Analysis, Design, and Software Architecture (BDSA)

Paolo Tell

Software process models

IT University
of Copenhagen

Outline

- Literature
  - [SE9] ch. 2+3
- Software Process Models
- Agile software development
  - XP
  - Scrum
- Also:
  - End-user development lecture
    - [http://www.itu.dk/Om-ITU/Events/Events/2015/Open-lecture-End-User-Software-Engineering---Beyond-the-Silos]
  - Course evaluation

- Software specification or Requirement Engineering
- Design and Implementation
- Validation
- Evolution

Software Processes

The software process

- A structured set of activities required to develop a software system.
- Activities common to all software processes:
  - Specification;
  - Design;
  - Validation;
  - Evolution.
- A software process model (or methodology) is an abstract representation of a process. It presents a description of a process from some particular perspective.

Plan-driven and agile processes

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In iterative or incremental processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In agile processes, development of "shippable" software take precedence over planning and documentation.
- In practice, most practical processes include elements of both plan-driven and iterative approaches.
- There are no right or wrong software processes.

Software process models

Examples

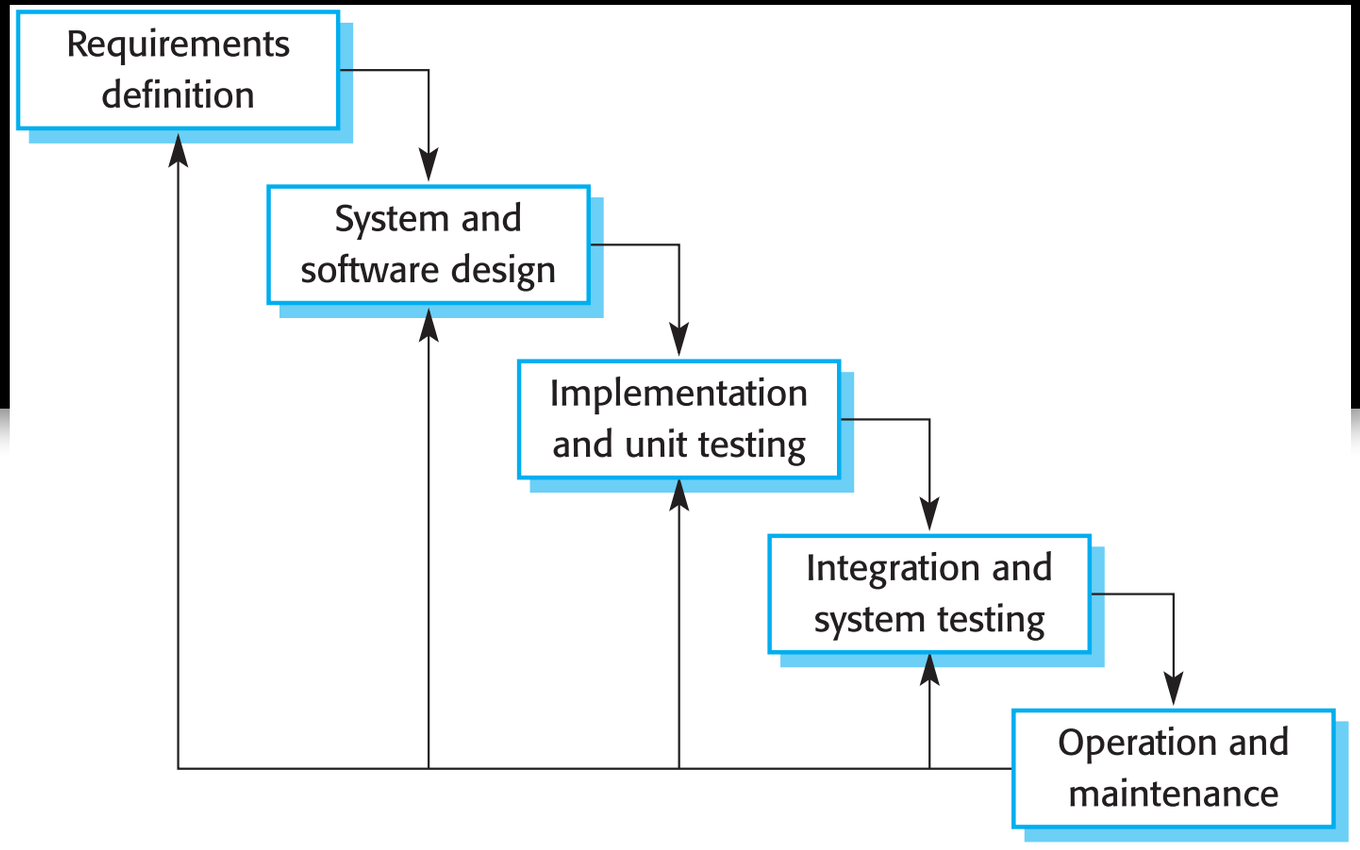
- The waterfall model
  - Plan-driven model.
  - Separate and distinct phases of specification and development.

- Incremental development
  - Specification, development and validation are interleaved.

- Reuse-oriented software engineering
  - The system is assembled from existing components. May be plan-driven or iterative/agile.

- SCRUM
  - The system is developed during a set of “sprints” where customer input is implemented.

Waterfall model phases



- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance

- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

Waterfall model problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.

- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Coping with change

Coping with change

- Change is inevitable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- Change leads to rework so the costs of change include both rework (e.g. re-analysing requirements) as well as the costs of implementing new functionality

Reducing the costs of rework

- Change avoidance
  - the software process includes activities that can anticipate possible changes before significant rework is required
  - E.g. a prototype system may be developed to show some key features of the system to customers

- Change tolerance
  - where the process is designed so that changes can be accommodated at relatively low cost.
  - E.g. incremental development where proposed changes may be implemented in increments

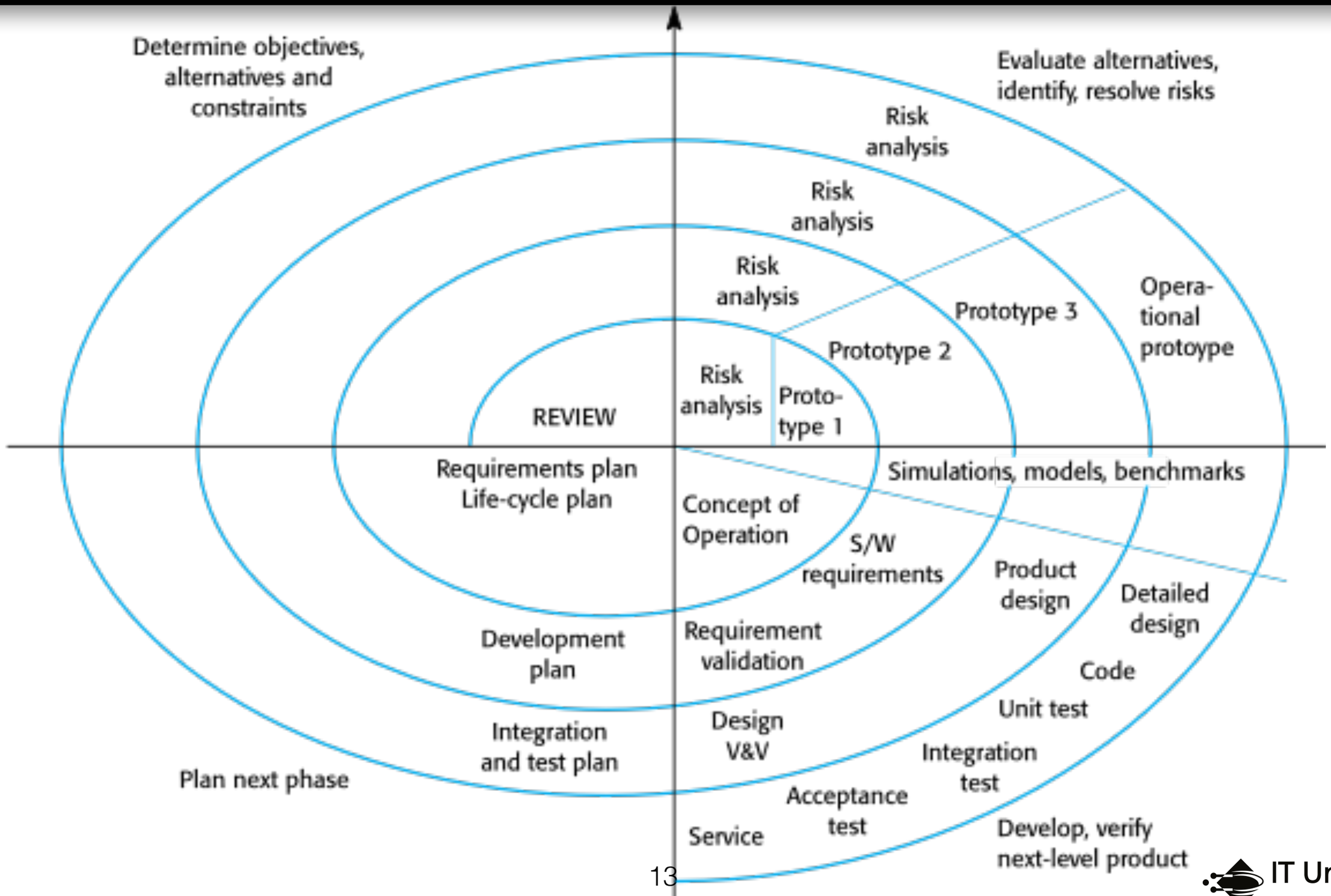
Process iteration

- System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems.

- Iteration can be applied to any of the generic process models.

- Two (related) approaches
  - Spiral development;
  - Incremental delivery.

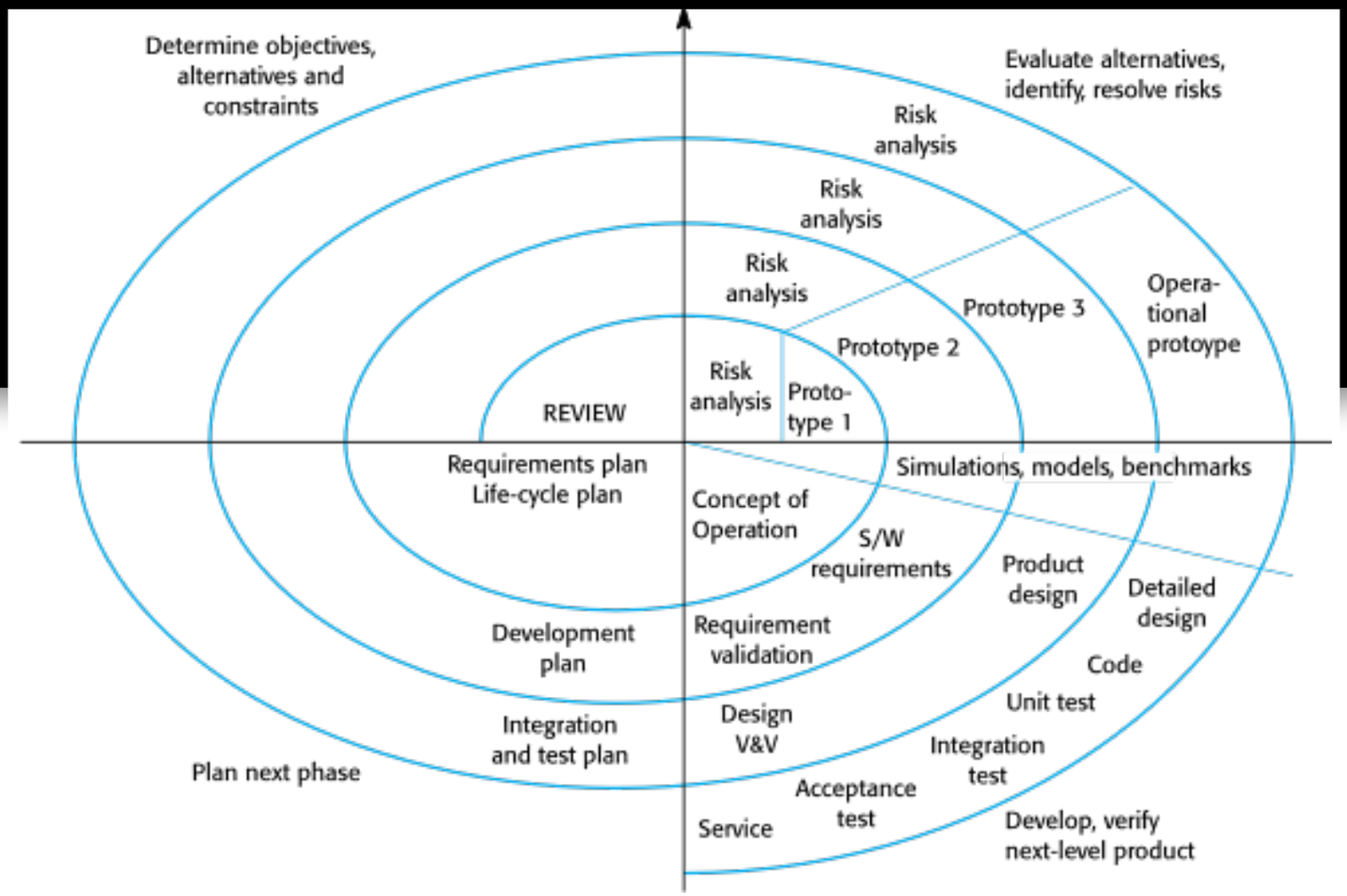
Spiral model of the software process



Iterative development

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

Spiral model sectors



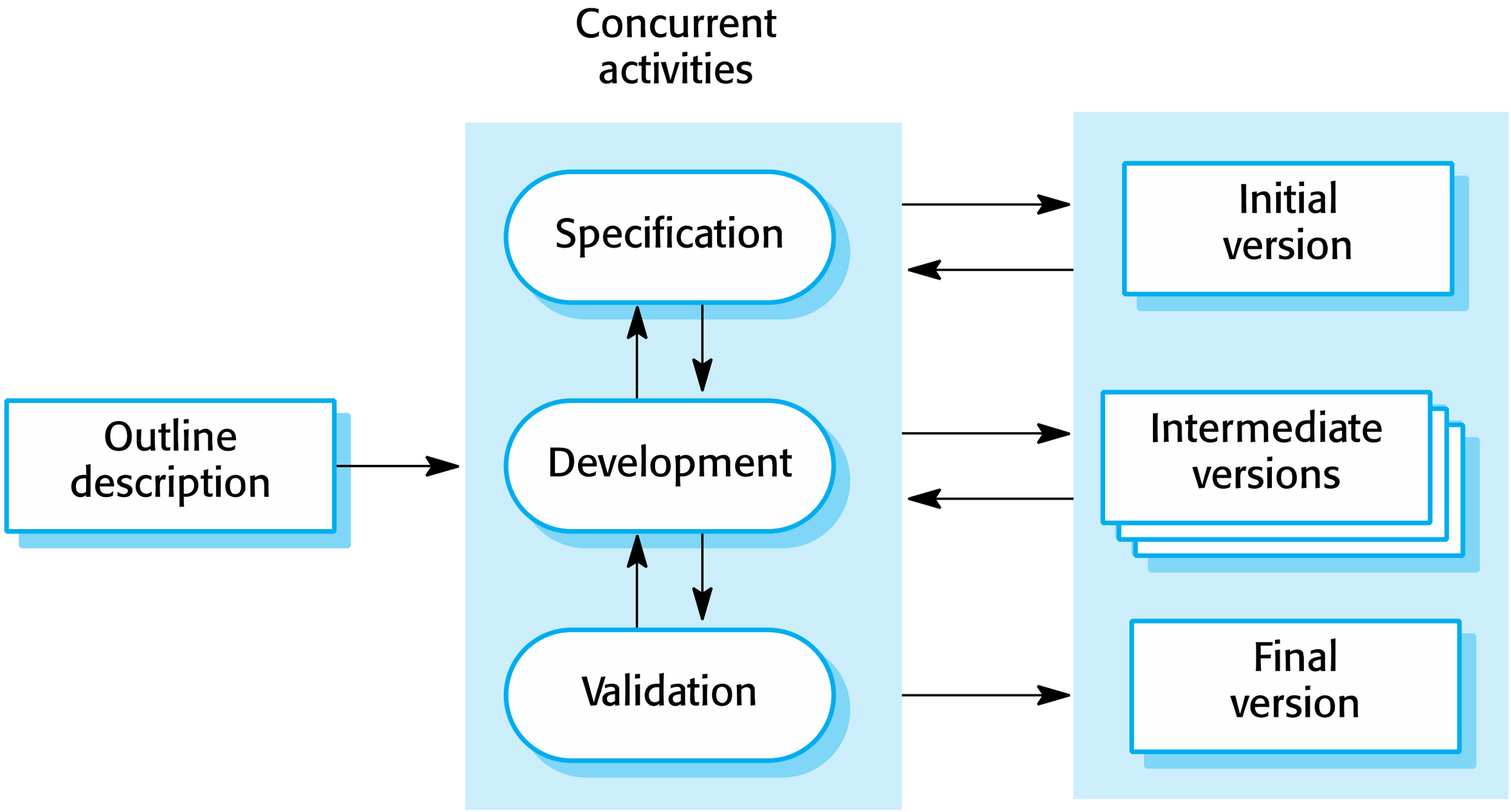
- Objective setting
  - Specific objectives for the phase are identified.

- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks.

- Development and validation
  - A development model for the system is chosen which can be any of the generic models.

- Planning
  - The project is reviewed and the next phase of the spiral is planned.

Incremental development



Incremental development benefits

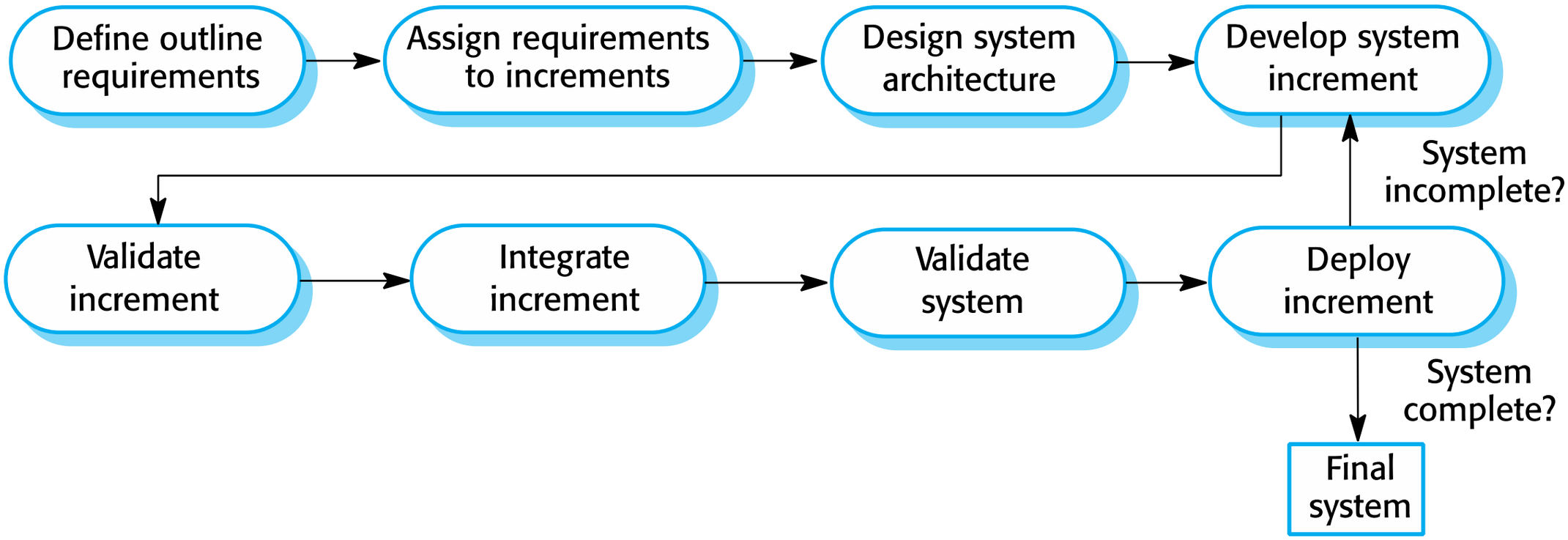
- The cost of accommodating changing customer requirements is reduced.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development problems

- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

- Applicability
  - For small or medium-size interactive systems;
  - For parts of large systems (e.g. the user interface);
  - For short-lifetime systems

Incremental delivery



- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritised and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

Incremental delivery advantages

- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

Incremental delivery problems

- Most systems require a set of basic facilities that are used by different parts of the system.
  - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.

- The essence of iterative processes is that the specification is developed in conjunction with the software.
  - However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.
  - Cost control and management becomes difficult

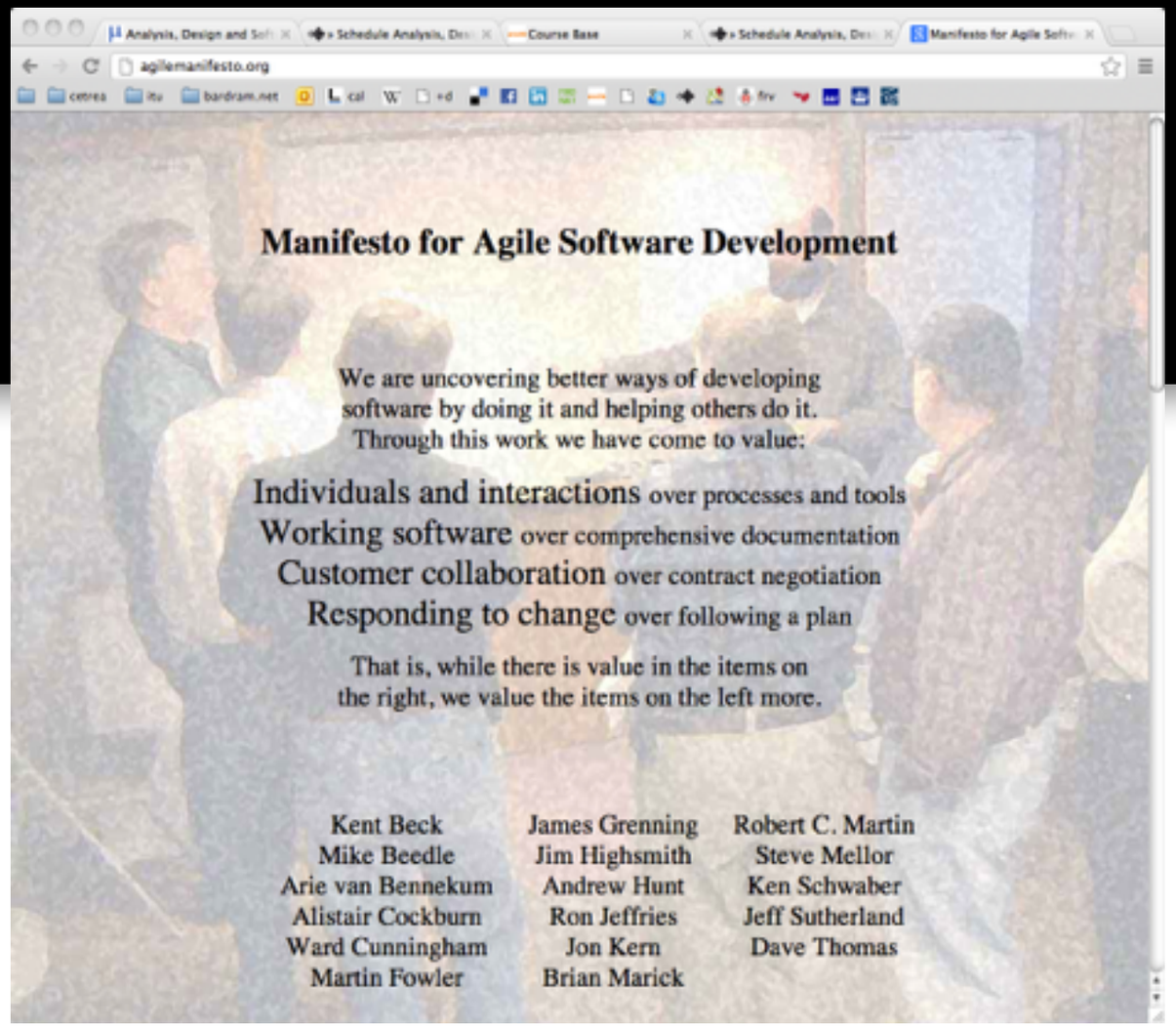
What is the difference between incremental development and delivery?

- Incremental development
  - Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
  - Evaluation done by user/customer proxy.

- Incremental delivery
  - Deploy an increment for use by end-users;
  - More realistic evaluation about practical use of software;
  - Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Agile processes

Agile methods



- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These methods:
  - focus on the code rather than the design
  - are based on an iterative approach to software development
  - are intended to deliver working software quickly and evolve this quickly to meet changing requirements.
- The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

The principles of agile methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

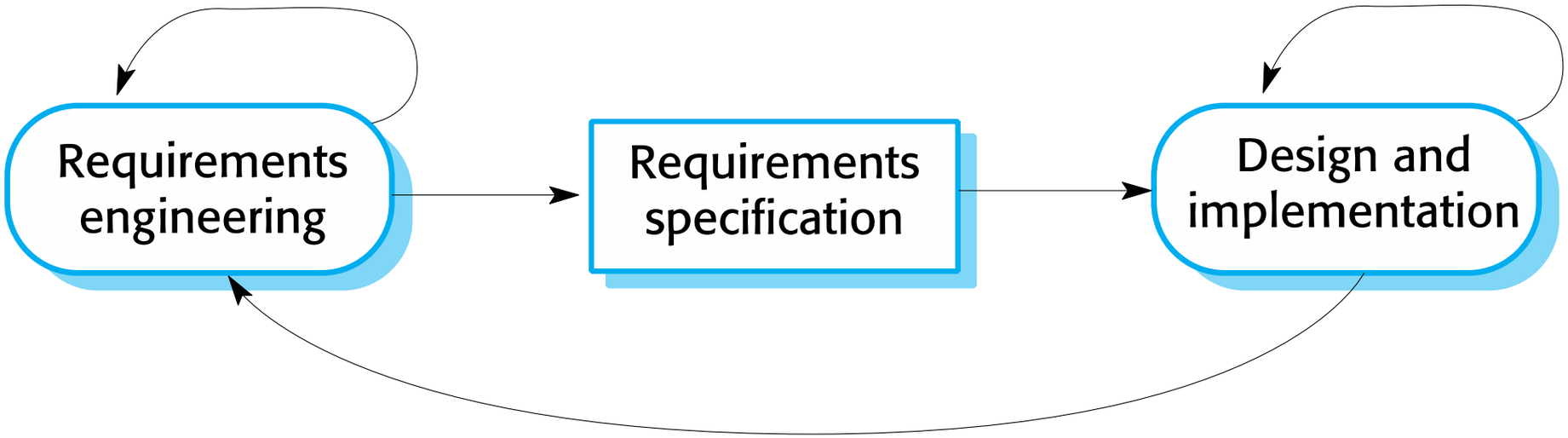
Plan-driven and agile development

- Plan-driven development
  - A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
  - Not necessarily waterfall model – plan-driven, incremental development is possible
  - Iteration occurs within activities.

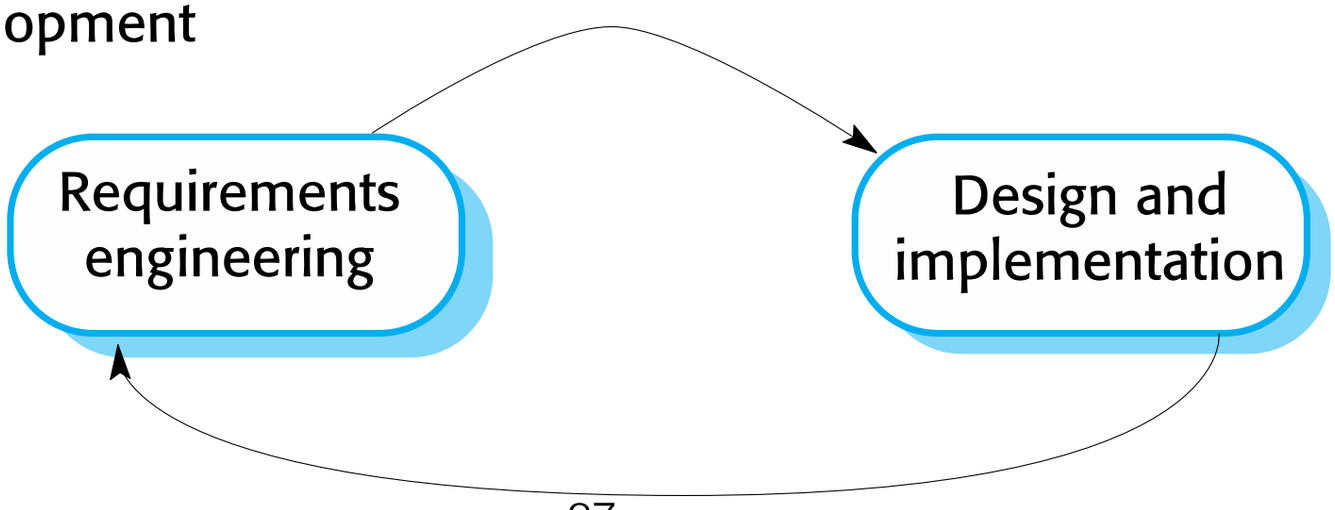
- Agile development
  - Specification, design, implementation and testing are inter-leaved
  - the outputs from the development process are decided through a process of negotiation during the software development process.

Plan-driven and agile specification

Plan-based development



Agile development



Agile method applicability

- Product development where a software company is developing a small or medium-sized product for sale.

- Custom system development within an organization in which there is a clear commitment from the customer to become involved in the development process and where there are not a lot of external rules and regulations that affect the software.

- Because of their focus on small, tightly-integrated teams, there are problems in scaling agile methods to large systems.

Problems with agile methods

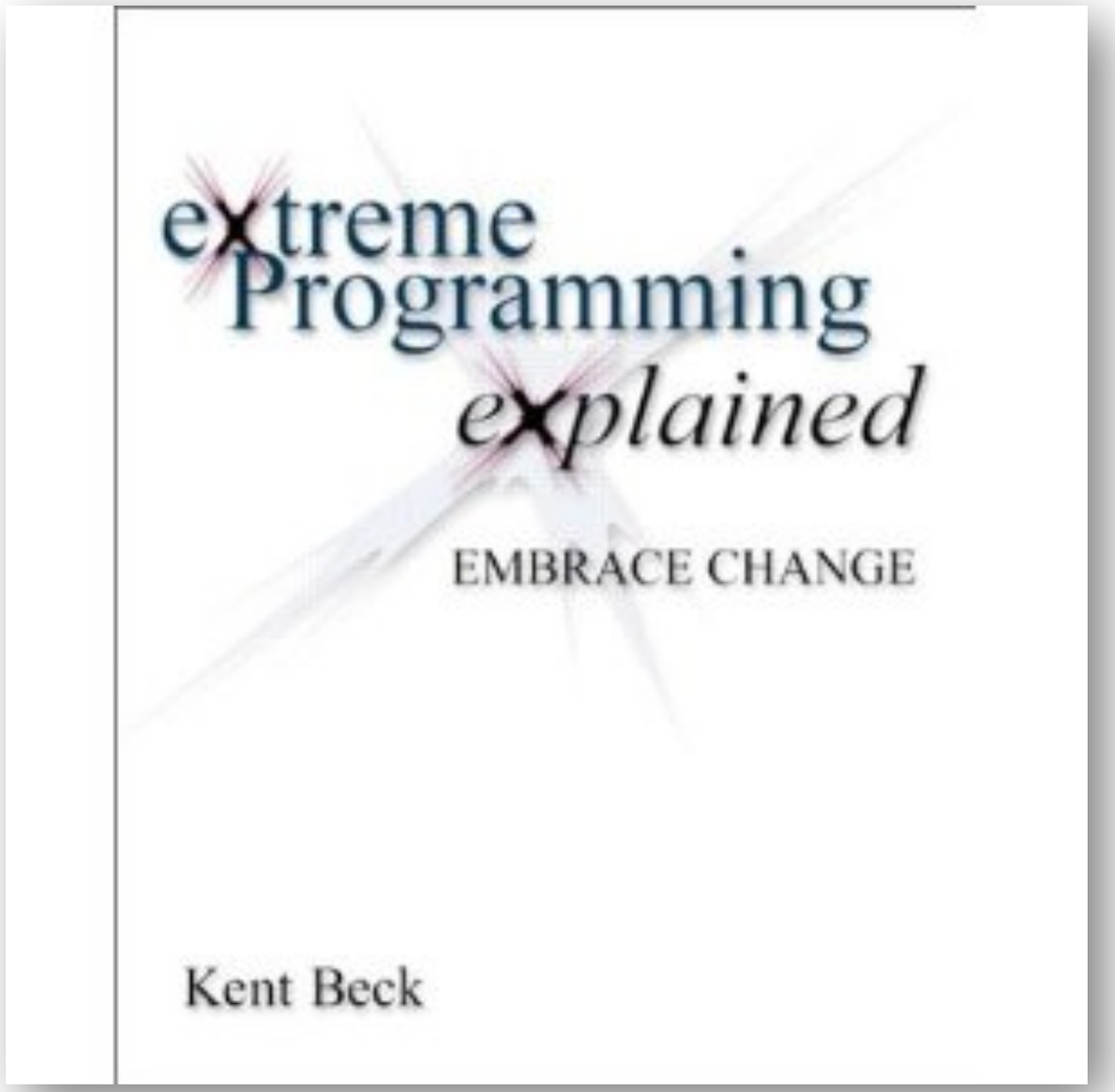
- It can be difficult to keep the interest of customers who are involved in the process.
- Team members may be unsuited to the intense involvement that characterises agile methods.
- Prioritising changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.
- Contracts may be a problem as with other approaches to iterative development.

Agile methods and software maintenance

- Most organizations spend more on maintaining existing software than they do on new software development. So, if agile methods are to be successful, they have to support maintenance as well as original development.

- Two key issues:
  - are systems that are developed using an agile approach maintainable, given the emphasis in the development process of minimizing formal documentation?
  - can agile methods be used effectively for evolving a system in response to customer change requests?

- Problems may arise if original development team cannot be maintained.



Extreme programming (XP)

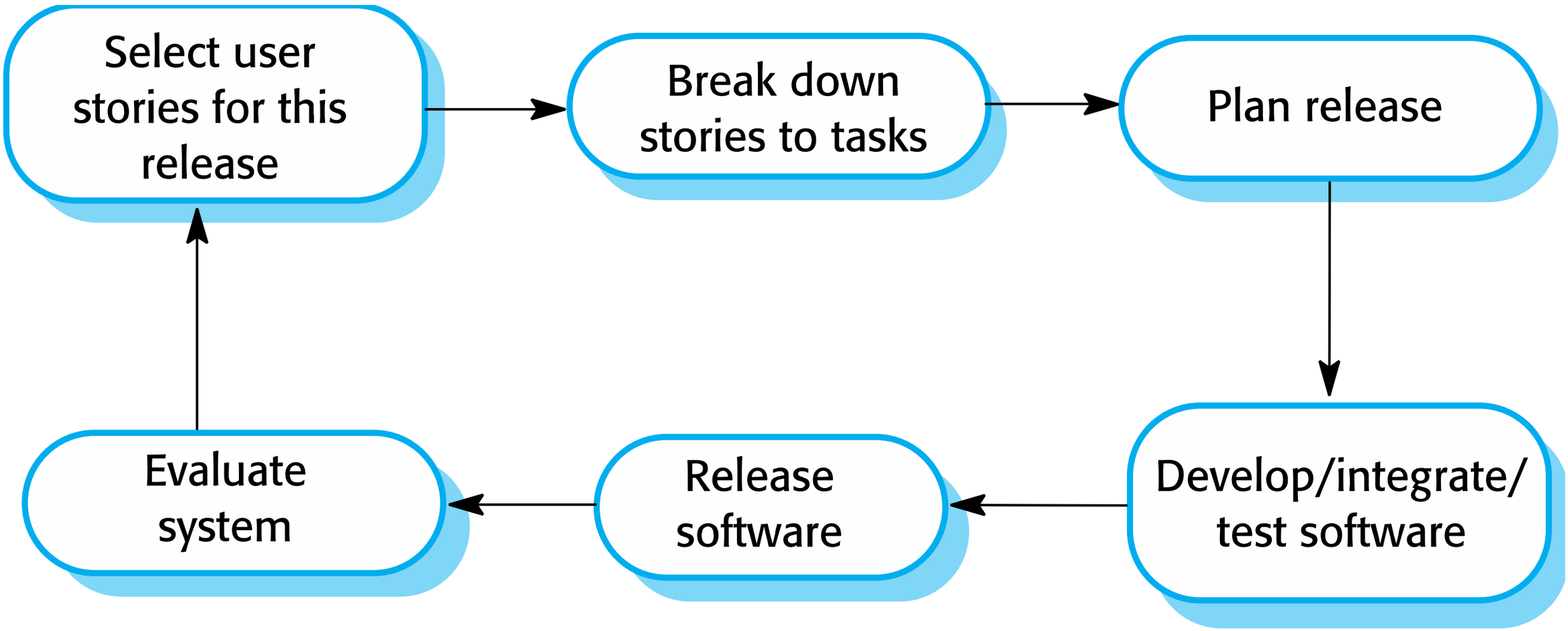
Extreme programming

- Perhaps the best-known and most widely used agile method.
- Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.

XP and agile principles

- Incremental development is supported through small, frequent system releases.
- Customer involvement means full-time customer engagement with the team.
- People not process through pair programming, collective ownership and a process that avoids long working hours.
- Change supported through regular system releases.
- Maintaining simplicity through constant refactoring of code.

The extreme programming release cycle



Extreme programming practices (a)

Principle or practice	Description
Incremental planning	Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

Extreme programming practices (b)

Principle or practice	Description
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity.
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

XP requirements and user stories

- In XP, a customer/user is part of the XP team and is responsible for making decisions on requirements.

- Requirements are elicited by writing User Stories with the client
  - User stories are high-level scenarios or use cases that encompass a set of coherent features

- Planning is driven by requirements and their relative priorities
  - Developers decompose each user story in terms of development tasks that are needed to realize the features required by the story
  - Developers estimate the duration of each task in terms of days
  - If a task is planned for more than a couple of weeks, it is further decomposed into smaller tasks

- The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

A ‘prescribing medication’ user story

Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either ‘current medication’, ‘new medication’ or ‘formulary’.

If you select ‘current medication’, you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, ‘new medication’, the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose ‘formulary’, you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click ‘OK’ or ‘Change’. If you click ‘OK’, your prescription will be recorded on the audit database. If you click ‘Change’, you reenter the ‘Prescribing medication’ process.

Examples of task cards for prescribing medication

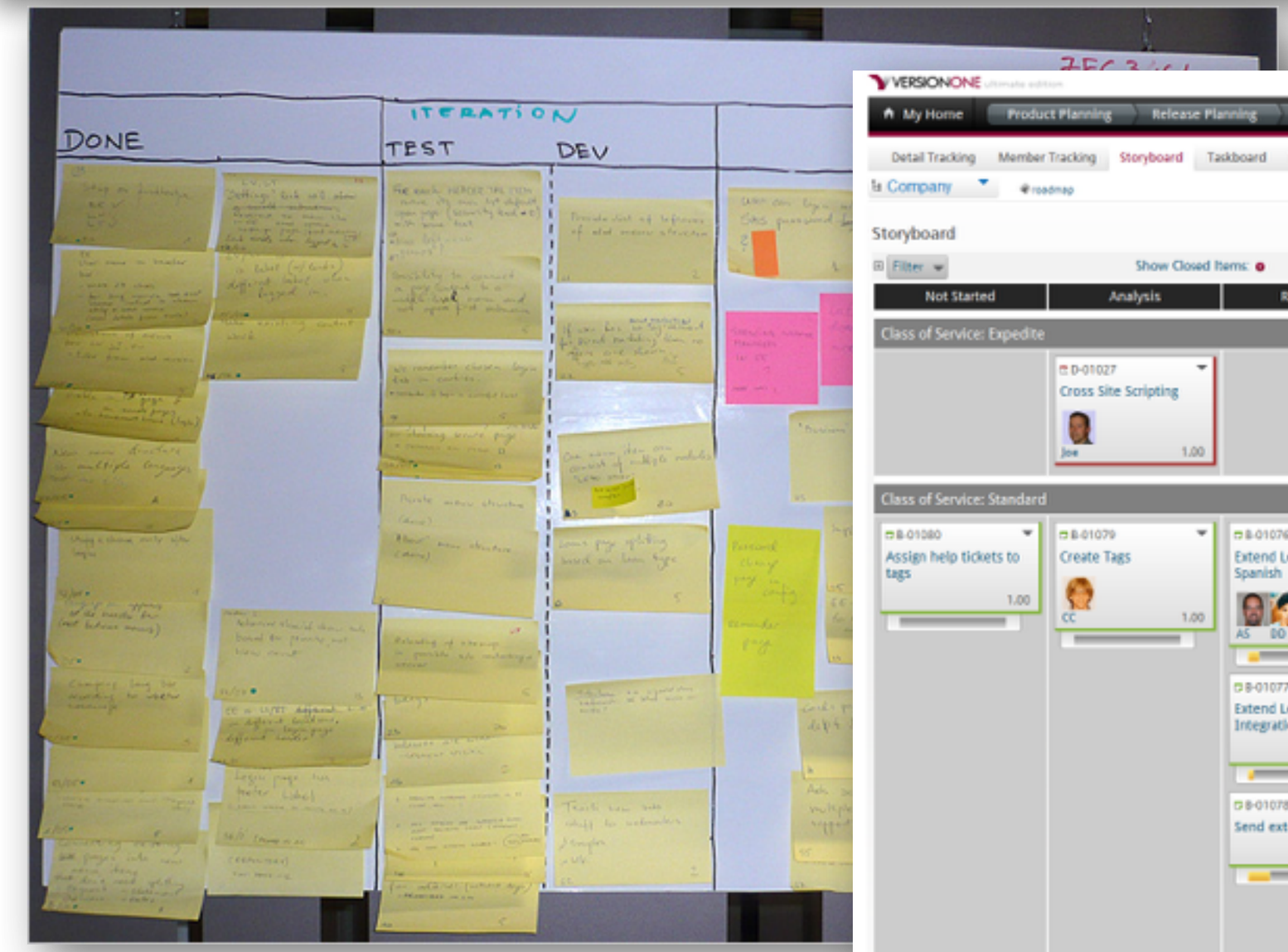
Task 1: Change dose of prescribed drug

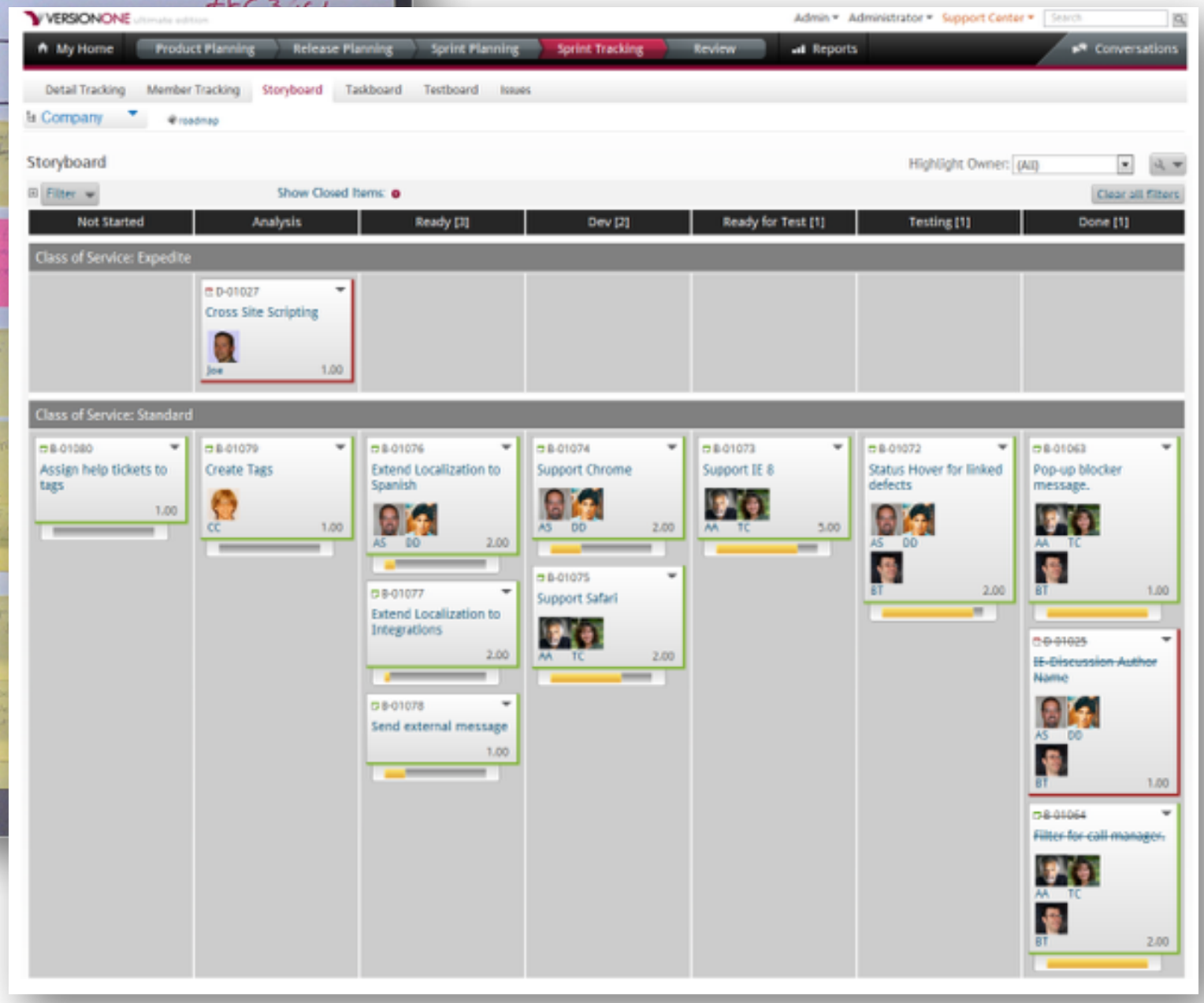
Task 2: Formulary selection

Task 3: Dose checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.
Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.
Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low.
If within the range, enable the 'Confirm' button.

Story Boards





Refactoring

- Programming team look for possible software improvements
  - … and make these improvements even where there is no immediate need for them.
  - This improves the understandability of the software and so reduces the need for documentation.
  - Changes are easier to make because the code is well-structured and clear.

- However, some changes requires architecture refactoring and this is much more expensive.

- Examples
  - Re-organization of a class hierarchy to remove duplicate code.
  - Tidying up and renaming attributes and methods to make them easier to understand.
  - The replacement of inline code with calls to methods that have been included in a program library.

Testing in XP

- Testing is central to XP, and XP has developed an approach where the program is tested after every change has been made.

- XP testing features:
  - Test-first development.
  - Incremental test development from scenarios.
  - User involvement in test development and validation.
  - Automated test harnesses are used to run all component tests each time that a new release is built.

Test-first development

- Writing tests before code clarifies the requirements to be implemented.
- Tests are written as programs rather than data so that they can be executed automatically. The test includes a check that it has executed correctly.
  - Usually relies on a testing framework such as Junit.

- Regression Testing
  - All previous and new tests are run automatically when new functionalities are added, thus checking that the new functionalities have not introduced errors.

Customer involvement

- The role of the customer in the testing process is to help develop acceptance tests for the stories that are to be implemented in the next release of the system.

- The customer who is part of the team writes tests as development proceeds. All new code is therefore validated to ensure that it is what the customer needs.

- However, people adopting the customer role have limited time available and so cannot work full-time with the development team.
  - They may feel that providing the requirements was enough of a contribution and so may be reluctant to get involved in the testing process.

Test case description for dose checking

Test 4: Dose checking

Input:

1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:

1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose * frequency is too high and too low.
4. Test for inputs where single dose * frequency is in the permitted range.

Output:

OK or error message indicating that the dose is outside the safe range.

XP testing difficulties

- Programmers prefer programming to testing and sometimes they take short cuts when writing tests.
  - For example, they may write incomplete tests that do not check for all possible exceptions that may occur.

- Some tests can be very difficult to write incrementally.
  - For example, in a complex user interface, it is often difficult to write unit tests for the code that implements the 'display logic' and workflow between screens.

- It is difficult to judge the completeness of a set of tests.
  - Although you may have a lot of system tests, your test set may not provide complete coverage.

Summary of the XP methodology

Planning	Collocate the project with the client, write user stories with the client, frequent small releases (1-2 months), create schedule with release planning, kick off an iteration with iteration planning, create programmer pairs, allow rotation of pairs.
Modeling	Select the simplest design that addresses the current story; use a system metaphor to model difficult concepts; use cards for the initial object identification; write code that adheres to standards; refactor whenever possible.
Process	Code unit test first, do not release before all unit tests pass, write a unit test for each uncovered bug, integrate one pair at the time.
Control	Code is owned collectively. Adjust schedule, rotate pairs, daily status stand-up meeting, run acceptance tests often and publish the results.

Scrum

Scrum

- Definition (Rugby): a Scrum is a way to restart the game after an interruption,
  - the forwards of each side come together in a tight formation and struggle to gain possession of the ball when it is tossed in among them.

- Definition (Software Development): scrum is an agile, lightweight process
  - to manage and control software and product development with rapidly changing requirements
  - based on improved communication and maximizing cooperation.

Why Scrum?

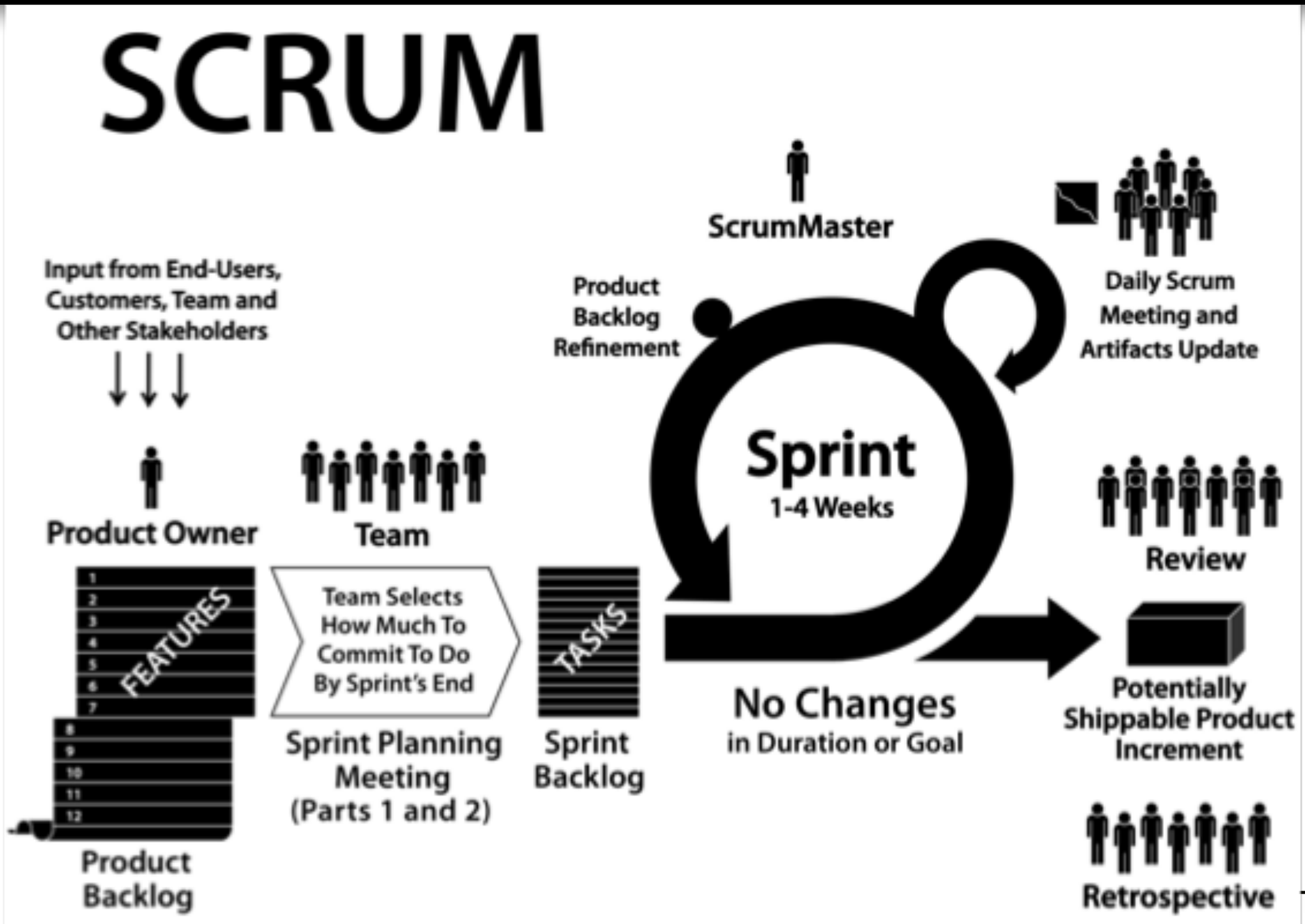
Traditional methods are like relay races

Agile methods are like rugby

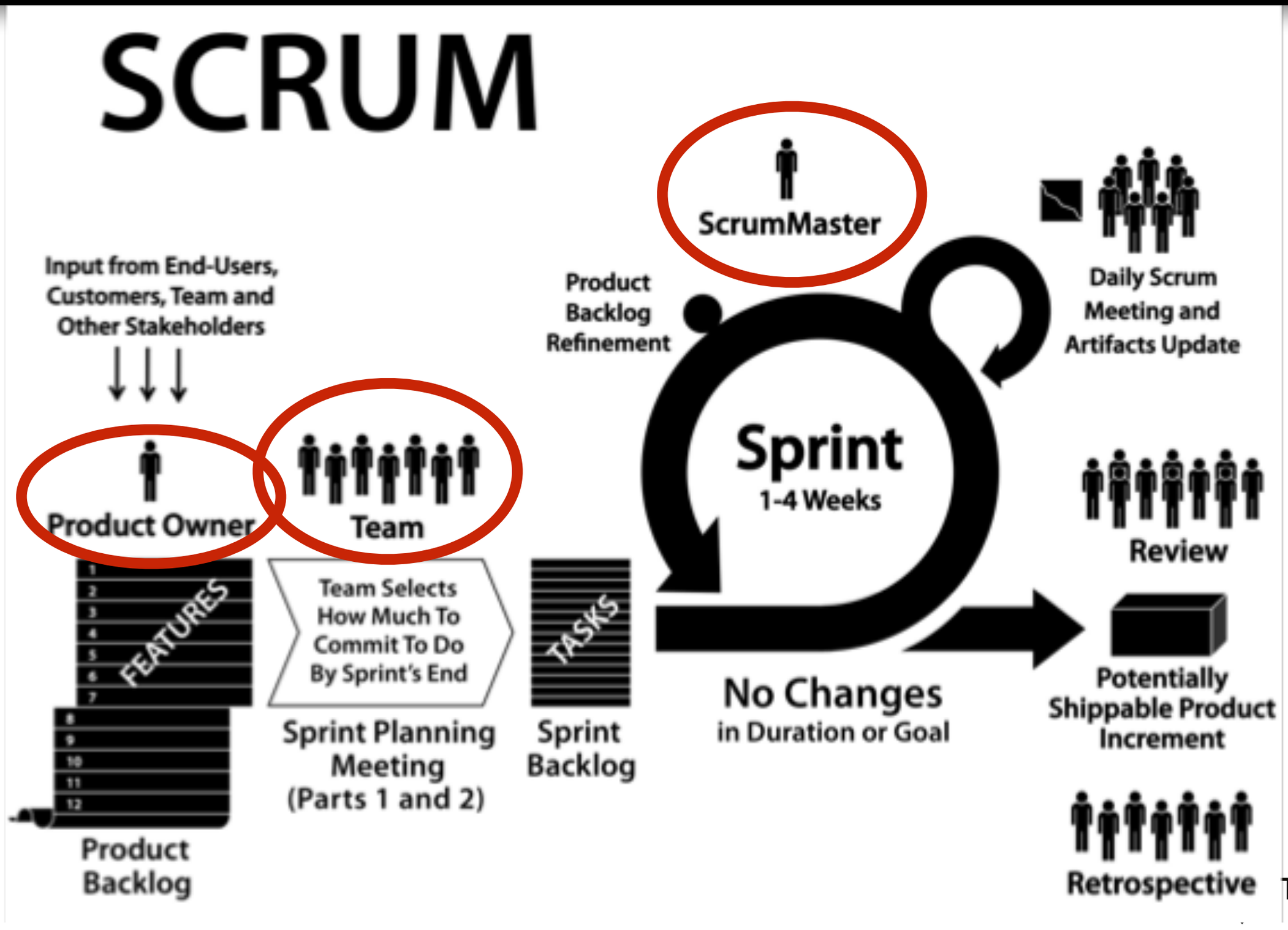
Scrum

- The Scrum approach is a general agile method but its focus is on managing iterative development rather than specific agile practices.
- There are three phases in Scrum:
  - the initial phase is an outline planning phase in which you establish the general objectives for the project and design the software architecture;
  - this is followed by a series of sprint cycles in which each cycle develops an increment of the system;
  - the project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

Scrum overview



Scrum roles


SCRUM
Input from End-Users, Customers, Team and Other Stakeholders
ScrumMaster
Product Backlog Refinement
Daily Scrum Meeting and Artifacts Update
Product Owner
Team
Sprint
1-4 Weeks
Review
FEATURES
Team Selects How Much To Commit To Do By Sprint's End
TASKS
Potentially Shippable Product Increment
No Changes
in Duration or Goal
Sprint Planning Meeting (Parts 1 and 2)
Sprint Backlog
Product Backlog
Retrospective

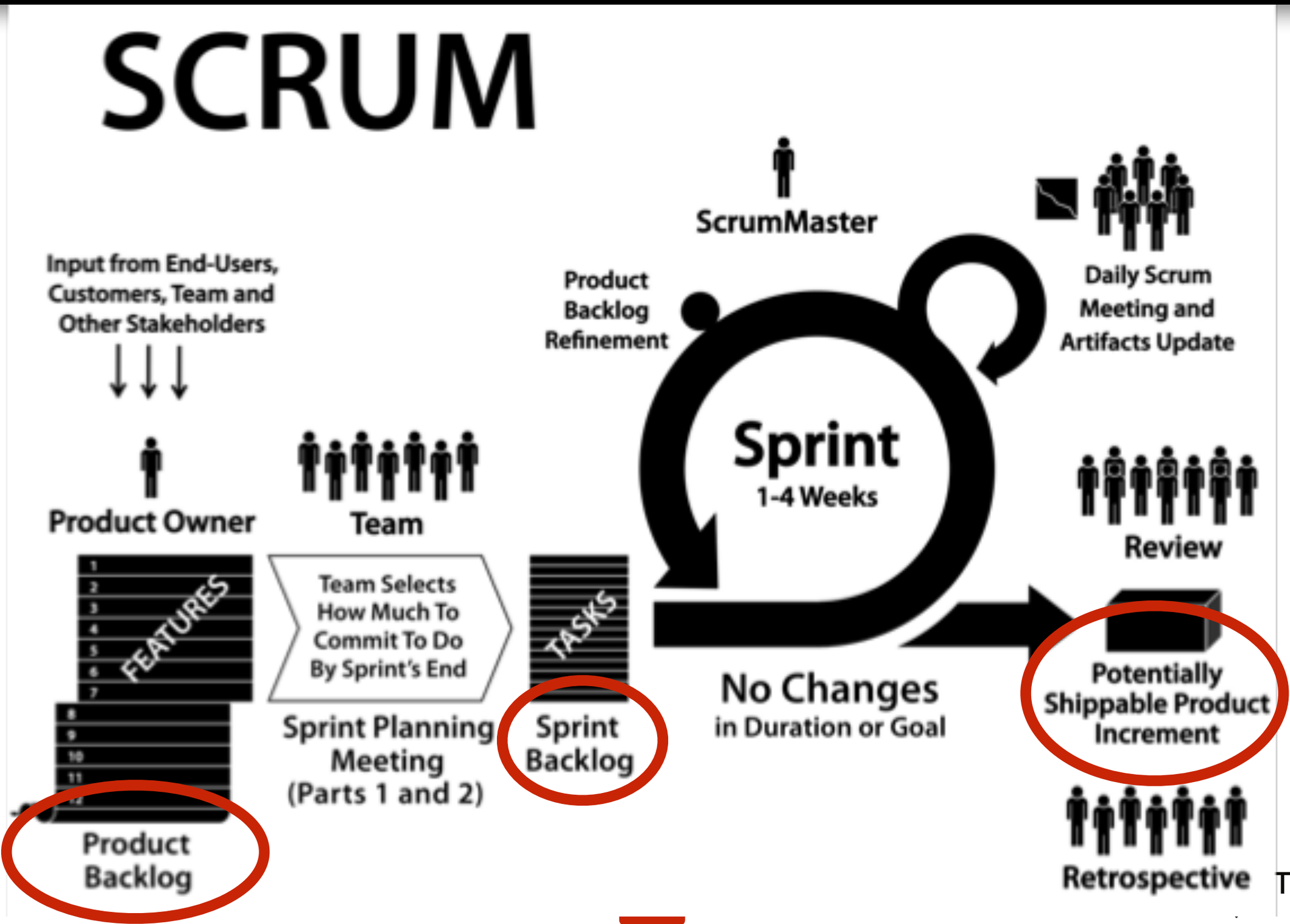
Scrum roles

- Product Owner
  - maximizing ROI, responsible for P&L
  - identifying, prioritizing, refining product features
  - product manager, customer, marketing manager, …
  - actively and frequently interact with the team

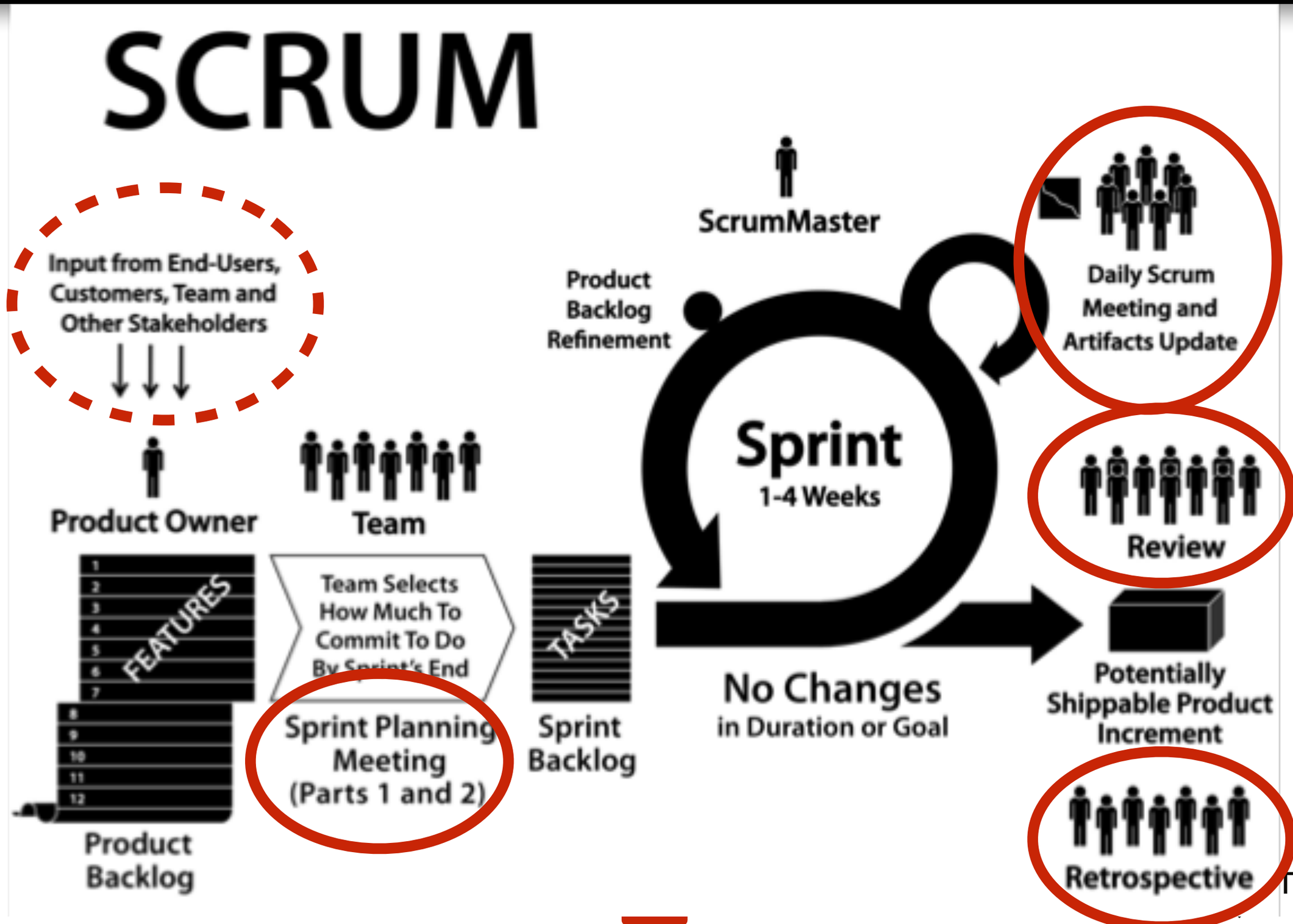
- The Team (the "pigs")
  - the team that builds the shippable software
  - cross-functional, self-organizing (self-managing)
  - 7 +/-2 persons, stable
  - dedicated to one product – avoid multi-tasking

- Scrum Master
  - helps the product owner + team to be successful
  - is not a manager in the traditional sense
  - facilitates communication, shields the team
  - in small team, a team member can be the scrum master, BUT the product owner and scrum master CANNOT be the same!

Scrum artefacts



Scrum activities



Sprint planning

- 2 meetings - ~4 hours (no more that 8!!)

- Sprint planning part I – "what" has to be done
  - all team review backlog
  - product owner states priorities
  - agreeing on "Definition of Done"

- Sprint planning part II – "how" it will be done
  - team "commit to tasks", in prioritized order
  - estimates "capacity" of team (i.e. total available time)
  - estimates tasks
  - adding them to the sprint backlog

- Important to note
  - once the sprint is planned and committed, nothing can change the sprint backlog

Capacity and sprint backlog

Sprint Length	2 weeks
Workdays during Sprint	8 days

Team Member	Available Days During Sprint*	Available Hours per Day	Total Available Hours
Tracy	8	4	32
Sanjay	7	5	35
Phillip	8	4	32
Jing			

* Net of v

				New Estimates of Effort Remaining as of Day...					
Product Backlog Item	Sprint Task	Volunteer	Initial Estimate of Effort	1	2	3	4	5	6
As a buyer, I want to place a book in a shopping cart	modify database		5						
	create webpage (UI)		8						
	create webpage (Javascript logic)		13						
	write automated acceptance tests		13						
	update buyer help webpage		3						
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5						
	complete machine order for pRank		8						
	change DCP and reader to use pRank http API		13						

Figure4. Sprint Backlog

Sprint Backlog Tasks on the Wall

Sprint and daily scrum

- Daily Scrum
  - 15 min., standing up, @0900 (severe penalty for being late)
  - what has been done, what is planned, any obstacles
  - no discussion
  - self-organizing, i.e. no “management” present

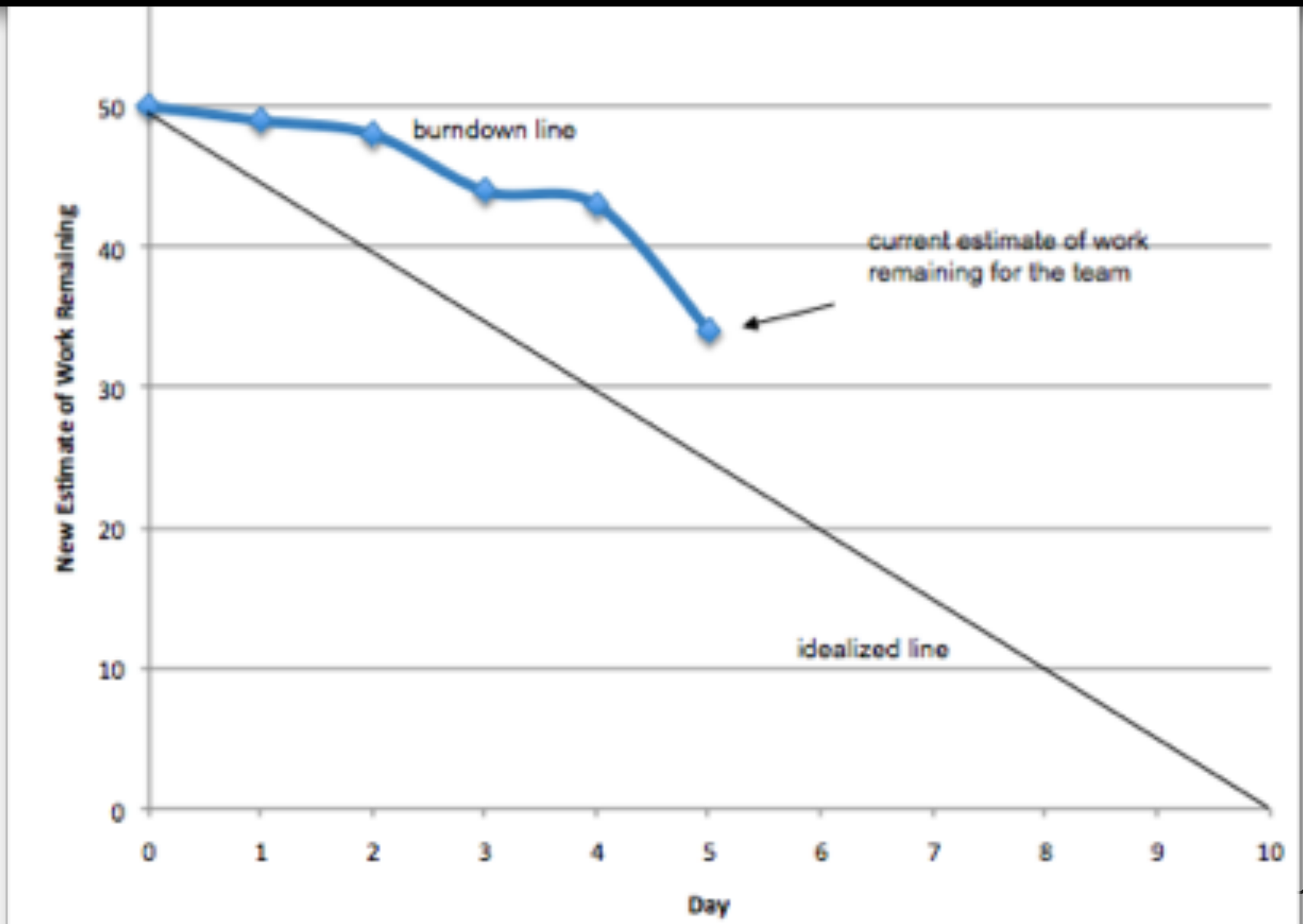
- Daily updating
  - sprint backlog
  - estimates
  - sprint burndown chart

Sprint backlog

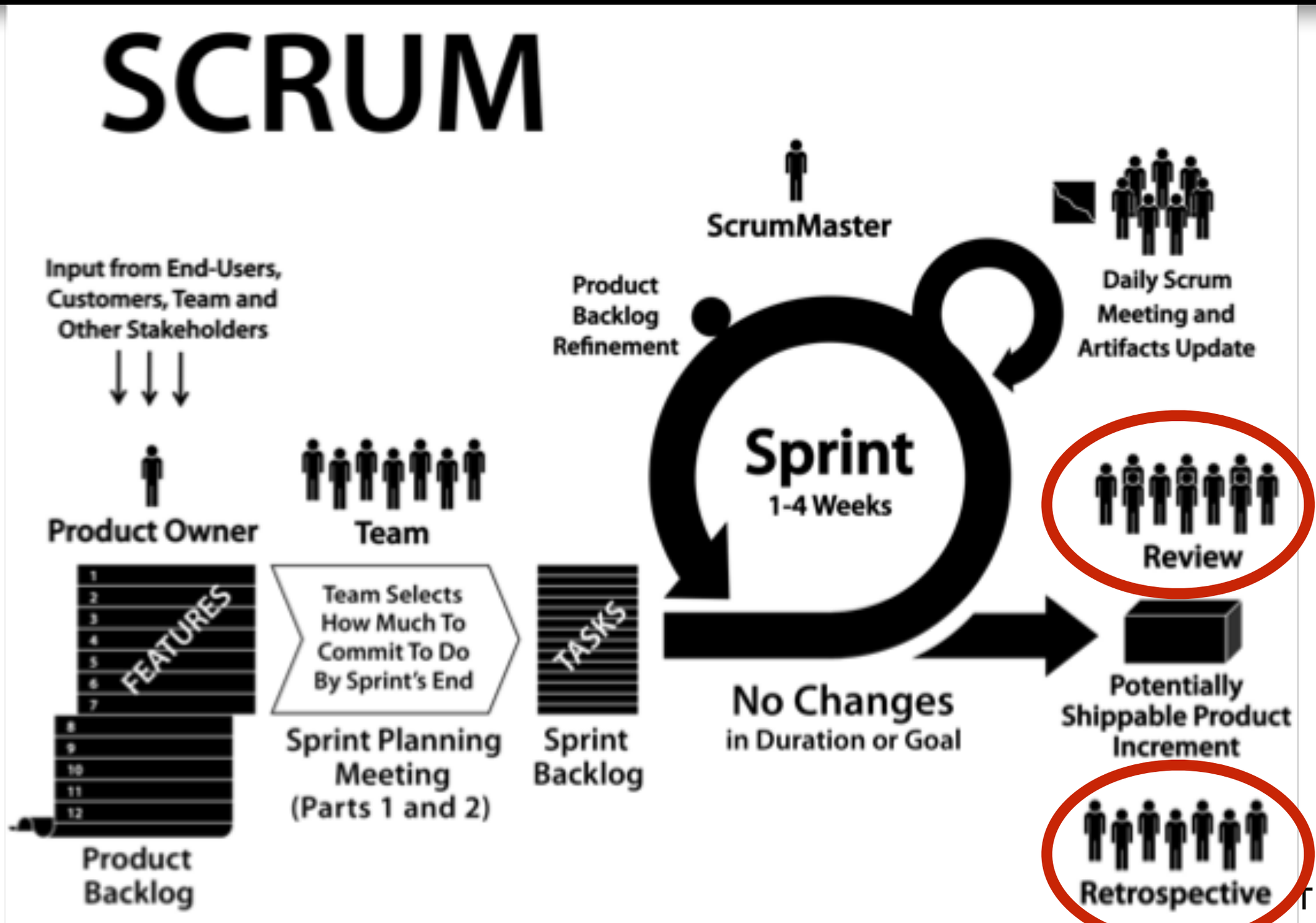
				New Estimates of Effort Remaining at end of Day...					
Product Backlog Item	**Sprint Task**	**Volunteer**	**Initial Estimate of Effort**	**1**	**2**	**3**	**4**	**5**	**6**
As a buyer, I want to place a book in a shopping cart	modify database	Sanjay	5	4	3	0	0	0	
	create webpage (UI)	Jing	3	3	3	2	0	0	
	create webpage (Javascript logic)	Tracy & Sam	2	2	2	2	1	0	
	write automated acceptance tests	Sarah	5	5	5	5	5	0	
	update buyer help webpage	Sanjay & Jing	3	3	3	3	3	0	
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5	5	5	5	5	5	
	complete machine order for pRank		3	3	8	8	8	8	
	change DCP and reader to use pRank http API		5	5	5	5	5	5	
	. . .	. . .	. . .						
		Total (person hours)	50	49	48	44	43	34	

Figure 6. Daily Updates of Work Remaining on the Sprint Backlog

Sprint burndown chart



Ending the sprint



Ending the sprint

- A sprint is never extended!
  - items that are not "done done" go back to the Backlog

- Sprint review
  - a "demo", conversation with product owner
  - no more than 30 min.
  - "inspect and adapt" wrt. the product

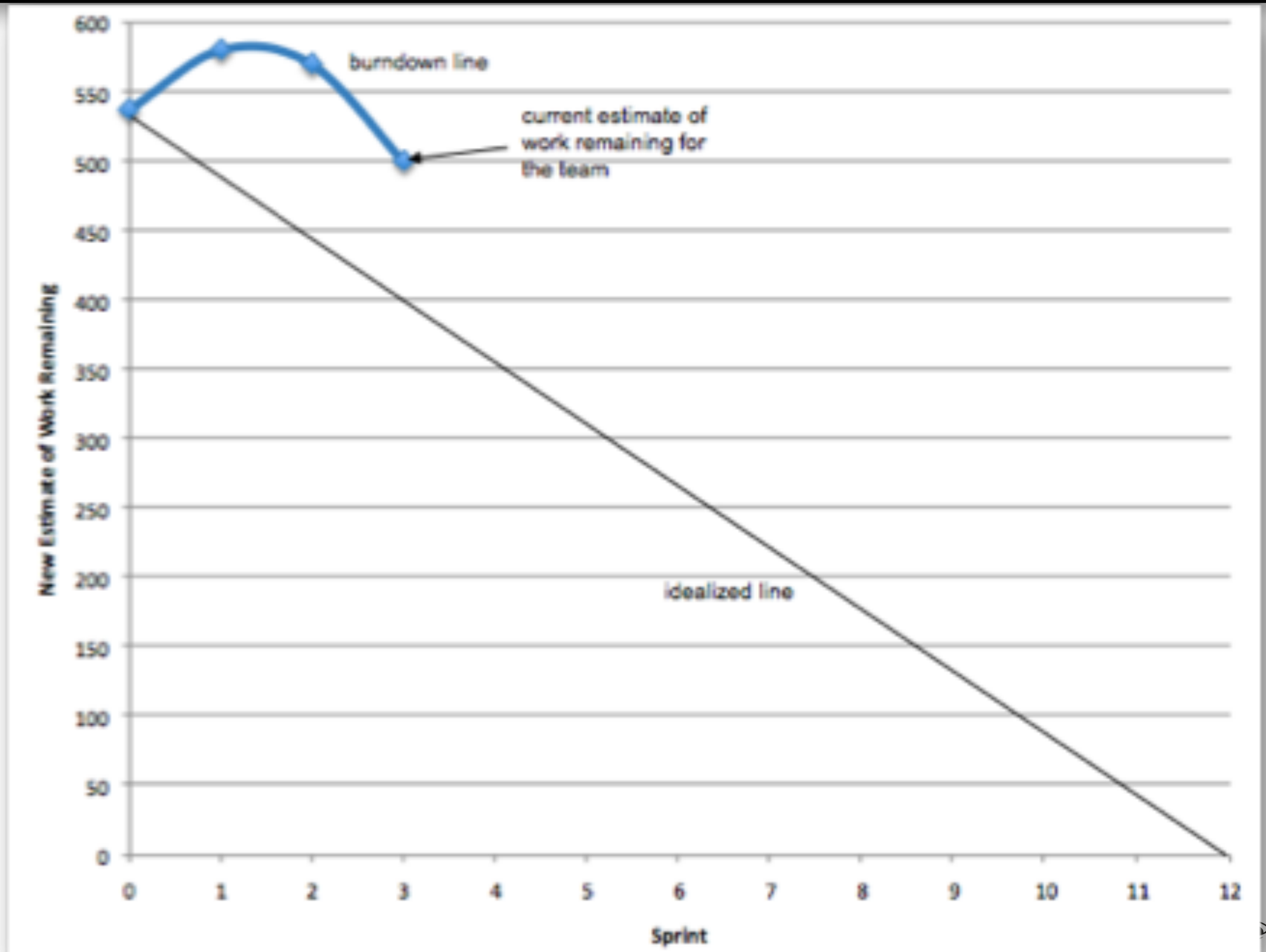
- Sprint Retrospective
  - "inspect and adapt" wrt. the process
  - discussing what is working, and what is not
  - use a whiteboard, two columns, each person adding notes

- Wrapping up
  - update the product backlog
  - update the burndown chart as the "release burndown chart"

Release backlog

					New Estimates of Effort Remaining at end of Sprint...				
Item	Details (wiki URL)	Priority	Estimate of Value	Initial Estimate of Effort	1	2	3	4	
As a buyer, I want to place a book in a shopping cart (see UI sketches on wiki page)	...	1	7	5	0	0	0		
As a buyer, I want to remove a book in a shopping cart	...	2	6	2	0	0	0		
Improve transaction processing performance (see target performance metrics on wiki)	...	3	6	13	13	0	0		
Investigate solutions for speeding up credit card validation (see target performance metrics on wiki)	...	4	6	20	20	20	0		
Upgrade all servers to Apache 2.2.3	...	5	5	13	13	13	13		
Diagnose and fix the order processing script errors (buggzilla ID 14823)	...	6	2	3	3	3	3		
As a shopper, I want to create and save a wish list	...	7	7	40	40	40	40		
As a shopper, I want to to add or delete items on my wish list	...	8	4	20	20	20	20		
. . .				. . .	. . .				
			Total	**537**	**580**	**570**	**500**		

Release burndown chart



Release sprint

- In principle, the product should be shippable after each sprint, but…

- Release sprint
  - a dedicated sprint targeted release to external customer
  - integration, scalability, documentation, …
  - QA – integration testing, testing at customer HW, …

Scrum benefits

- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team has visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

Common challenges

- "Hybrid Scrum"
  - pretending to do Scrum, but manage like a waterfall model
- Large Systems
  - larger, longer projects, multiple development teams, working in different locations & countries
- Top Reasons To Not Go Scrum
  - your development team is geographically dispersed
  - if you are currently meeting deadlines and release dates
  - if you cannot get complete buy-in or 100% commitment from management
  - if people need complete clarity about the solution before even starting the project (e.g. legal constraints)
  - you have a fixed deadline, with a fixed set of requirements

http://www.klocwork.com/blog/2009/06/top-reasons-to-not-go-scrumagile/

Concluding

Key points I

- Agile methods are
  - incremental development methods that focus on rapid development, frequent releases of the software, reducing process overheads and producing high-quality code.
  - involve the customer directly in the development process.

- The decision on whether to use an agile or a plan-driven approach to development should depend on
  - the type of software being developed,
  - the capabilities of the development team and the culture of the company developing the system.

- Extreme programming is a well-known agile method that
  - integrates a range of good programming practices such as frequent releases of the software, continuous software improvement and customer participation in the development team.
  - uses automated tests before a program feature is created. All tests must successfully execute when an increment is integrated into a system.

- The Scrum method is
  - an agile method that provides a project management framework.
  - is centred around a set of sprints, which are fixed time periods when a system increment is developed.

- Scaling agile methods for large systems is difficult. Large systems need up-front design and documentation.

Outline

- Literature
  - [SE9] ch. 2+3
- Software Process Models
- Agile software development
  - XP
  - Scrum
- Also:
  - End-user development lecture
    - [http://www.itu.dk/Om-ITU/Events/Events/2015/Open-lecture-End-User-Software-Engineering---Beyond-the-Silos]
  - Course evaluation


time to evaluate