Technical data
text & images







Automatic
Translation



Technical doc

Local updates

Any repairer
in Europe

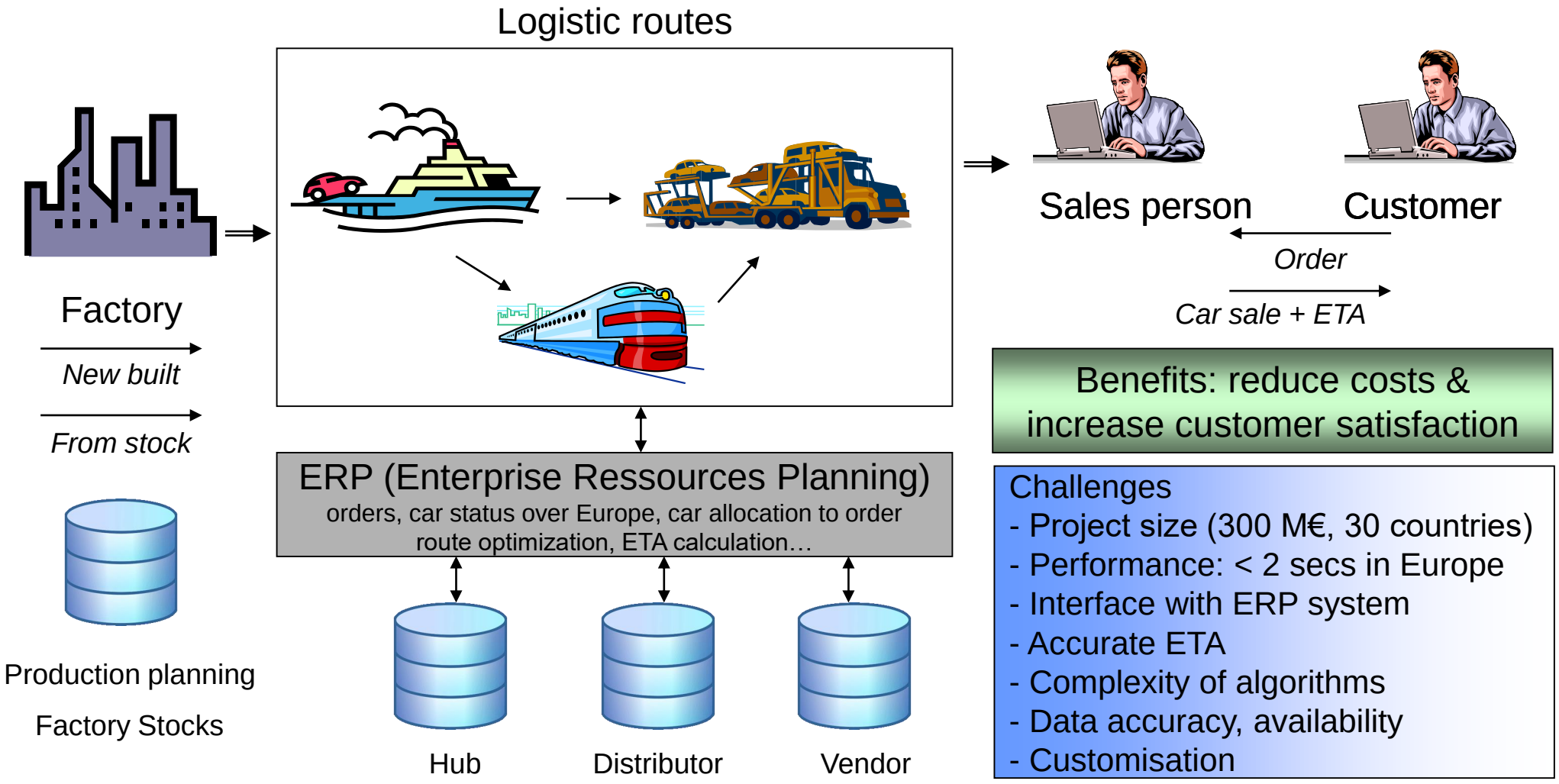
Engineering Data
(3 languages)

European Data
(>20 languages)

Benefits: increase quality
& European compliance

Challenges
- Project size (>10M€; huge # of users)
- Interactive graphical diagrams
- Automated search & link with repairer tools
- Automated translation in many languages
- Performance and availability
- Data accuracy

Example of a large IT project: Vehicle order mgt



IT applications support the business

All important IT application issues will have business impacts.

- Application availability & reliability matters (must “keep the lights on”).
  - Bugs have business consequences.
- Application performance matters.
  - Delays also have business impact.
- Respecting deadlines matters.
  - Many deadlines are “business critical”: they can’t be moved without impact.
- Data accuracy matters.
  - Impact of data errors goes from inconvenience to critical liability.
- Ease of maintenance is important.
  - Business will want new functionalities. Maintenance cost can eat 60% of IT budget

Agenda

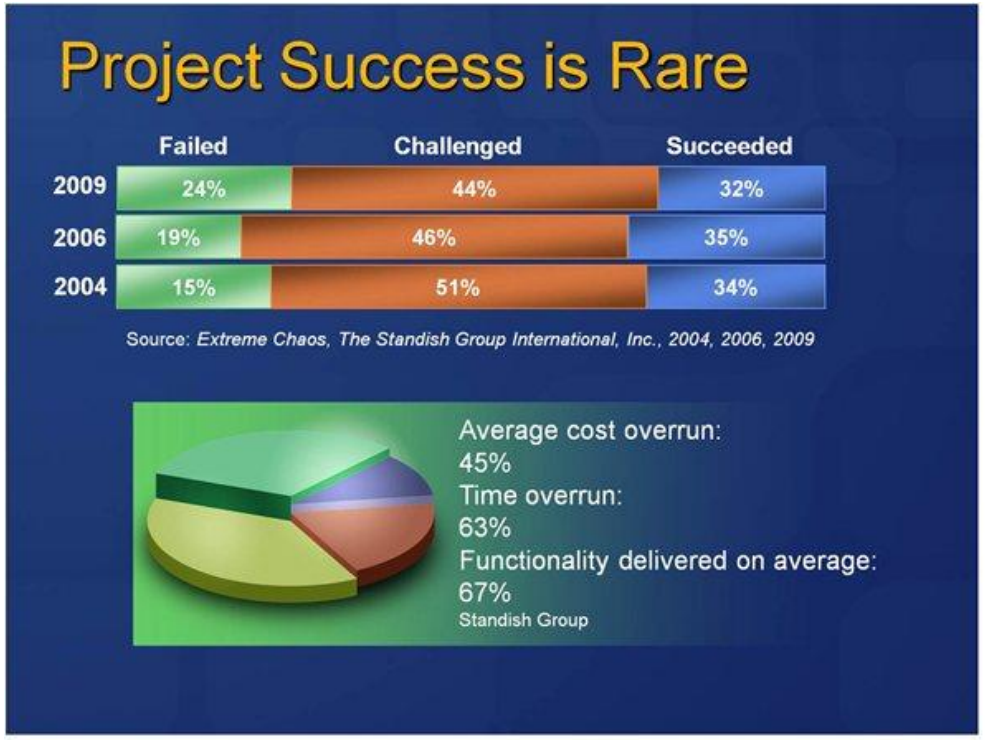
1. IT Applications and IT Projects
2. Challenges of an IT project
3. Organizing an IT project
4. Project Management & deliverables

The difficulty of delivering IT projects

The Standish group (IT project management research and consulting firm) delivers each year a “Chaos report” surveying project success rate.

The first report (1995) was a landmark:

- 16% success;
- 31% failure;
- 51% challenged.



More recent Standish data

Slight improvement in project success rate, but still many failures !

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.

CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

Better results with agile methodologies for small projects.

Source : Standish Chaos report 2015

Why so many project failures ?

- Requirement issues.
- Architectural or technical mistakes.
- Lack of organization or discipline.
- Difficulty of integrating / managing all aspects.
- Difficulty of managing consensus within team.
- Competition for scarce budget, resources, and time from other projects.
- Difficulty of managing consensus between stakeholders.
- Management pressures.
- Psychology and politics.
- …

You will face at least these challenges

Agenda

1. IT Applications and IT Projects
2. Challenges of an IT project
3. Organizing a software development project
4. Project Management & deliverables

What is a project ?

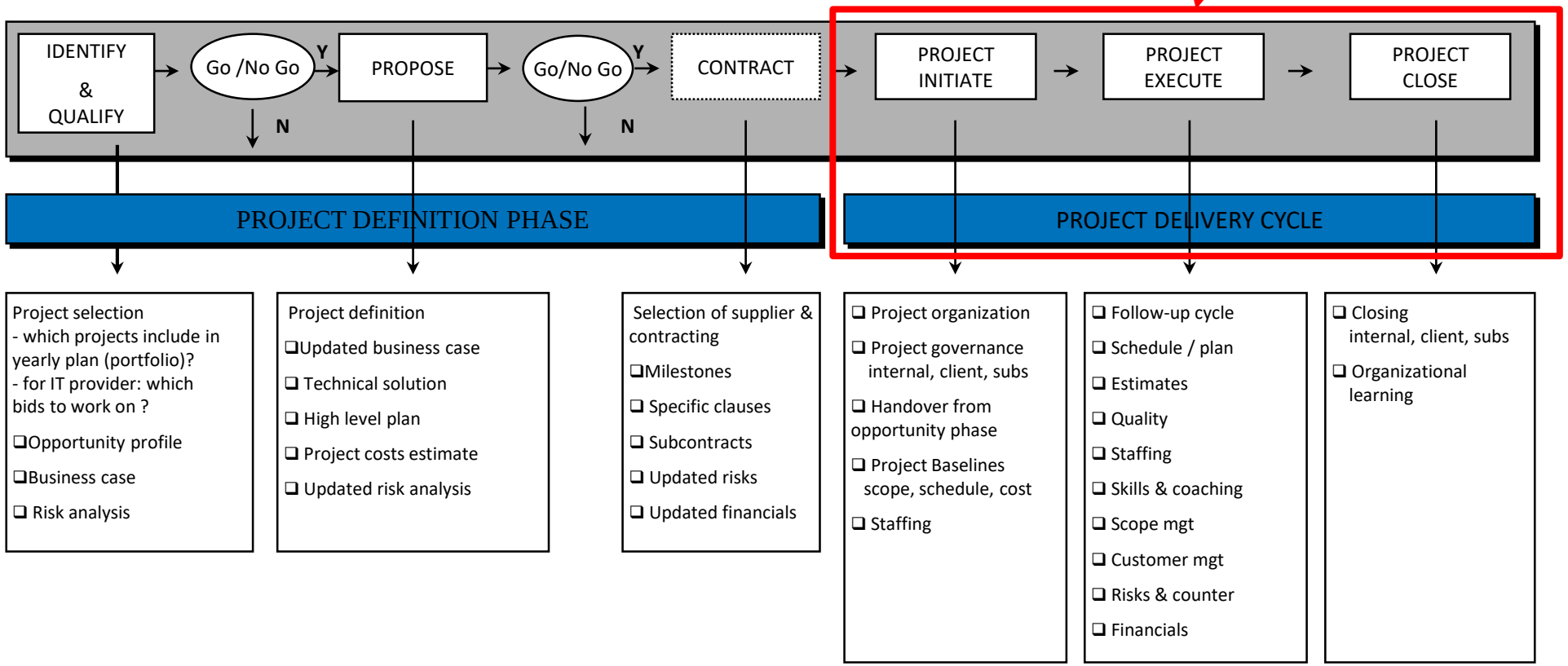
- A project is a temporary endeavor undertaken to create a unique product or service (PMI).
  - Temporary: it has a definite start and a definitive end.
  - Unique: it is not repetitive (not production nor maintenance).
- A project has three main constraints :
  - Time (respect of deadline / schedule);
  - Cost (respect of budget, which can be “fixed price”);
  - Scope (what needs to be delivered).

Quality (of the delivered product) results of the interaction between these constraints.

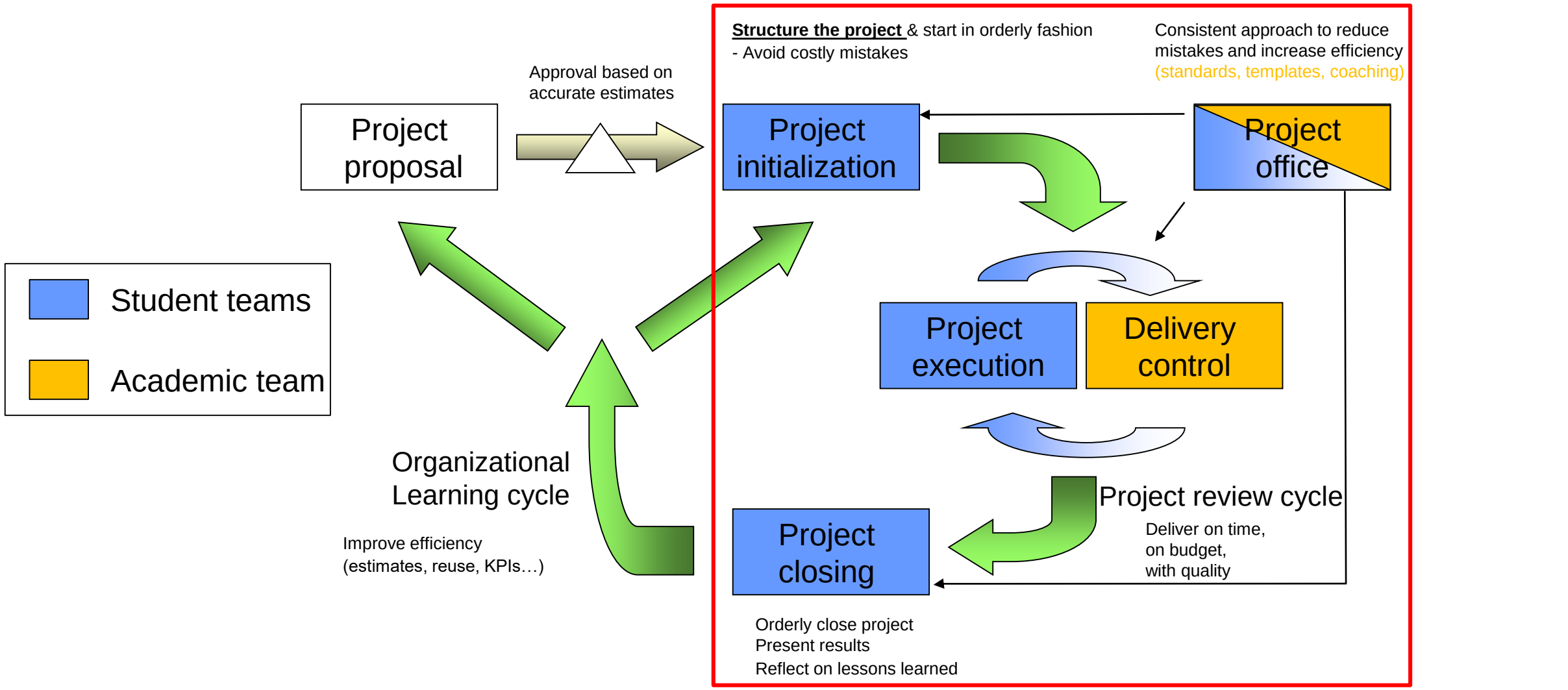
- A project involves many additional domains, such as :
  - Human resources, communication, risks, procurement, integration …

The project life cycle

The scope of this Integrated Project covers the Project Delivery phase (with a limited set of deliverables).



Project delivery follows a systematic delivery cycle



Software development activities

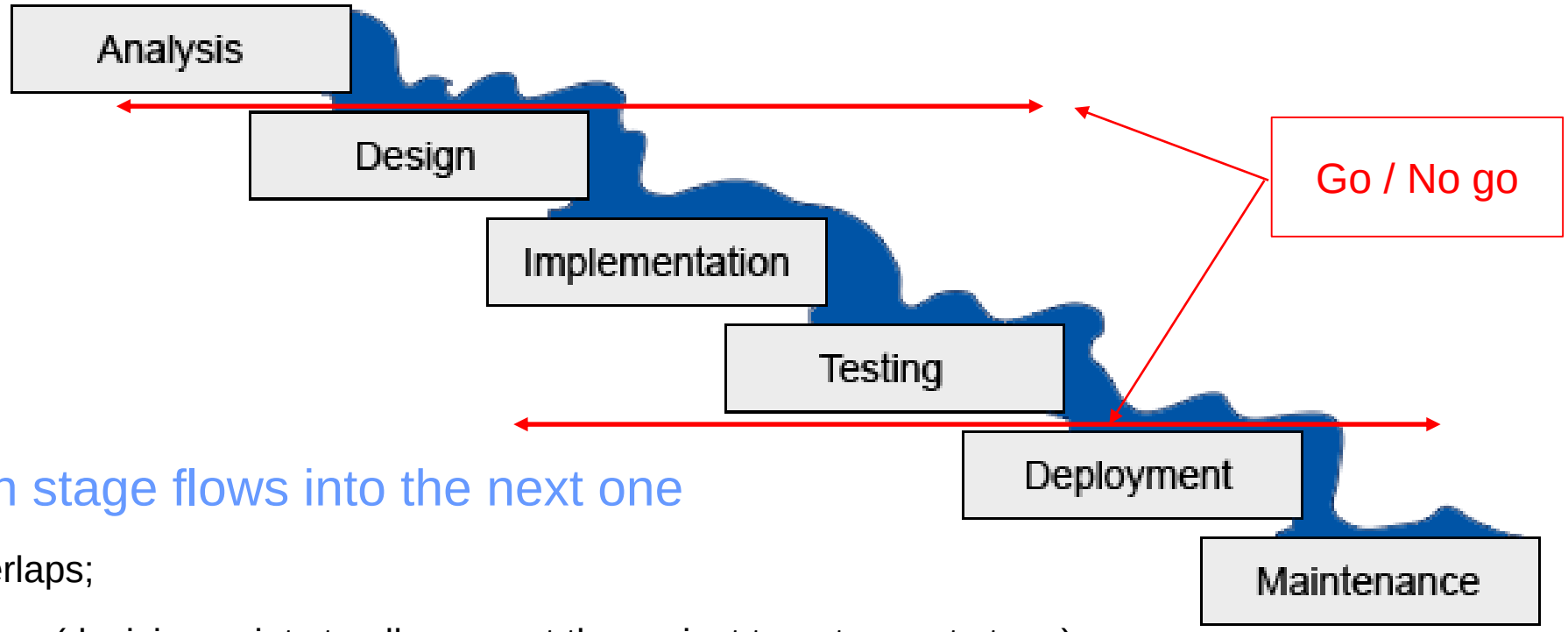
A software development project executes specific activities, classified as follows:

- Specifying the requirements.
- Analyzing the problem to be solved.
- Finding solutions: designing the architecture of the system, designing algorithms to solve particular problems, …
- Implementing the result.
- Testing the product.
- Deploying the product in its target production environment(s).
- Maintaining the software product (correcting bugs, enhancing functionalities, …).

Development process

- These activities must not necessarily be performed in sequential order.
- A development process is an organization of software development activities:
  - When are the activities performed? For how long? In what order? By whom?
  - With what kind of expected outcomes/results (deliverables)?
- A software development project is a disciplined execution of the development process.
  - That minimizes the risks of not delivering a suitable product;
  - That produces high quality software corresponding to user needs;
  - That satisfies as best as possible the time, cost and scope constraints and manages resources efficiently.
- To organize the development process, different methods are available. We will see :
  - The classical (waterfall) method;
  - Iterative methods.

The waterfall process

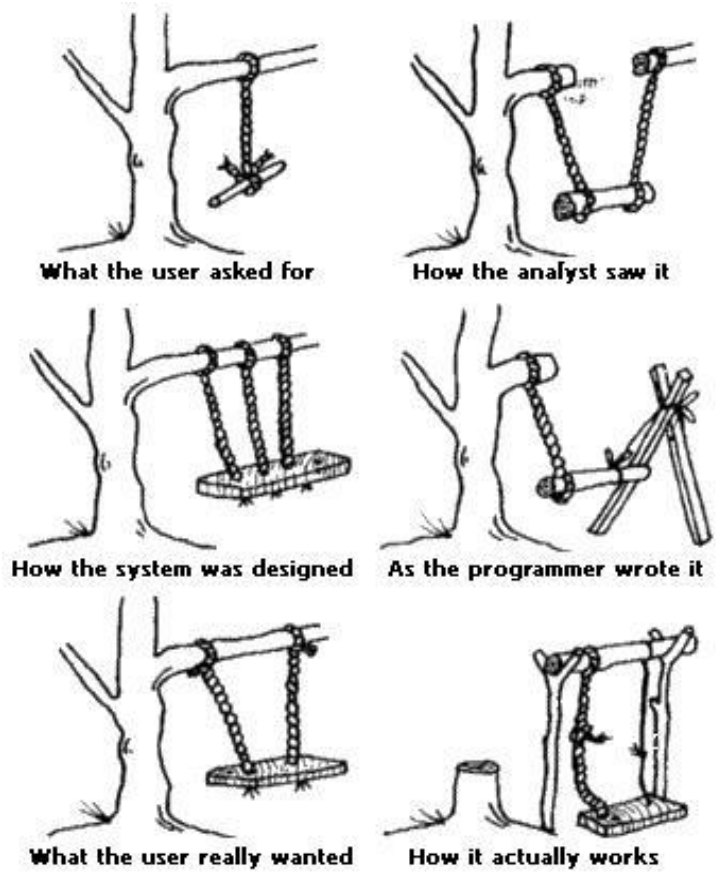


- Results of each stage flows into the next one
  - with possible overlaps;
  - with go/no go gates (decision points to allow or not the project to enter next stage).

Comments on the waterfall approach

- If flaws are detected the corresponding activities have to be reworked, leading to:
  - Missing deadlines, possibly release of flawed products, exceeding allocated resources.
- The waterfall approach is well adapted to disciplines such as civil engineering:
  - Requirements are reasonably easy to formulate;
  - The cost of modifying products once realized is very high;
  - Powerful tools are available for checking the quality of intermediate results.
- It is less adapted to software engineering (is actually seen as culprit of project failures)
  - Specifying requirements is a non trivial problem;
  - Requirements can change during the lifetime of the project;
  - Consequences of bad design choices sometimes become only apparent at a late stage;
  - In particular some correctness and performance issues are only detected during testing.

Challenges of the classical (waterfall) approach



Old joke, still true

- Customer may have only a high level view of his needs and be unclear.
- He is often not IT/process literate.
- But he has to accept system and pay the bill.

Challenges for IT project

- Elucidate requirements (first in NL).
- Detect incompleteness, inconsistency, risks.
- Formalize functional requirements and data.
- Obtain sign-off in reasonable time frame.

Examples of project failures

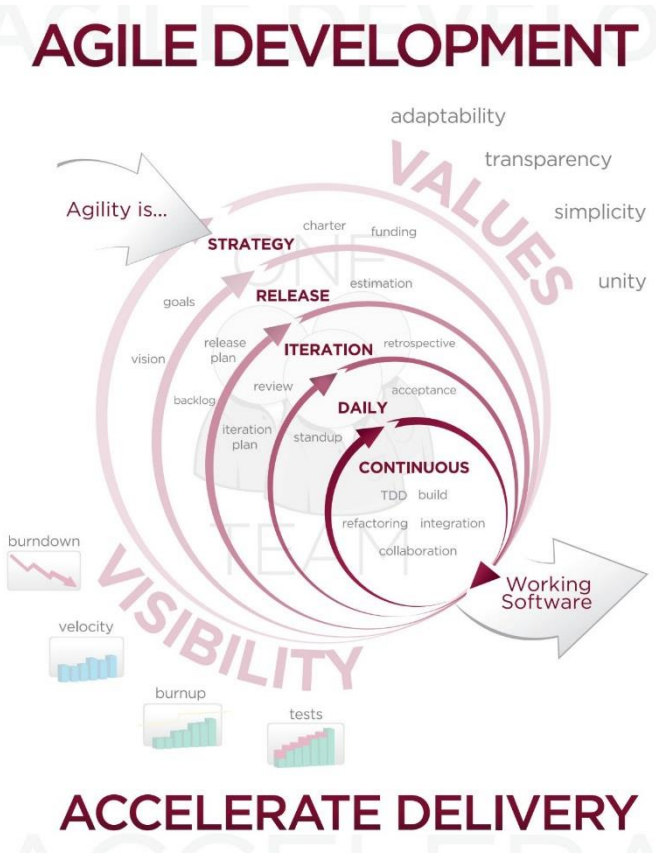
Too much requirement formalization

- Major global credit card company wanted to implement global settlement system including functions of its European partner.
- 30 European people sent to USA to work 2 years on requirements.
- Business lost faith after 2 years of “paper” and cancelled project at huge cost.

Too little requirement formalization

- Customer wanted to migrate automatically a web system and database to new technology; was sure to have required it.
- Service provider was convinced automatic migration was unfeasible; was sure to have agreed manual solution.
- Documents existed but no clear evidence of agreement.
- Customer refused to accept systems; several months delay and 1 M€ over-costs needed to agree and implement compromise.

Iterative methods : agile development



Manifesto for Agile Software Development in 2001: promote

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

APPROACH

- The project is built as a sequence of of small iterative cycles.
- Using a time-boxed approach : each cycle has a fixed, short duration.
- Each iteration delivers a working product.
- During each cycle, the team works intensely through all software activities.
- The team is cross-functional and self-organizing.
- Face to face communication and user involvement are privileged.

Agile versus classical

Agile : adaptive

+ Valuable user feedback is obtained early.
+ Easier to adjust to changing realities : involved users may decide scope changes as they go.
+ A functional version of the product is available at each iteration for evaluation or demonstration purpose.
+ Risks of not meeting requirements or exceeding budget are mitigated, as objectives can be adjusted after each iteration.

- Difficult to describe future tasks.
- Difficult to freeze contractually agreed content.

Classical : predictive

- Main user feedback obtained late, when product is tested.
- Signed-off project definition, budget and plan make it difficult to change direction.
- The product is only available at the end of the project, which makes it difficult to detect unwanted deviations.
- Requirement deviations are detected late and budget deviations tend to accumulate.

+ Structured plan allows to predict future tasks.
+ Agreed scope describes future content.

Agile versus classical

Agile : adaptive

Well suited for :

- Small rapid software development projects.
- Larger software projects divided into smaller subprojects.

Not well suited for:

- Disciplines where refactoring is too expensive or impossible (e.g. civil engineering).

Classical : predictive

Well suited for :

- Disciplines where requirements must be fixed, are reasonably easy to formulate, and the cost of refactoring is too high (e.g. civil engineering).
- Software projects where requirements are stable or cannot be modified easily : legal requirements, multi-country rollouts …

Not well suited for:

- Rapid software development.

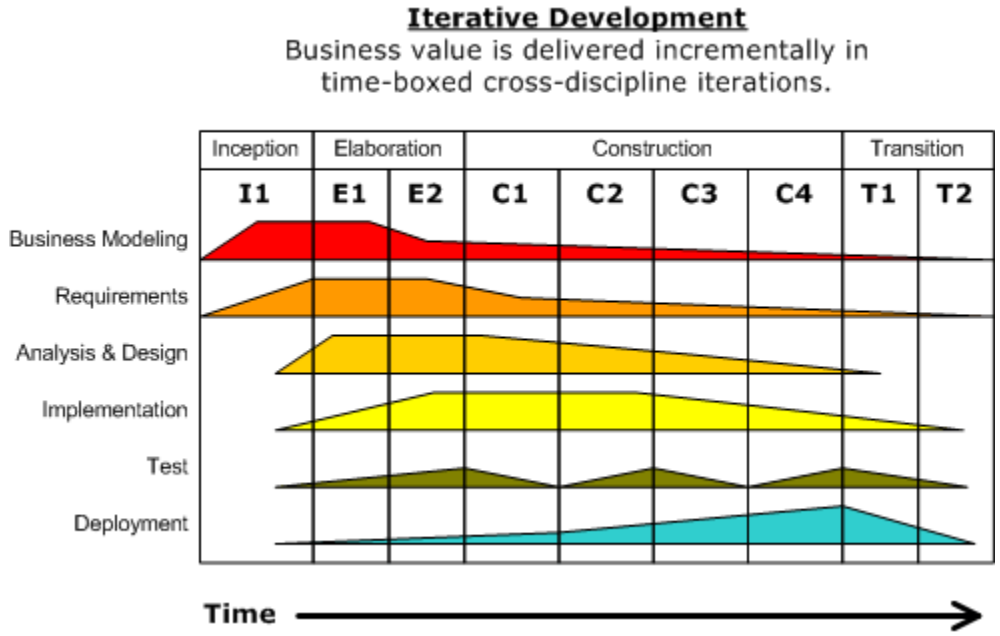
Iterative example 1: the Unified Process

The development cycles are structured in four phases:

- Inception:
  - Define the main goals of the project.
  - Estimate the main risks.
  - Decide whether the project is worth being continued.
- Elaboration:
  - Define a base architecture for the product.
  - Specify detailed requirements.
  - Estimate the costs and plan the schedule.
- Construction: build the product.
- Transition: make the product ready for commercial release.

Iterative example 1: the Unified Process

- Phases are divided into time boxed iterations
  - This can be true of all phases, but mostly Elaboration, Construction and Transition.
  - Each iteration results in an increment, which contains enhanced functionality compared to previous releases.



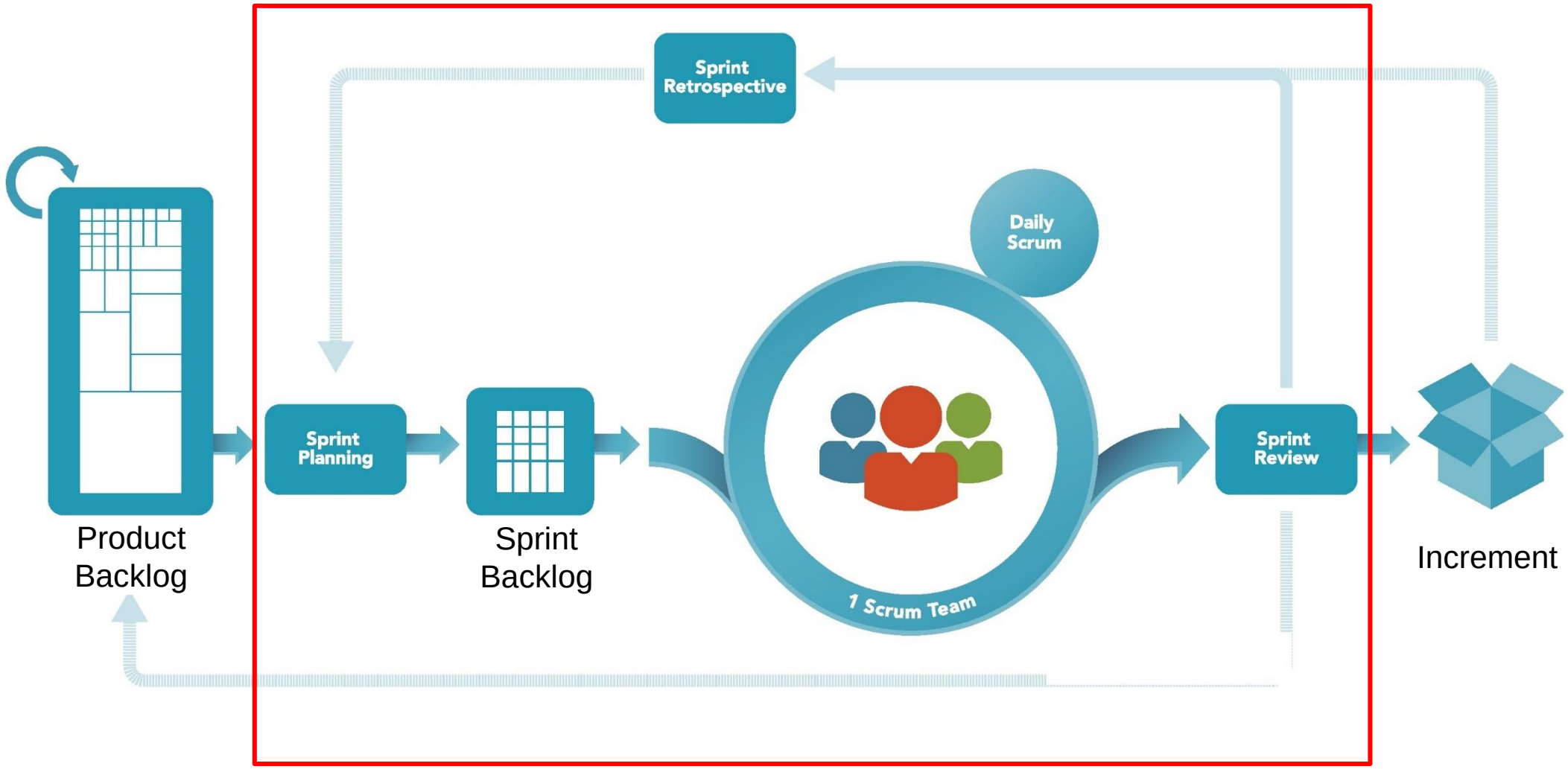
Iterative example 2: SCRUM

This year, we will use an agile methodology based on the Scrum* framework

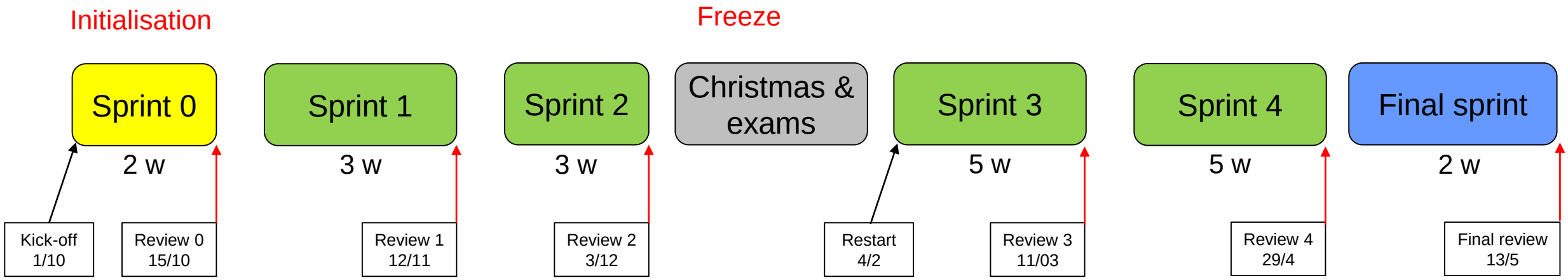
- Scrum : agile methodology standard created by Jeff Sutherland, 1993.
- Designed for small development teams (3 to 9 people).
- Work is divided into actions completed within timeboxed iterations, called *sprints.*
- These sprints are of short, fixed duration (typically 1 month or less). Advantages :
  - The short time horizon limits risks of changes to the definition of what is to be built.
  - It increases predictability and quality by insuring inspection and adaptation every month.
  - It limits cost risks to one calendar month.
- Each sprint is aiming to deliver a done, usable product implement.
- Scrum is not an acronym but a term borrowed from rugby.

Scrum Framework

Project Delivery cycle



Organisation of sprints for this year



This is the basic plan of your project. (please check the dates and number of weeks for each sprint !).

Agenda

1. IT Applications and IT Projects
2. Challenges of an IT project
3. Organizing an IT project
4. Project management and deliverables

Role of the Project Manager

- **Project** : The mission of Project Manager is to **deliver** the result of the project
  - On time.
  - Within budget.
  - With quality.

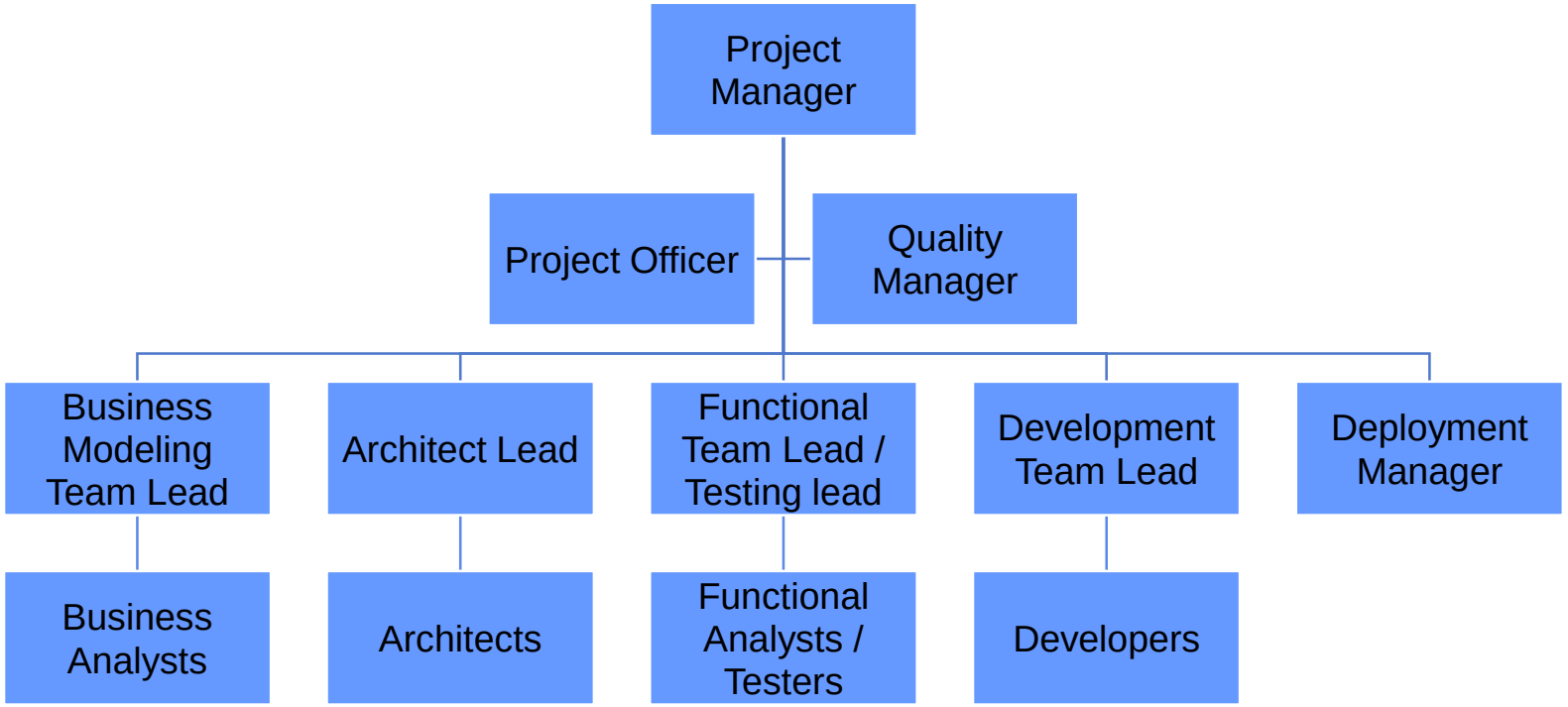
- **Manager**: The Project Manager is not a content expert nor an administrator. His job is to **manage** the project :
  - Planning.
  - Assigning tasks, delegating.
  - Monitoring and controlling.
  - Assessing risks, making decisions, anticipating.
  - Leading and motivating a team.
  - Communicating and negotiating.

The role is evolving in Agile
(less control, more team coaching)
but still important.

Project Management standards

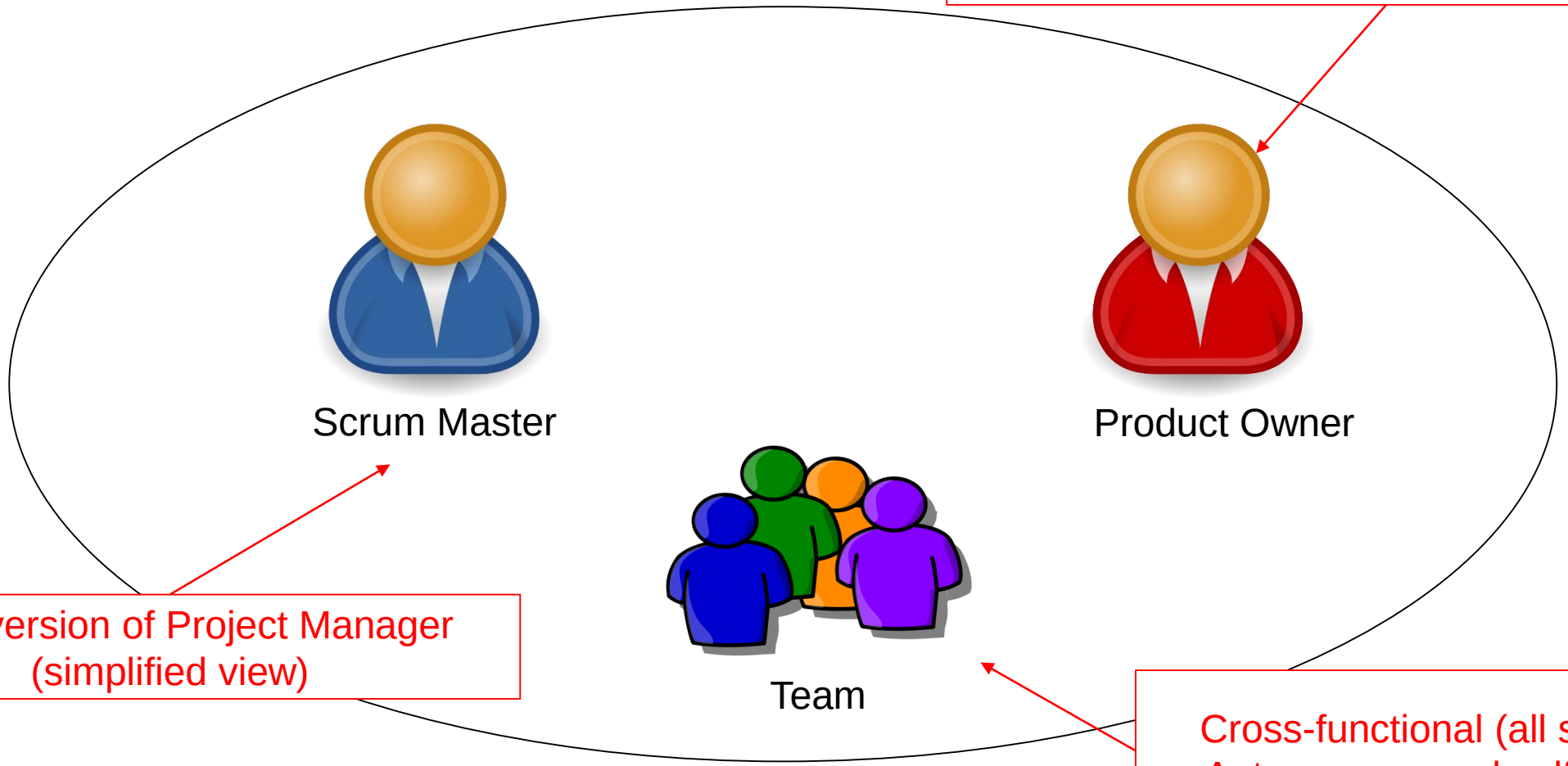
- Standards provide methods, templates… to assist the project manager in properly managing a project.
- Several important international standards exist, among which :
  - PMI: Project Management Institute
    (publishes the PMBOK: Project Management Book of Knowledge).
  - Prince 2 (**PRojects IN Controlled Environments**)
    initially developed for the British government, now recognized as an international standard.
  - Scrum : agile standard
    initially created by Jeff Sutherland, 1993. Now led by the Scrum Alliance and Scrum.org.
- They also provide certification of project managers.

Project organisation and roles – classical view



Agile view : key Scrum roles

Represents the business / customer
Owns selection and prioritization of functionalities



Agile version of Project Manager
(simplified view)

Cross-functional (all skills mixed)
Autonomous and self-organizing

Role of the Project Manager in Agile

- Project Manager => Scrum Master (in first approximation)

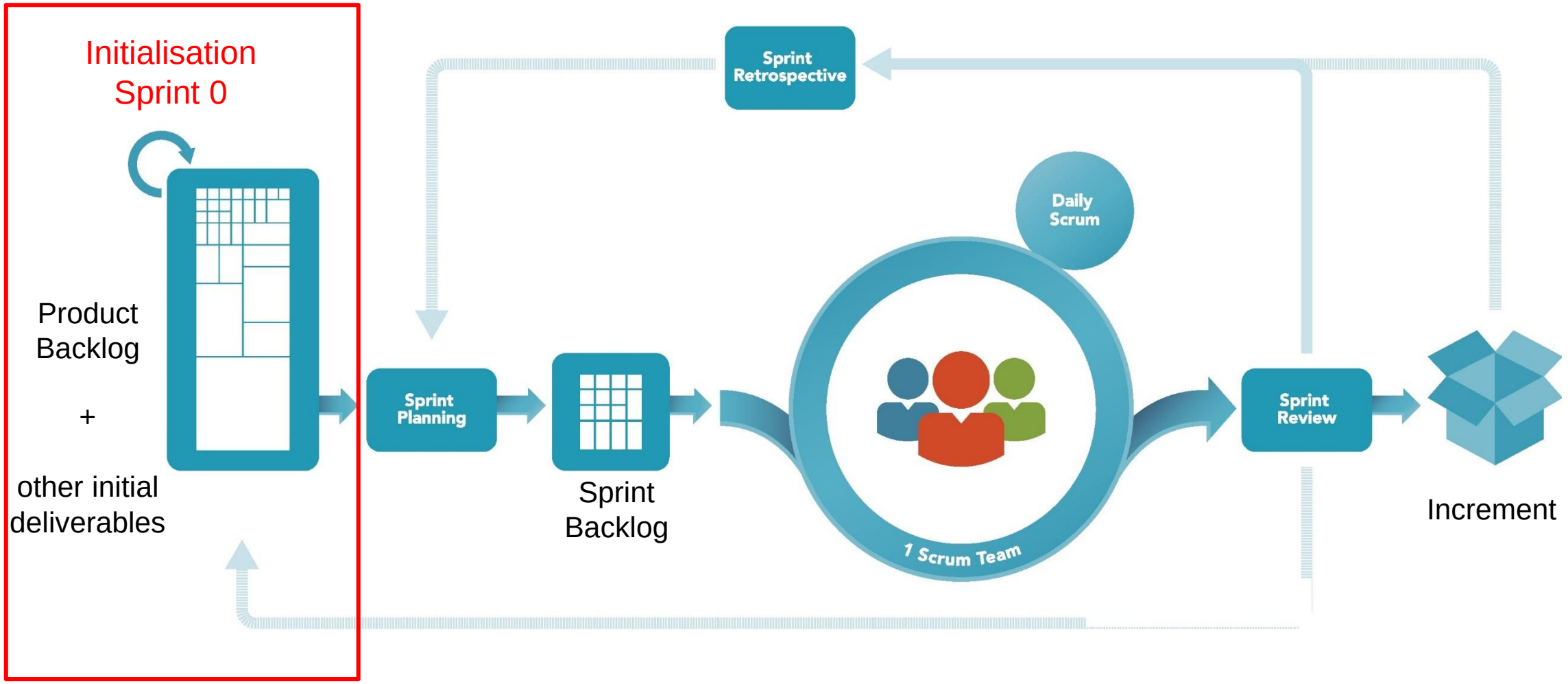
  Main functions :

- Coach: “leading servant”.
  - leader and support of team, facilitator, interface with management.
  - organises, advises, removes obstacles – does not “order” but helps the team to work autonomously.
- Responsible for use of methodology (owns the process).
  - Ensures that methodology is followed by team, status reported …
  - Produces progress reports and metrics, using available tools.

Not applicable: impossible for a team of students

Is part of Integrated Project
Role will be rotated among students (starting sprint 1)

Scrum Framework 1: initialisation sprint



The Product backlog

- Is an ordered list of everything needed in the product.
- Is the only source of the requirements of the product.
  - Will contain all features, functions, enhancements and bug fixes.
- Is owned by the Product Owner (client representative), responsible for its content and prioritization.
- Is evolving during the project (more than in the classical approach):
  - Changes in the product (addition, modification, suppression) are recorded in the Product backlog.
  - Its initial definition only contains initially known and understood requirements.

Product backlog and stories

Requirements are expresses as stories following a story template.

Stories are prioritized.

Type	Key	Summary	Priority	Status	Created	Story Points
Story	TRIAL-1	As an Agile team, I'd like to learn about Scrum	Medium	To Do	26/août/16	5
Story	TRIAL-2	As a product owner, I'd like to express work in terms of actual user problems, aka User Stories, and place them in the backlog	Medium	To Do	26/août/16	3
Story	TRIAL-4	As a team, I'd like to estimate the effort of a story in Story Points so we can understand the work remaining	Medium	To Do	26/août/16	4
Bug	TRIAL-8	As a product owner, I'd like to include bugs, tasks and other issue types in my backlog	Medium	To Do	26/août/16	

A first estimate of the size initial stories should be done in Sprint 0.

Sample data extracted from Atlassian Jira

User stories and Epics

- User stories are short, simple descriptions of a feature told from the perspective of the user or customer.
  - The idea is to move from written formal requirements to team discussion of user expressed needs.
- They typically follow a simple template:
  - *As a <type of user/customer>, I want / can <achieve some goal> so that <some reason>*
- Large user stories are called Epics.
  - As a seller of books online, I want to give users the possibility to manage their accounts so that they can change their security parameters, review their orders and manage their methods of payment.
- They can be split into (possibly many) simple user stories.
  - As a new customer, I can create a login with a password meeting specific security criteria.
  - As a existing customer, I can add a new method of payment.

Deliverables for sprint 0

- Initial version of product backlog (explained before).
- Product vision (explained next).
- High level architecture.

  In Agile, architecture is not frozen but will evolve during your sprints.
  To make sure you understand where you start, you are asked initially to explain :

  - A High level view of the main functional blocks of the product and their interactions
  - The identification of the challenging parts and algorithms to be designed
  - The initial technical architecture : choice of tools, programming languages, etc.
- Project budget (will be explained later).

The Product Vision

- To ensure that the team understands the business purpose of the product to develop
- Must be succinct (pass the elevator test : the ability to explain the project to someone within two minutes).

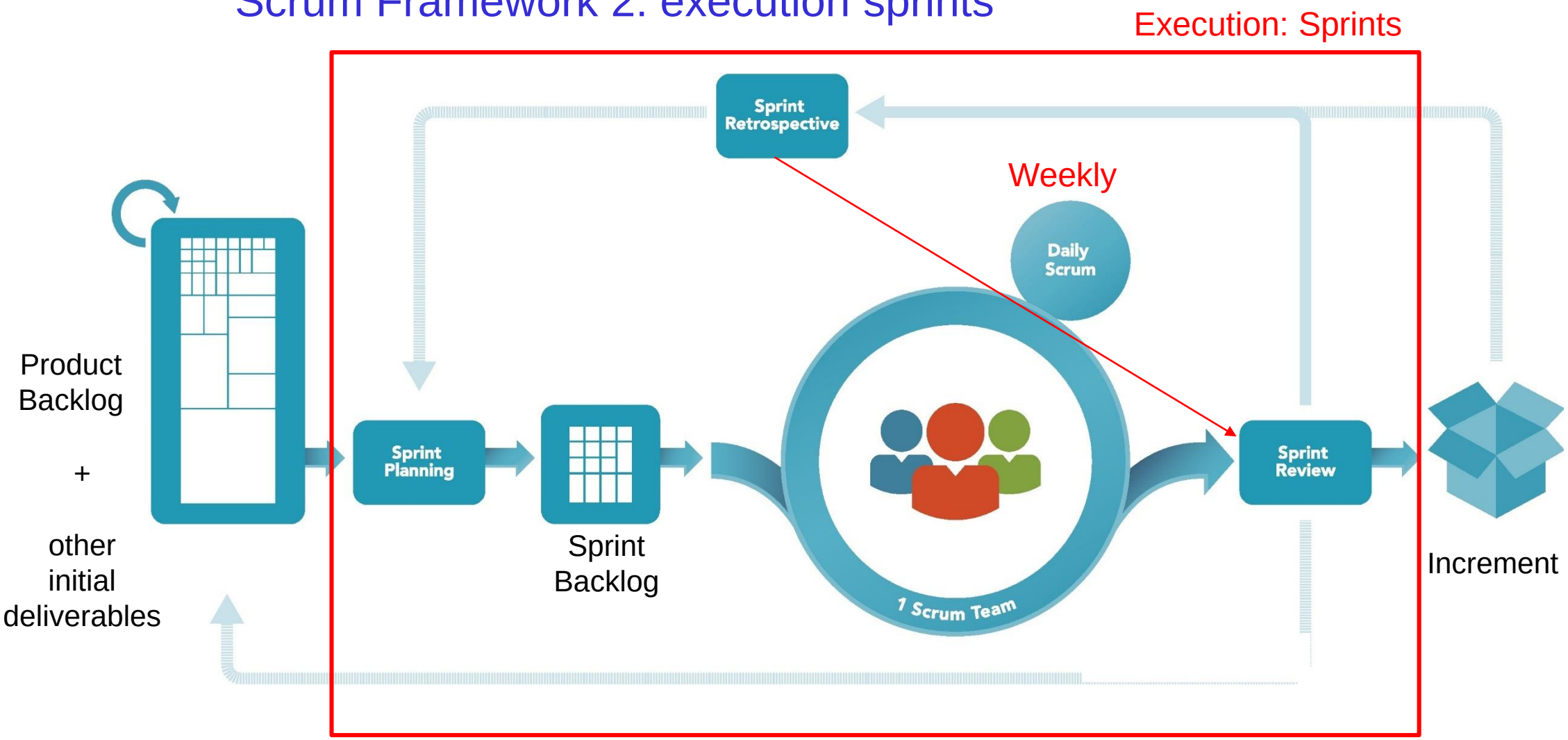
Template (Geoffrey Moore : Crossing the Chasm):

- For (target customer)
- Who (statement of the need or opportunity)
- The (product name) is a (product category)
- That (key benefit, compelling reason to buy)
- Unlike (primary competitive alternative)
- Our product (statement of primary differentiation)

Example

- For a mid-sized company's marketing and sales departments
- Who needs basic CRM functionality
- The CRM-Innovator is a Web-based service
- That provides sales tracking, lead generation, and sales representative support features that improve customer relationships at critical touch points.
- Unlike other services or package software products
- Our product provides excellent services at a moderate cost.

Scrum Framework 2: execution sprints



The Sprint

- A sprint is a time boxed effort of fixed duration, typically one month or less.
- During which a done, usable and potentially releasable product increment is created.
  - We will precise later the meaning of “done” for the integrated project.
- Performed by a small project team working intensively (it is a sprint !).
- Interacting and aligning their efforts through a set of meetings.
- A sprint has a sprint goal, defined during the sprint planning.
  - No changes can be made which will endanger the sprint goal.
  - However relevant scope may be clarified and re-negotiated.
  - The scope of the sprint, relevant for the sprint goal, is committed in a sprint backlog.

Key Sprint meetings

Scrum team meetings should be daily, but in this course we only ask for a weekly meeting

- Scrum planning meeting: to define the sprint goal, and select and estimate relevant product backlog items, which are put in the sprint backlog.
- ~~Daily~~ weekly scrum: team synchronization meeting to discuss status and adapt plans.
- Sprint retrospective: opportunity for the team to inspect its own progress, what went well and not well and decide on corrective actions.
- Sprint review: to inspect the product increment realized and adapt the product backlog if needed.
  - For the Integrated Project this will be a formal review with the academic team (content described later).
  - The preparation of the review will be used to manage product backlog and perform sprint retrospective.
- Frequent meetings with product owner (product planning & follow-up).

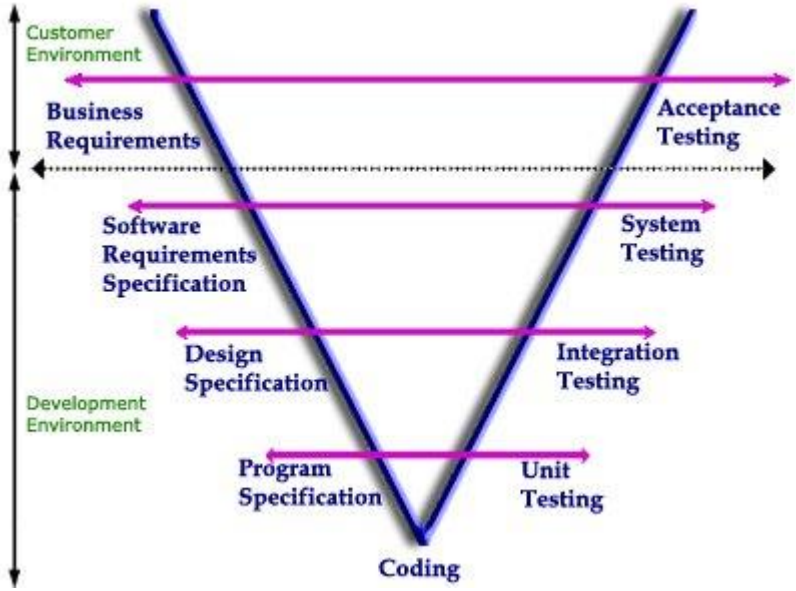
In this course, the sprint retrospective will be integrated in the sprint review report

Minimum 1 per sprint

What does « done » means ?

- A sprint should produce a done, potentially releasable product increment.
- Hence, the functionality produced must be tested !

The classical "V" of testing



The results of each sprint should be tested with :

- Unit tests;
- Integration tests.
- Performance and scalability tests.

In this course we only ask for full testing in sprint 3 and 4, and basic unit testing in sprint 2.

Project planning

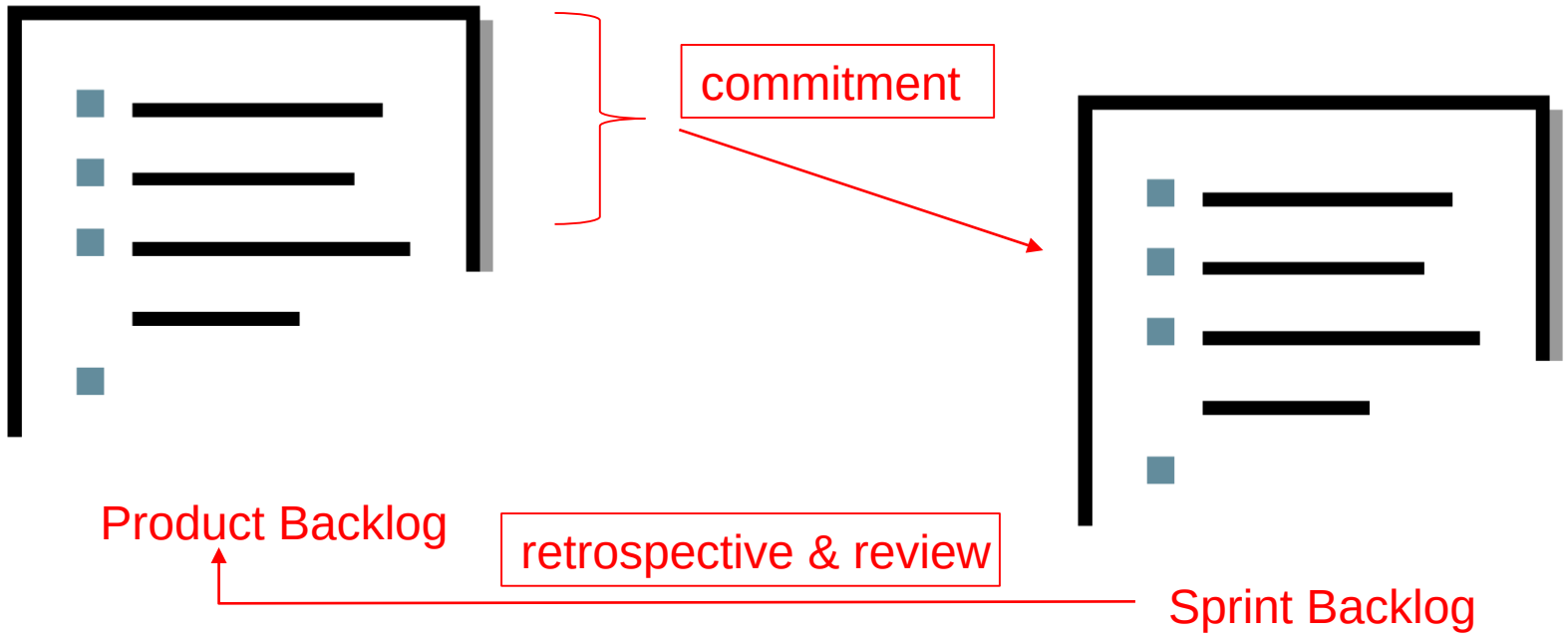
A fully detailed project plan is more typical of the classical approach.
It can be defined as :

- PMI: a formal, approved document used to guide both *project execution* and *project control…* documenting approved scope, cost, and schedule *baselines.*
  - The project management plan is the document that the project manager builds to describe in more details the planning of the project and its organization.
- PRINCE2: a statement of how and when a project's objectives are to be achieved, by showing the major products, milestones, activities and resources required on the project.

In Agile, sprint planning is done somewhat differently.

Sprint planning in Agile

- In Agile, planning is basically made from lists :
  - The team will select from the Product backlog relevant items, which are put in the Sprint Backlog.
  - Once filled the Sprint backlog represents the commitment of the Team for that sprint.
  - At the end of the sprint, the Product backlog is adjusted based on the results of the sprint.



Sprint backlog example

More planning information available

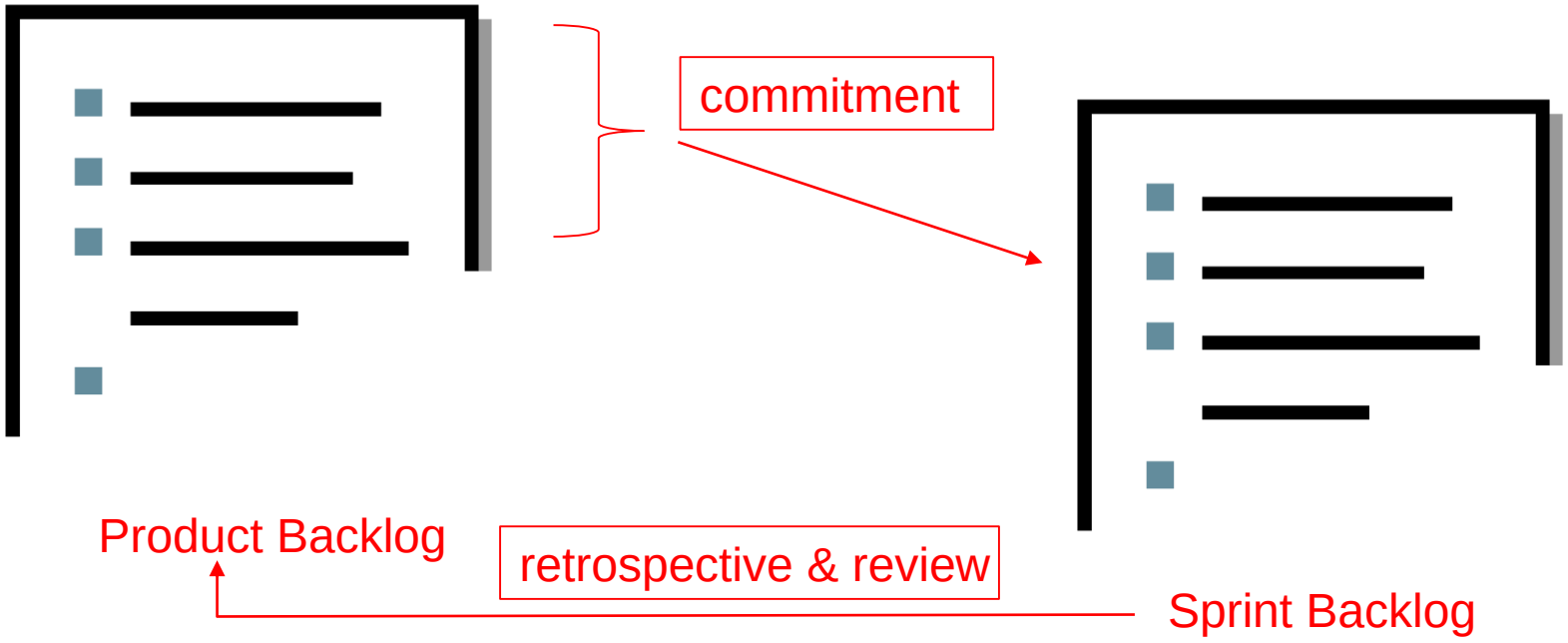
Type	Key	Summary	Assignee	Priority	Status	Updated	Due	Story Points
Story	TRIAL-10	As a developer, I can update story and task status with drag and drop	jean-louis binot	Medium	In Progress	5/10/2016	31/10/2016	5
Bug	TRIAL-13	As a developer, I can update details on an item using the details view	jean-louis binot	Medium	To Do	5/10/2016	31/10/2016	
Story	TRIAL-14	As a user, I can find important items on the board by using the customisable "Quick Filters" above	jean-louis binot	Medium	To Do	14/10/2016	31/10/2016	3
Story	TRIAL-15	As a scrum master, I can see the progress of a sprint via the Burndown Chart	jean-louis binot	Medium	Done	5/10/2016	31/10/2016	4
Story	TRIAL-16	As a team, we can finish the sprint by clicking the cog icon next to the sprint name above the "To Do" column then selecting "Complete Sprint"	jean-louis binot	Medium	Done	9/10/2016	31/10/2016	2
Bug	TRIAL-17	Instructions for deleting this sample board and project are in the description for this issue	jean-louis binot	Medium	Done	5/10/2016	31/10/2016	

Bugs can also be in backlog

Sample data extracted from Atlassian Jira

Key points in managing the Sprint backlog

1. Planning : select from the Product backlog prioritized stories relating to your goal.
2. Filling the Sprint Backlog to capacity (maximum of what you can do in one sprint).
3. Managing the sprint commitment and backlog changes.



1. How to plan the sprint backlog ?

- Identify carefully the sprint goal (high level view of the objective of the sprint).

  Ex : Implement backbone architecture, *implement a high level business functionality*, *establish and confirm feasibility of machine learning approach, implement user interface (basic screens, menus, login …).*

  - A sprint should ideally deliver a “done” component, but it is not always possible at start of project. There may be other deliverables than code (design, state of the art research, proof of feasibility…) .
  - Think about your critical path (things that will make you delay or fail if not solved, e.g. challenging design decisions, algorithms to design ….). The user interface is not always the highest priority.
  - Take priorities into account.

- Identify all stories relevant to the sprint goal.
- Check all dependencies to make sure needed / prerequisite stories are included.
  - Is it generally not a good idea to mix stories from several incomplete goals.
  - Only include other stories if you have enough capacity; ideally they will implement a second (sub)goal.
- Sprint planning may take more than one meeting; take the time to do it right.

2. How to fill the sprint backlog to capacity ?

- Two aspects must be considered :
  - Estimating the effort for each story,
  - Defining the capacity of the team for one sprint.
- How to estimate the efforts needed for completing individual stories ?
  - Efforts for completing stories are estimated in story points, not in person-hours.
  - Story points intuitively compare the relative size of stories compared to each other (cf. next slide).
- Why this change ? Because :
  - Estimating precisely in hours usually requires more information than available at start of sprint => not agile.
  - Precise estimating methods (e.g. function points) are complex.
  - Precisely tracking small tasks in hours requires heavy administrative project tracking => not agile.
  - Developers are better in assessing « how big this is » than in putting a number of hours.

2. How to fill the sprint backlog to capacity ? ./.

- How can we assess how large the stories are relatively to each other ?
  - Select a reference small example of story as being of size 1 and compare the others to it.
  - Use a scale to attribute points to small, medium, large, very large stories.
  - In the Integrated Project, we will use a popular scale: Fibonacci numbers: 1, 2, 3, 5, 8 et 13.
- Agreeing on estimates is done through a “planning poker” process :
  - During the planning meeting each team member assigns separately a estimate to the story;
  - If there is agreement the estimate is confirmed; if not the different opinions are discussed briefly.
- Note : why do we still keep track of efforts spent in timesheets ?
  - To keep a view on the balance of efforts between team members.
  - To be able to feed actuals into your budget, as an exercise in (simple) budget management.

2. How to fill the sprint backlog to capacity ? ./.

- How to estimate Sprint capacity for a inexperienced team ?
  - Initially, take a rough estimate : 1 story point = 1 “ideal” day of work.
  - Compute “ideal” days by applying a reduction factor (for example 25%) for meetings, learning, testing.
  - Possible initial estimates for a 4 week Sprint, 4 people team :
    - 4 * 4 * 5 * 0.75 = 60 story points.
    - 4 * 4 * 5 * 0.63 = 50 story points.
  - For the subsequent sprints, you will track and take into account your team velocity (see velocity chart, which will give you some more experience-based data.
- Important note : this is just a way to set an initial reference for team capacity in story points when you have no history concerning the performance of your team.

  You should not estimate all your story points in person-days during the project.

3. How to manage sprint commitment and backlog changes ?

- Sprint commitment is fixed as soon as possible after sprint planning.
  - Report any changes identified during sprint planning in the product backlog, not in the sprint backlog.
  - When sprint content is clear, populate the sprint backlog with the appropriate stories; the content of the sprint backlog represents your sprint commitment.
  - Set the start of sprint in the tool after sprint planning, when the sprint backlog is fixed.
- Sprint commitment should normally not be changed.
  - Sprint commitment is used to clarify objectives, allow follow-up analysis and build a forecasting capability for planning of next sprints; avoid any action that would negatively impact that.
- How to handle special situations ?
  - It is acceptable to add new stories on request of the project owner and if the team agrees, although not advisable; only do so if there is spare capacity or if these stories must be done in current sprint.
  - Do not suppress stories from sprint backlog; if they couldn't be done, explain why at end of sprint.
  - Do not revise the story point estimate of your stories during the sprint.

Other tips

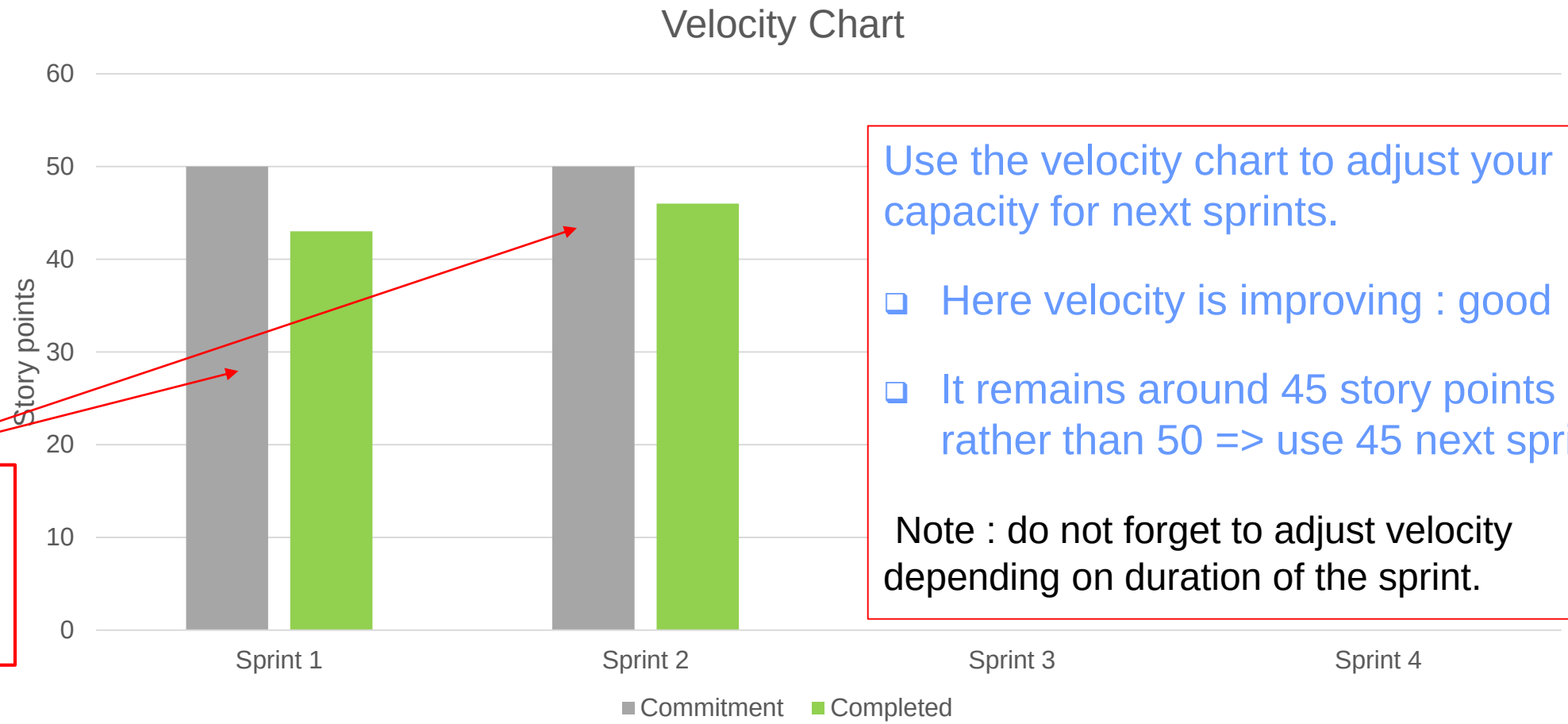
- You may include stories for learning tasks, or other investigation activities.
- Do not add meetings as stories in sprint commitment.
  - You can deal with them either as technical tasks, if the tools allows it, or by spreading the burden on all tasks (this is ok as we are not tracking hours for planning purpose – see later).

Deliverables for execution sprints (1 to 4)

- Project management deliverables :
  - Product Backlog status.
  - Sprint Backlog status.
  - Velocity chart.
  - Sprint burndown chart.
  - Defects status table.
  - Project budget.
  - Project status report (only for sprints 1 and 3).
- Common team deliverables :
  - Results of the sprint (code, other deliverables), with appropriate documentation.
  - Review presentation slides.
  - Individual efforts, in hours, recorded by each team member in the tool.

Sprint Velocity

Story points are only counted when the corresponding stories are DONE.



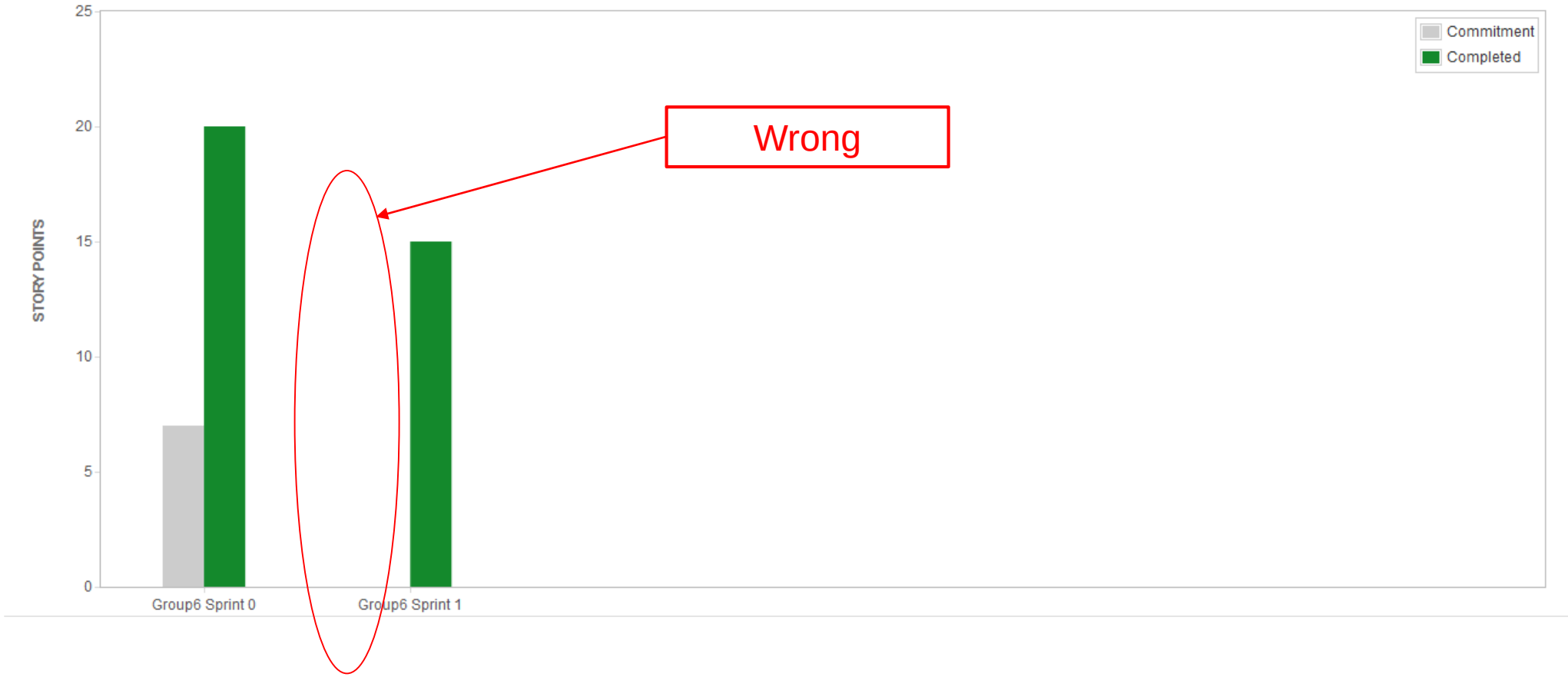
The first bar materializes your sprint commitment

Use the velocity chart to adjust your capacity for next sprints.

- Here velocity is improving : good !
- It remains around 45 story points rather than 50 => use 45 next sprint.

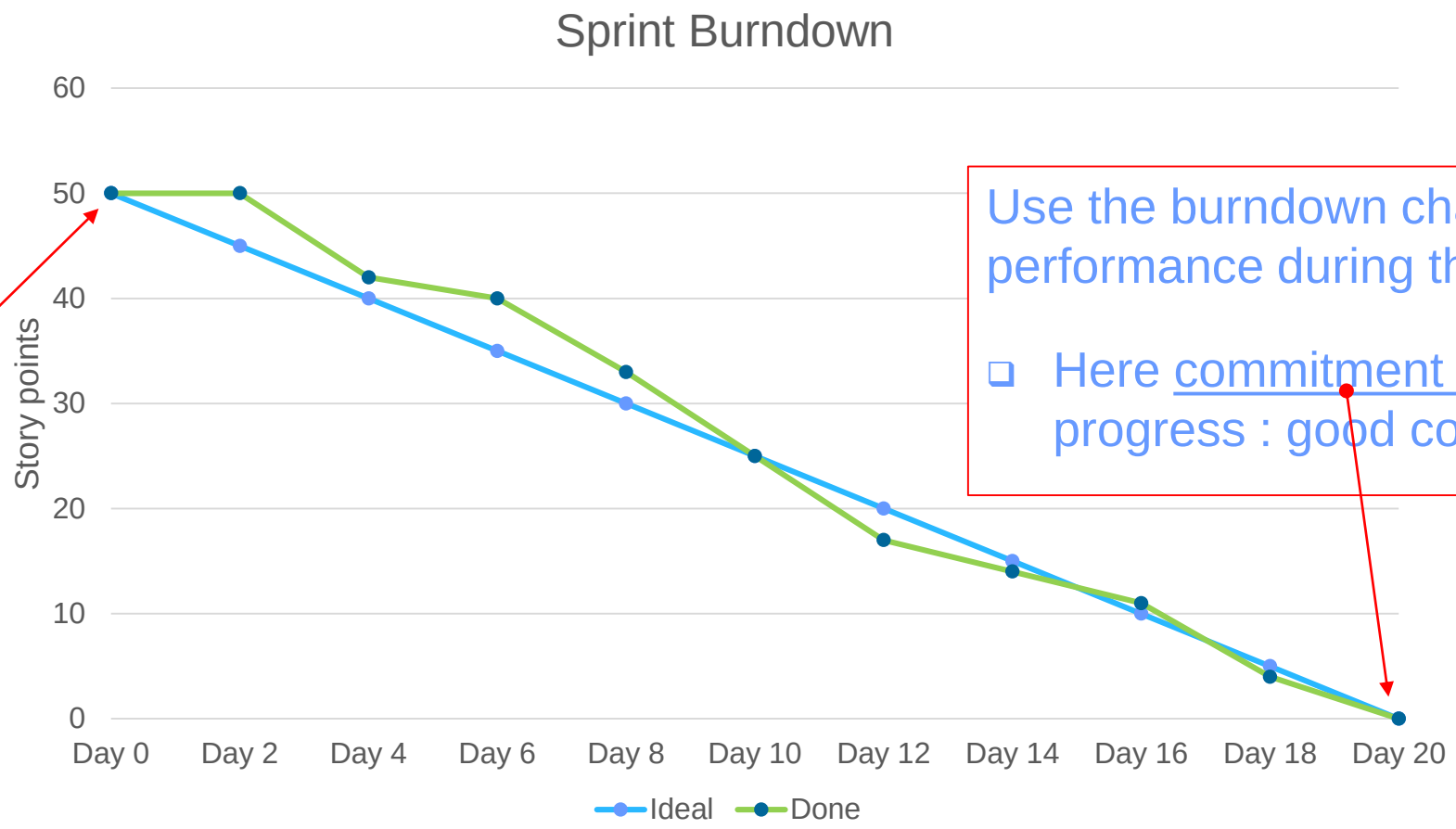
Note : do not forget to adjust velocity depending on duration of the sprint.

Velocity chart: commitment not set



Burndown chart

Story points are only counted when the corresponding stories are DONE.

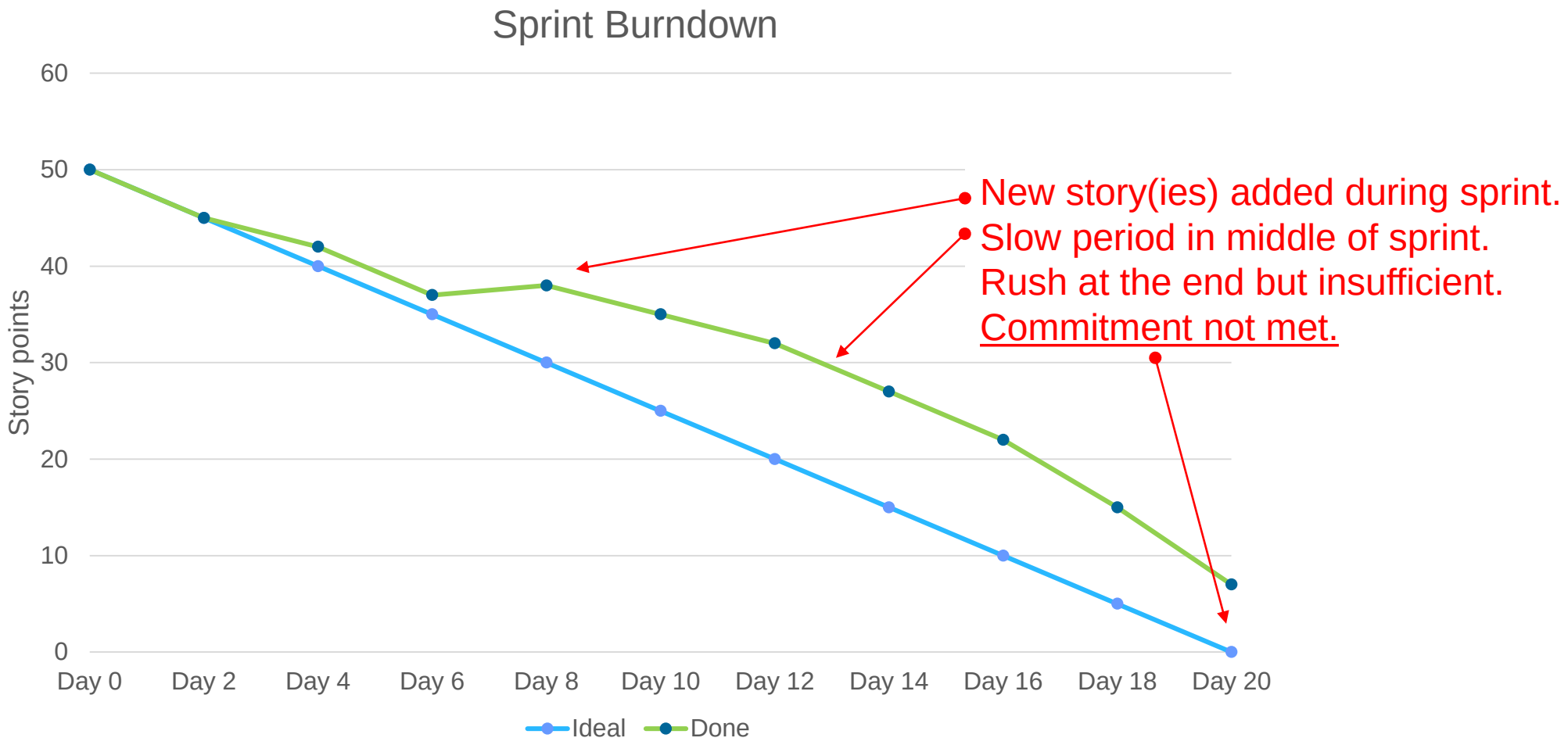


The starting dot materializes your commitment.

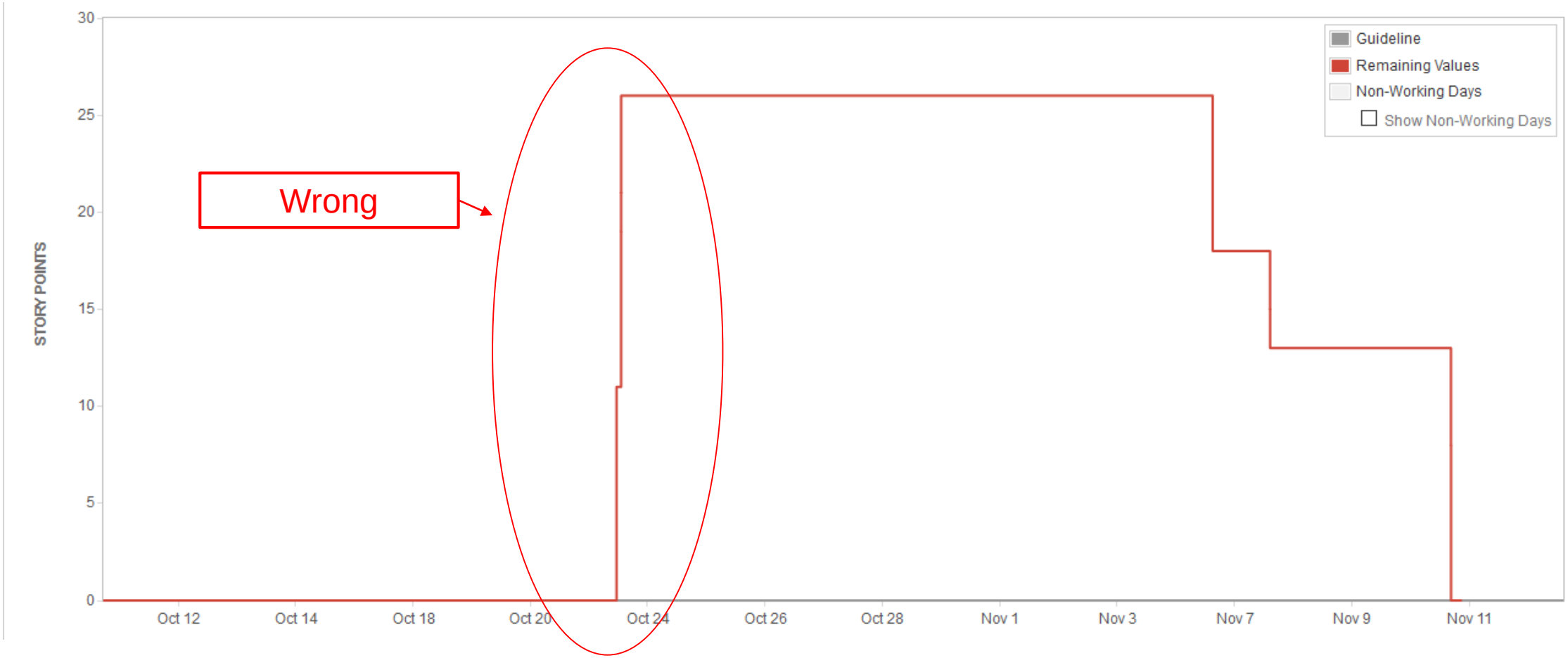
Use the burndown chart to analyse your performance during the sprint.

- Here commitment met and smooth progress : good control !

Burndown chart: Sprint commitment not met



Commitment not set at start of sprint, plus removing stories



Project Budget example

This budget is only indicative. You are responsible for doing do your own budget, and checking the numbers!

In many cases, one still needs to maintain an overall view of the Project Budget
In this project you are asked for a summary view, and to keep track of efforts through timesheets.

Indicative budget for a team of 4 people: 4 * 280h = 1120 h.
Over 20 weeks of effective work: 14h per week per person.

In this example, sprint 0 is done, sprint 1 is underway.

In the integrated Project you will normally only report at the end of a sprint.

	Budget (h) a	Actuals b	To complete c	At completion d = b + c	Budget variance e = d - a
Sprint 0 (2w)	112	104	0	104	-8
Sprint 1 (3w)	168	34	164	198	+30
Sprint 2 (3w)	168	0	168	168	0
Sprint 3 (5w)	280	0	280	280	0
Sprint 4 (5w)	280	0	280	280	0
Final defense preparation (2w)	112	0	112	112	0
Total	1120	138	1004	1142	+22

Understand what the numbers means to check for mistakes !

Management directors will assess your performance & credibility based on them.

Actuals are coming from your timesheets

This would mean you do not plan to work on sprint 1 and less than half on sprint 4 !!!

	Budget (h) a	Actuals b	To complete c	At completion d = b + c	Budget variance e = d - a
Sprint 0 (3w)	112	92	0	92	-20
Sprint 1 (3w)	168	0	0	0	-168
Sprint 2 (4w)	168	0	168	168	0
Sprint 3 (4w)	300	0	300	300	0
Sprint 4 (4w)	280	0	100	100	-180
Final defense preparation (2w)	112	0	112	112	0
Total	1140	92	680	800	-340

This is your baseline budget.
It is not supposed to change

This would mean 340h will not be used. Impossible !

Double total : total in lines and columns must be the same !

Managing budget changes

Your total budget is not supposed to change but events can create a need for adjustments.
Agile provides some flexibility on scope in exchange for a commitment on time and capacity.

If sprint capacity exceeds commitment, it can be used for additional features.
If sprint capacity has not been delivered, the project should aim to compensate in next sprints.

Underspending in sprint 1.

Plan to compensate in sprints 3 & 4.

Total budget has not changed.

	Budget (h) a	Actuals b	To complete c	At completion d = b + c	Budget variance e = d - a
Sprint 0 (3w)	112	92	0	92	-20
Sprint 1 (3w)	168	138	0	138	-30
Sprint 2 (4w)	168	0	168	168	0
Sprint 3 (4w)	280	0	315	315	35
Sprint 4 (4w)	280	0	315	315	35
Final defense preparation (2w)	112	0	112	112	0
Total	1120	230	910	1140	20

The defect status chart (table) : reporting on testing results

- **Critical/Showstopper :** system fails to do what it is actually supposed to do or does not work at all; there is no workaround solution (no other way to actually make the feature/system work).
- **Major/High :** major functionality or data of the system are affected. There may be a workaround but the workaround is not obvious or difficult to use.
- **Minor :** minor functionality or data defect but system is still operative. There is an easy workaround.
- **Cosmetic/Nice to have :** functionality or data not compromised. Defect is only an inconvenience.

Defect status	Start of sprint	Closed in sprint	Opened in sprint	End of sprint
Critical	2	1	1	2
Major	4	3	2	3
Minor	5	0	1	6
Nice to have	10	2	5	13

Acceptance is often based on solving all critical and a part of the major, depending on the available workarounds.

Sprint Review agenda

Expected agenda template for your Sprint review presentation (need to adjust for sprint 0)
The whole team takes part in the presentation. Slides must be sent beforehand.

- Sprint review :
  - The work the team committed to delivering and the work they completed, based on the sprint backlog status and product backlog status.
  - Technical content : progress and key decisions made during the iteration/sprint (requirements, architecture, algorithm design …), with justifications.
  - Project metrics (burndown charts, velocity chart, defects status chart, project budget, timesheet status).
  - Demo of the work done.
  - Priority vision of next iteration/sprint.
- Sprint retrospective
  - What went well.
  - What went not well and corrective actions.

Sprint status report : indicative table of content

Topics are essentially the same as the review slides, which capture the main points of the report. You may add topics or information but the points below should be present.

- Abstract.
- Review of Product Backlog and Sprint backlog status.
- Technical content.
  - Progress and key decisions made during the iteration/sprint (requirements, architecture, algorithm design …), with justifications.
- Project Management
  - Commented project metrics (burndown charts, velocity chart, defects status chart, project budget).
  - Timesheet status.
- Sprint retrospective.
- Priority vision of next iteration/sprint.
- Conclusion.

Final sprint deliverables

- Final application content (fully tested and documented).
- Final project report (template will be provided) including :
  - Summary of the project.
  - Characteristics of the solution :
    - Proposed architecture with justification;
    - Implemented functionalities;
    - Current limitations of the system;
    - Evaluation of the system.
  - Evaluation of the project (methodology and project management) :
    - Time management : based on velocity charts as they were at the end of sprint 4;
    - Cost management : based on final budget status (including time charged for final sprint);
    - Scope management : based on final status of Product Backlog;
    - Quality management : based among others on final defects status.
- Final presentation and demonstration of the system.

Summary: sprints, roles and deliverables

Responsibility color code	Initiation (sprint 0) Review 0	Execution (normal sprints) Reviews 1, 2, 3, 4	Closing (final sprint) Final review
Project manager (role rotating for deliverables for reviews 1 to 4)	Product vision Product backlog High level architecture Review presentation Timesheets	Project charts (all sprints): *Product backlog status* *Sprint backlog status* *Sprint burn down chart* *Velocity chart* *Defects status chart* Project status report (sprints 1 & 3) *Note : charts and information on backlog status should also be in your review slides*	Final application content (fully documented and tested) Final project report using project charts : *Final backlog status* *Final budget status* *Velocity chart at end of sprint 4* *Defects status chart*
All		Sprint application content Review presentation Timesheets	Final presentation & demo Timesheets

Timesheets need to be maintained during all sprints (including final sprint)

Conclusion

- Project management is a key discipline to ensure the success of projects (whatever the methodology chosen).
- It is also a suitable career path for a computer scientist, which can for example lead to management roles.
- Based on our experience and feedback from previous years, your soft skills will be challenged. But they are also needed in your professional life.
- One word of advise:
  - In agile, a Sprint means you have to SPRINT !

    To put it otherwise: a project is only won at the end but it can be lost at the beginning.

THANK YOU