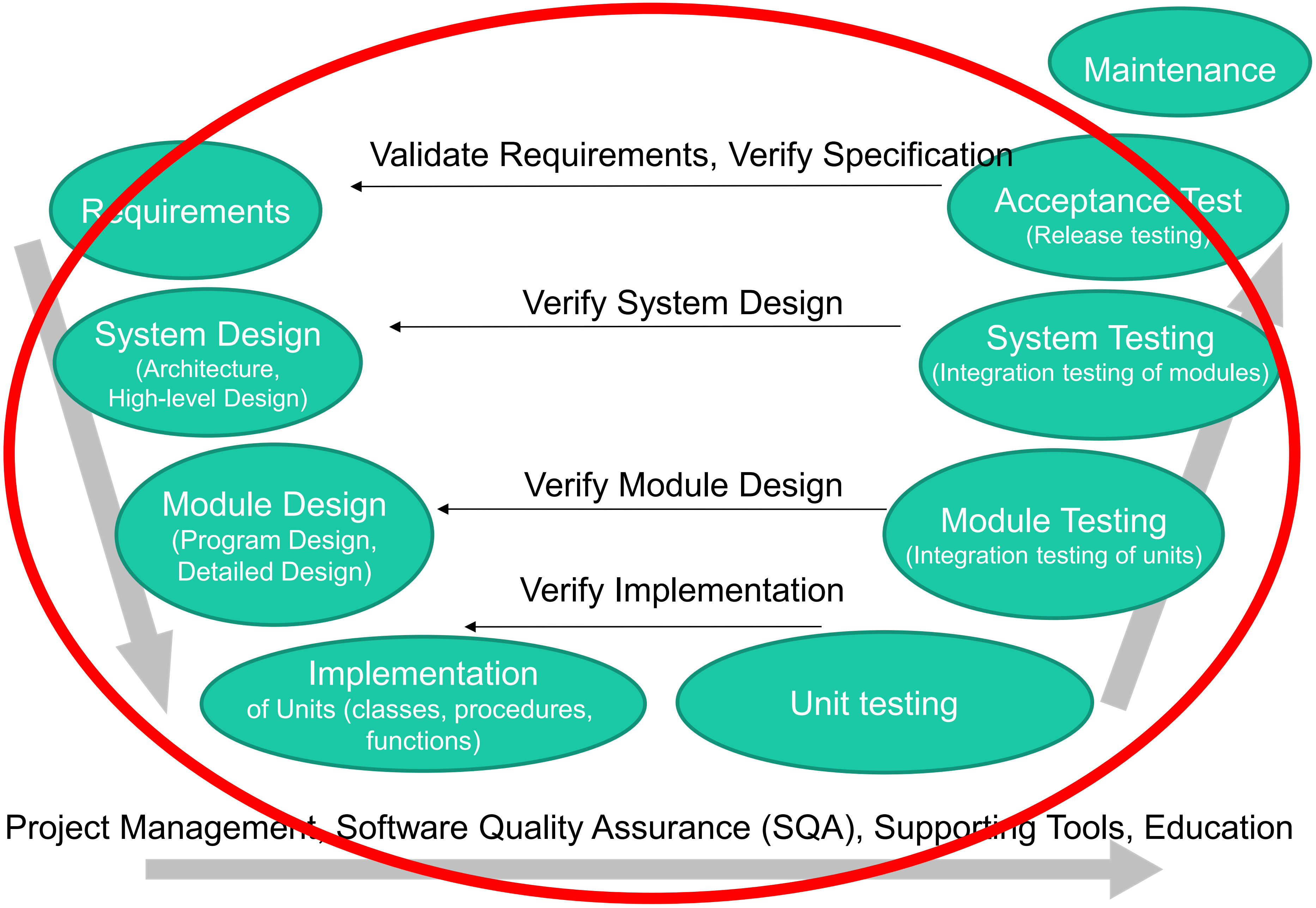
Processes and Life-Cycles

Kristian Sandahl

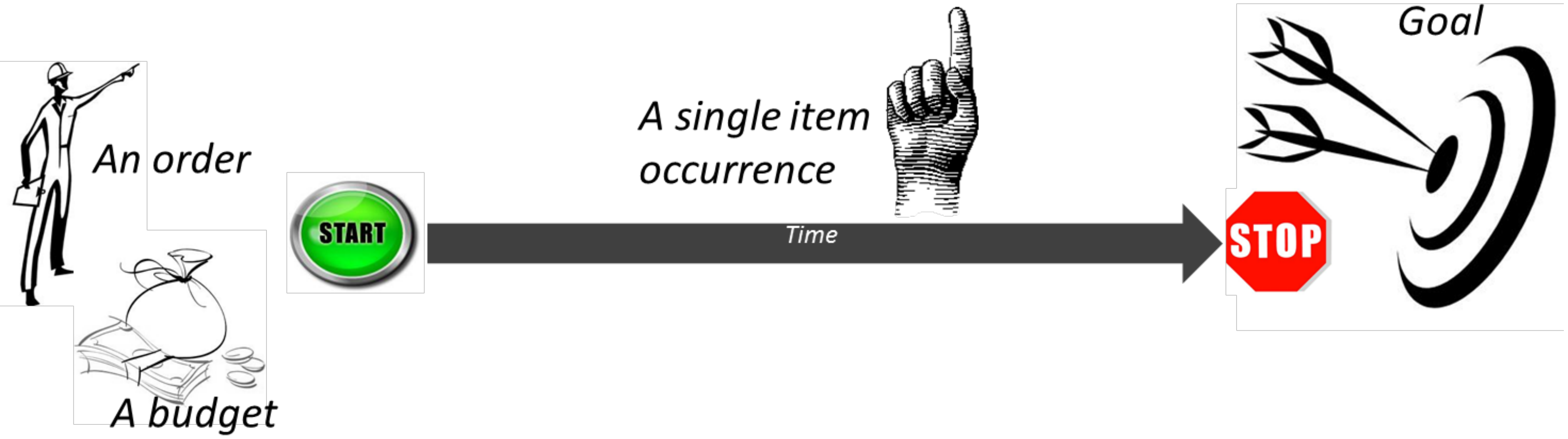
LINKÖPING UNIVERSITY

Agenda:

- Definition of process
- Life-cycle models
  - V and Waterfall
  - Incremental and Iterative
- Method frameworks
  - OpenUP
  - Essence Kernel
  - eXtreme Programming
  - SCRUM
  - KANBAN

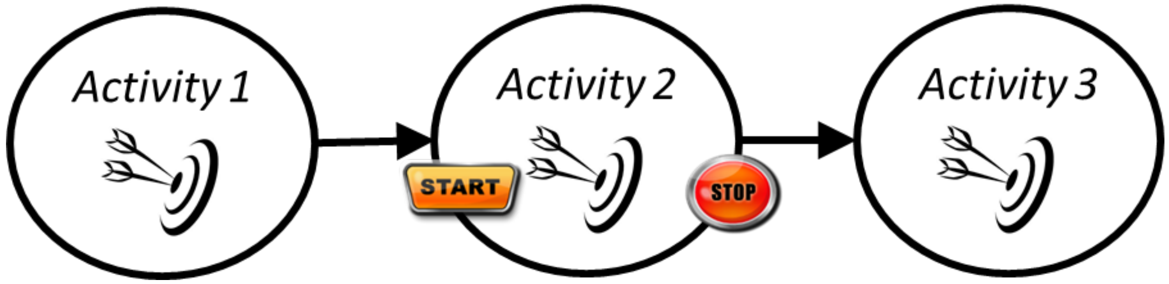

Maintenance
Validate Requirements, Verify Specification
Requirements
Acceptance Test
(Release testing)
Verify System Design
System Design
(Architecture,
High-level Design)
System Testing
(Integration testing of modules)
Verify Module Design
Module Design
(Program Design,
Detailed Design)
Module Testing
(Integration testing of units)
Verify Implementation
Implementation
of Units (classes, procedures,
functions)
Unit testing
Project Management, Software Quality Assurance (SQA), Supporting Tools, Education

Remember the necessary parts of a **project**?



Processes are reoccurring

- Ordered set of activities
- May contain sub-processes
- Goal of each activity
- Each activity has entry/exit criteria and input/output
- Constraints



Agenda:

~~Definition of process~~

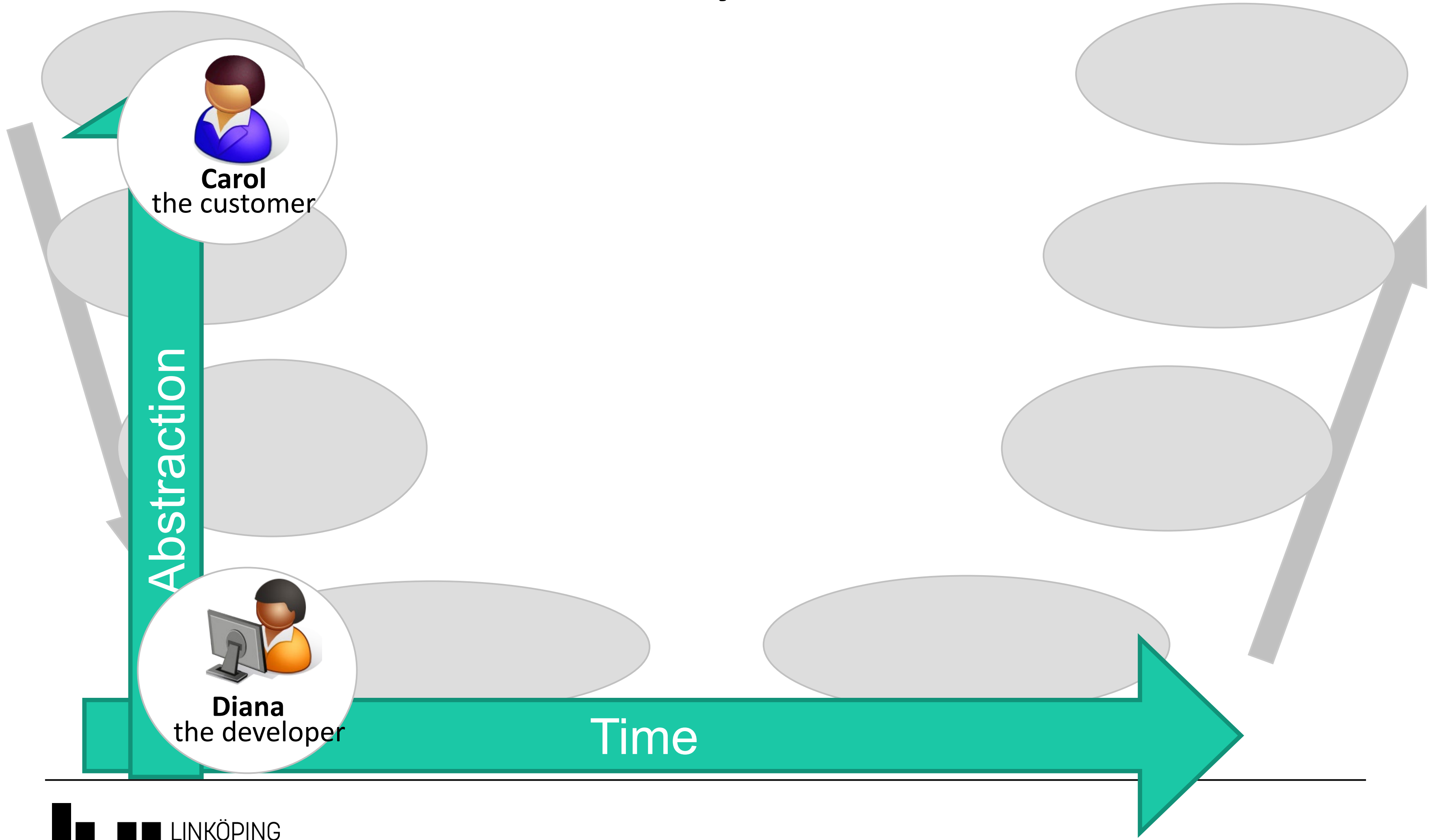
Life-cycle models

- V and Waterfall
- Incremental and Iterative

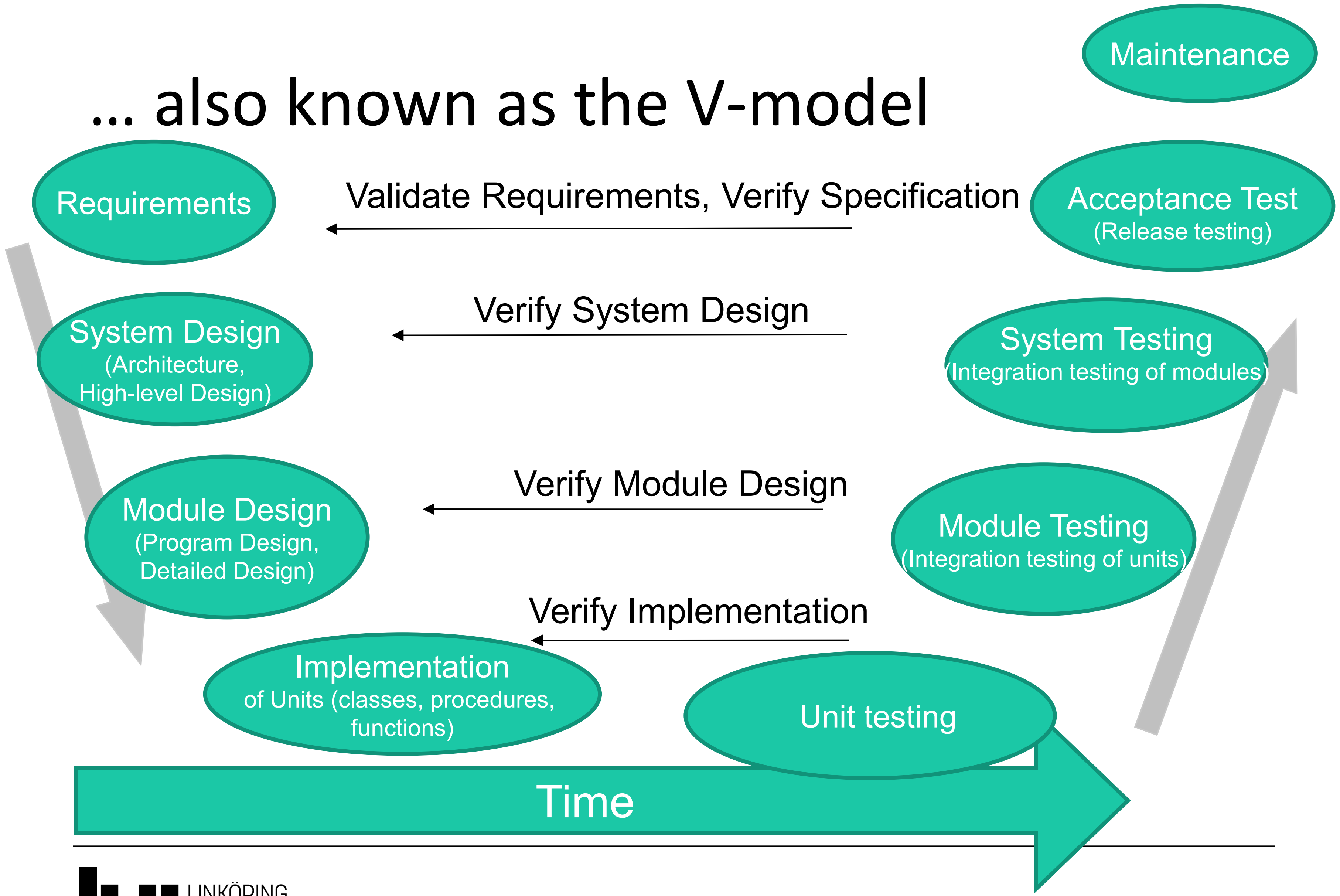
Method frameworks

- OpenUP
- Essence Kernel
- eXtreme Programming
- SCRUM
- KANBAN

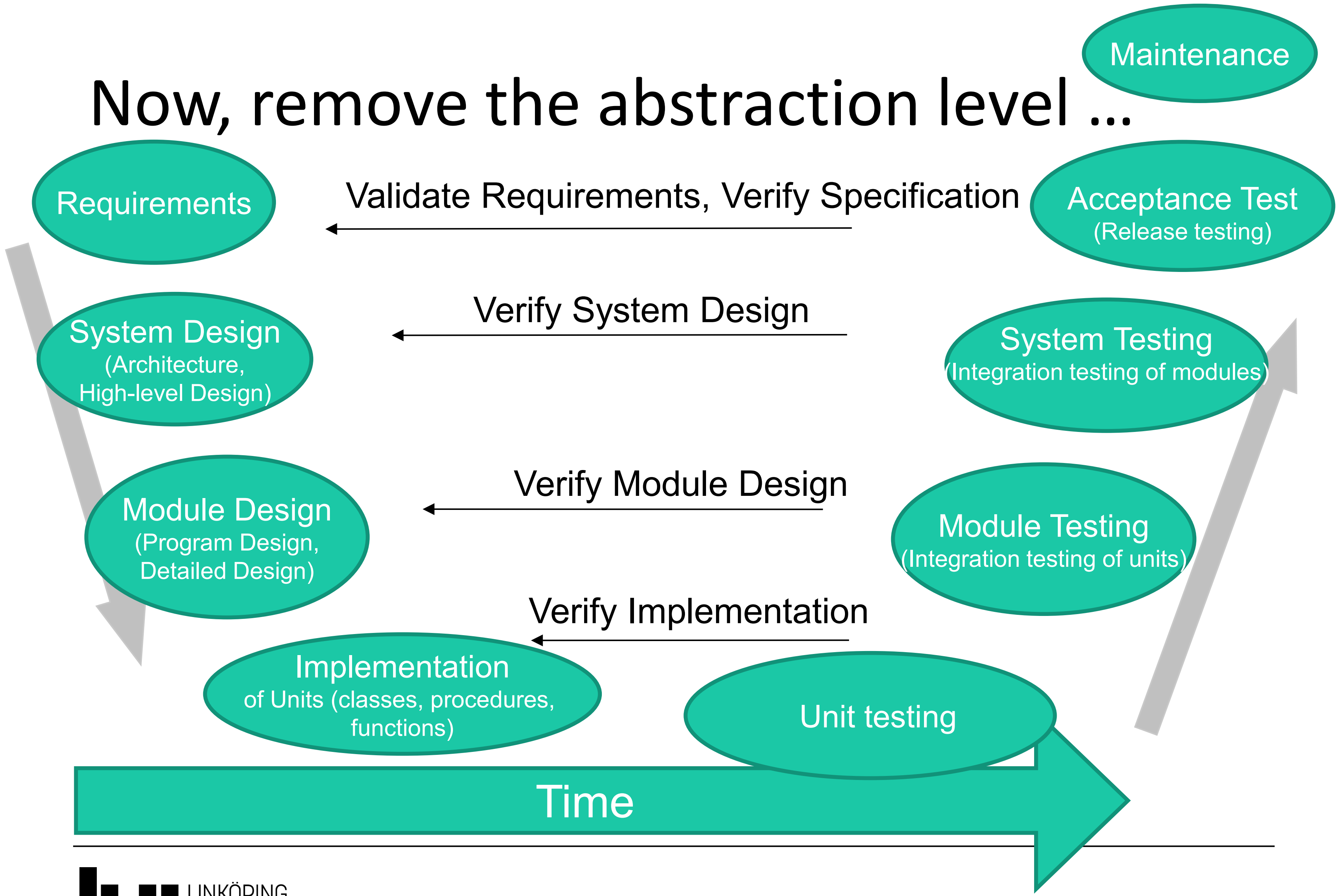
Remember our life-cycle model?...



… also known as the V-model



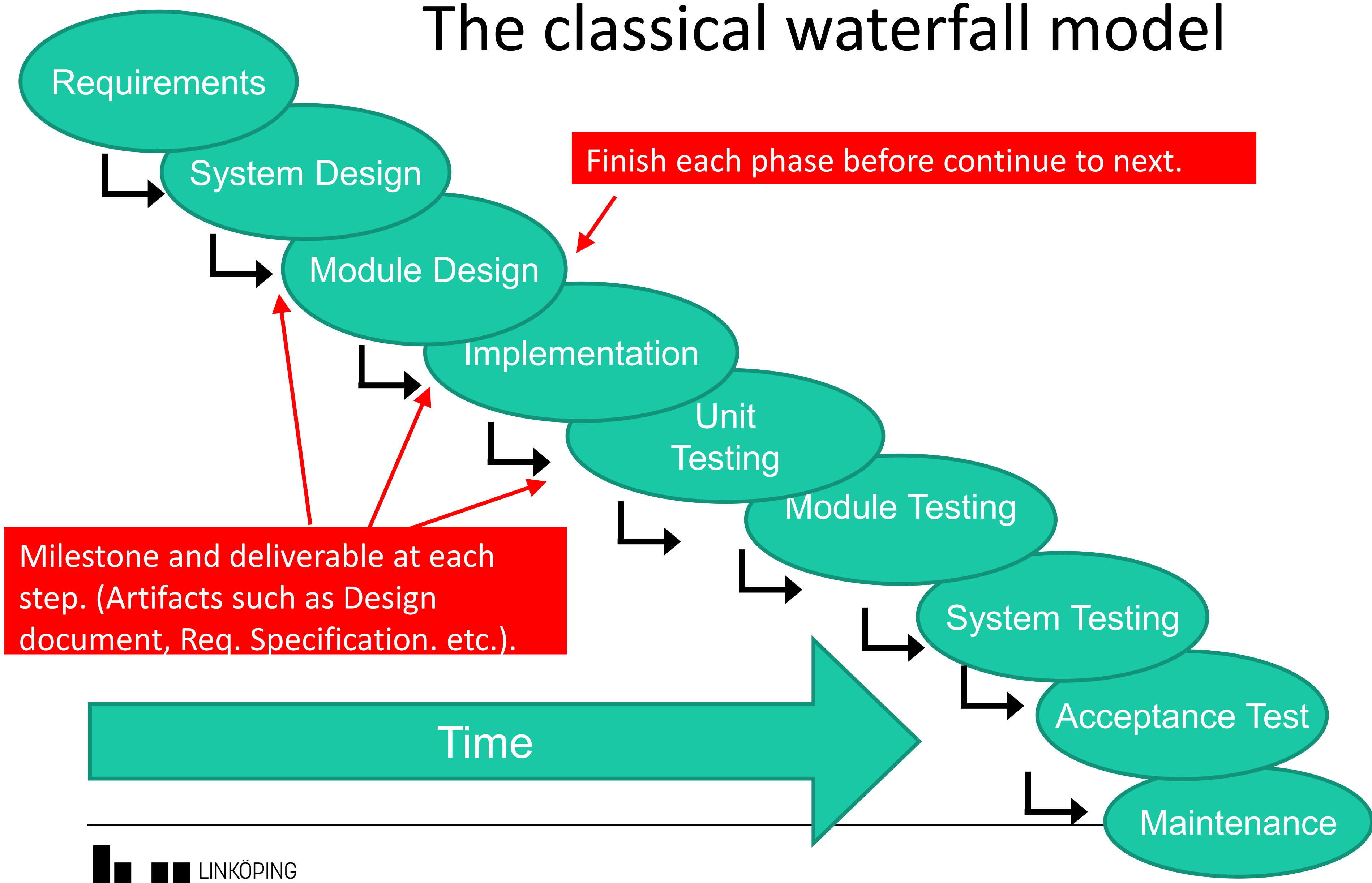
Now, remove the abstraction level …



… and we got the waterfall model!

- One of the first life-cycle models (Royce, 1970)
- The waterfall development model originates in the manufacturing and construction industries
- Very common, very criticized

The classical waterfall model



Finish each phase before continue to next.

Milestone and deliverable at each step. (Artifacts such as Design document, Req. Specification. etc.).

What are the potential drawbacks of the classical waterfall model?

https://www.menti.com/bpsk2rjzxt

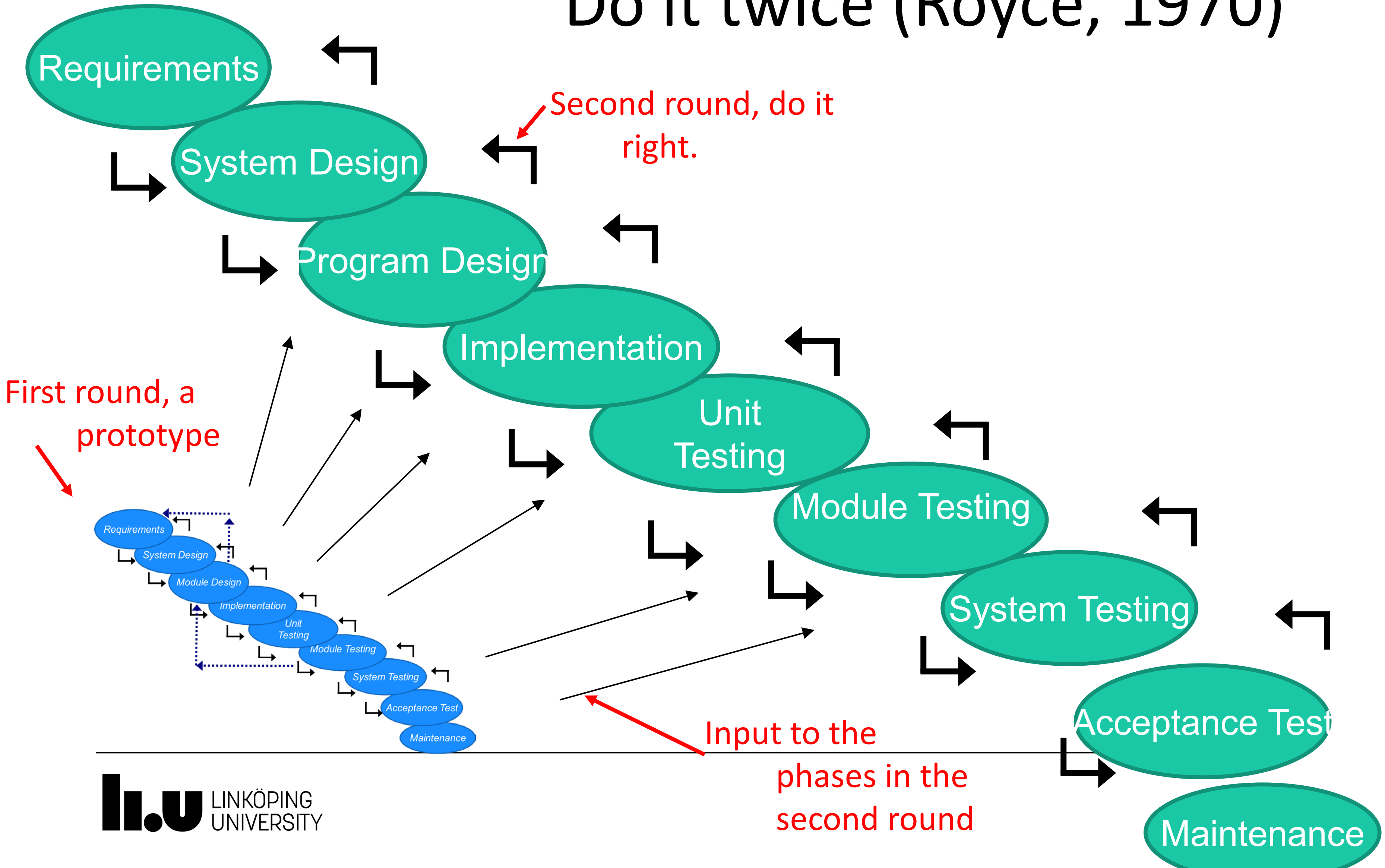
Problems with the waterfall model

- Software requirements change, hard to sign-off on a SRS.
- Early commitment. Changes at the end, large impact.
- Feedback is needed to understand a phase. E.g. implementation is needed to understand some design.
- Difficult to estimate time and cost for the phases.
- Handling risks is not an explicit part of the model. Pushes the risks forward.
- Software "is not" developed in such a way. It evolves when problems are more understood. Little room for problem solving.

Advantages with the waterfall model

- Simple, manageable and easy to understand
- Fits to common project management practices (milestones, deliverables etc.)
- Can be suitable for short projects (some weeks)
- Can be used at a large system level (several years)
- Can be suitable for "stable" projects, where requirements do not change
- Focus on documents, saves knowledge which can be reused by other people.
- Can be suitable for fixed-price contracts

Do it twice (Royce, 1970)



Agenda:

Definition of process

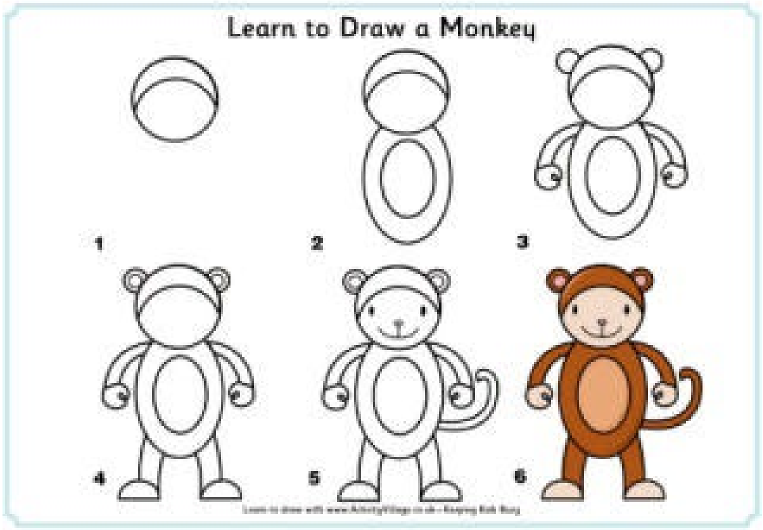
Life-cycle models

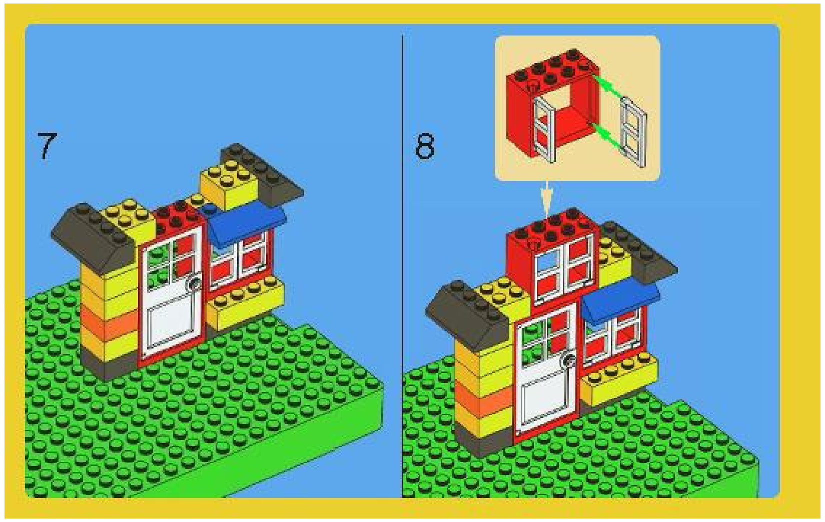
- ~~V and Waterfall~~
- Incremental and Iterative

Method frameworks

- OpenUP
- Essence Kernel
- eXtreme Programming
- SCRUM
- KANBAN

Iterative and Incremental methods





Sources: Activity village and Lego

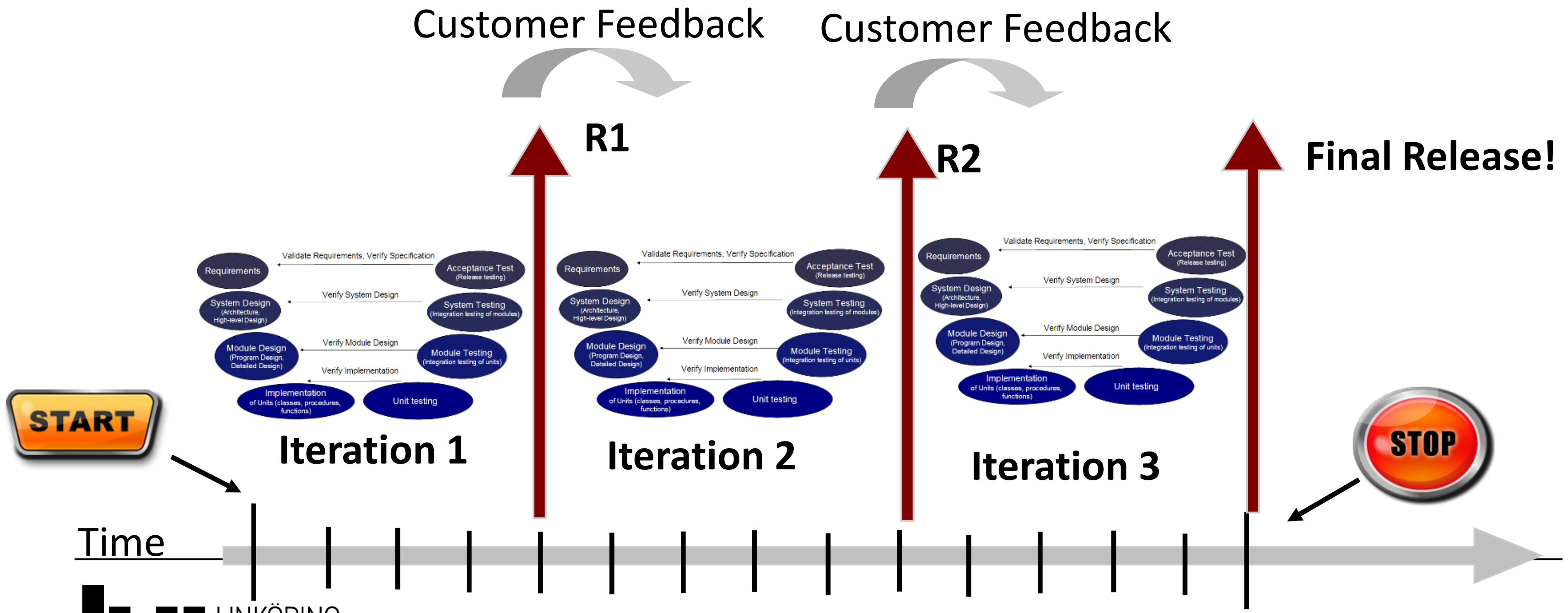
Iterative development

When **should the releases take place?**

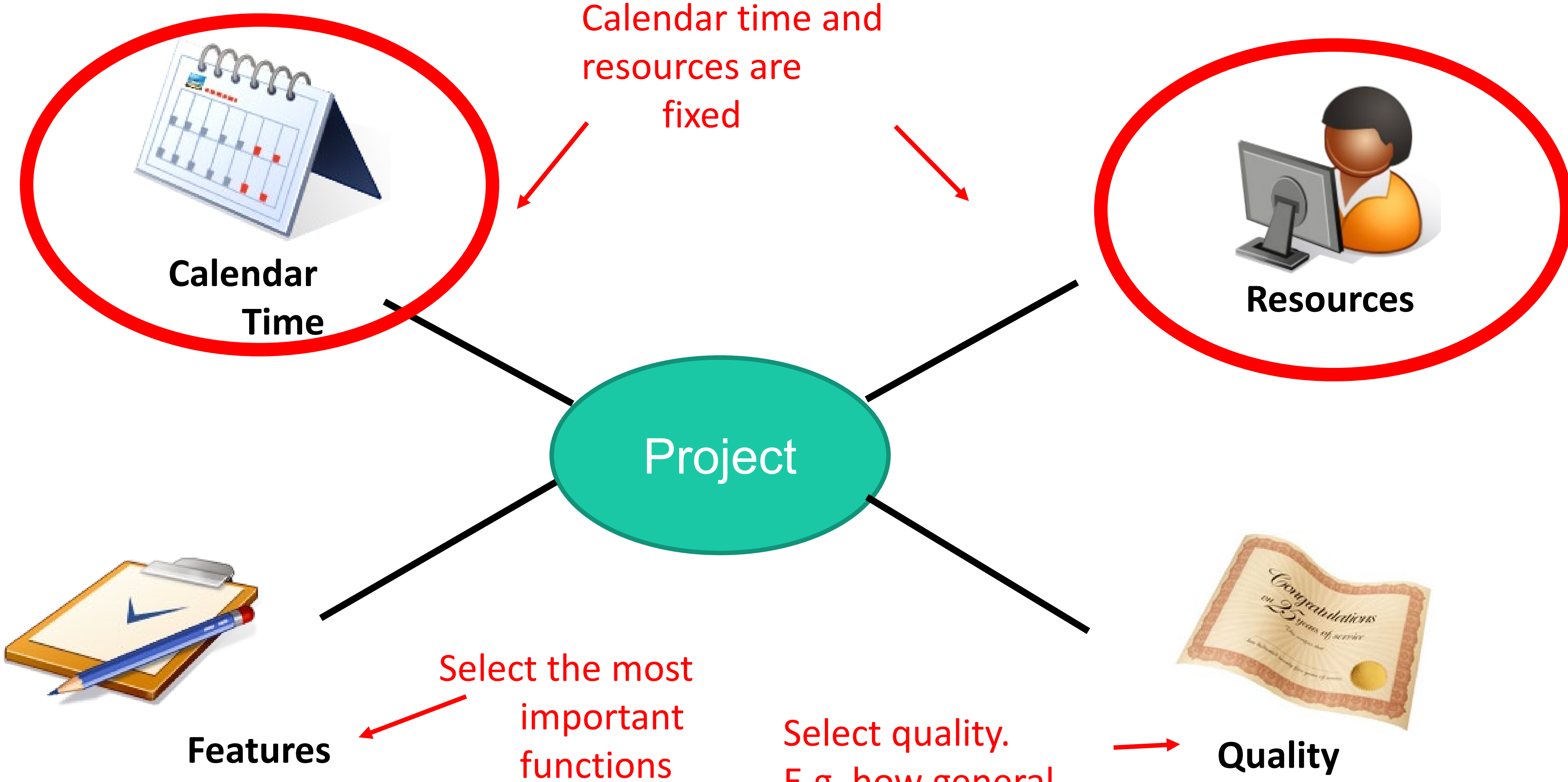
Time-boxing - The time period is fixed for each iteration.

What **should be included in the release?**

Prioritized functionality - Do the most important parts first.


Customer Feedback
Customer Feedback
R1
R2
Final Release!
START
STOP
Requirements
Validate Requirements, Verify Specification
Acceptance Test (Release testing)
System Design (Architecture, High-level Design)
Verify System Design
System Testing (Integration testing of modules)
Module Design (Program Design, Detailed Design)
Verify Module Design
Module Testing (Integration testing of units)
Verify Implementation
Implementation of Units (classes, procedures, functions)
Unit testing
Iteration 1
Iteration 2
Iteration 3
Time

Dependent project parameters revisited

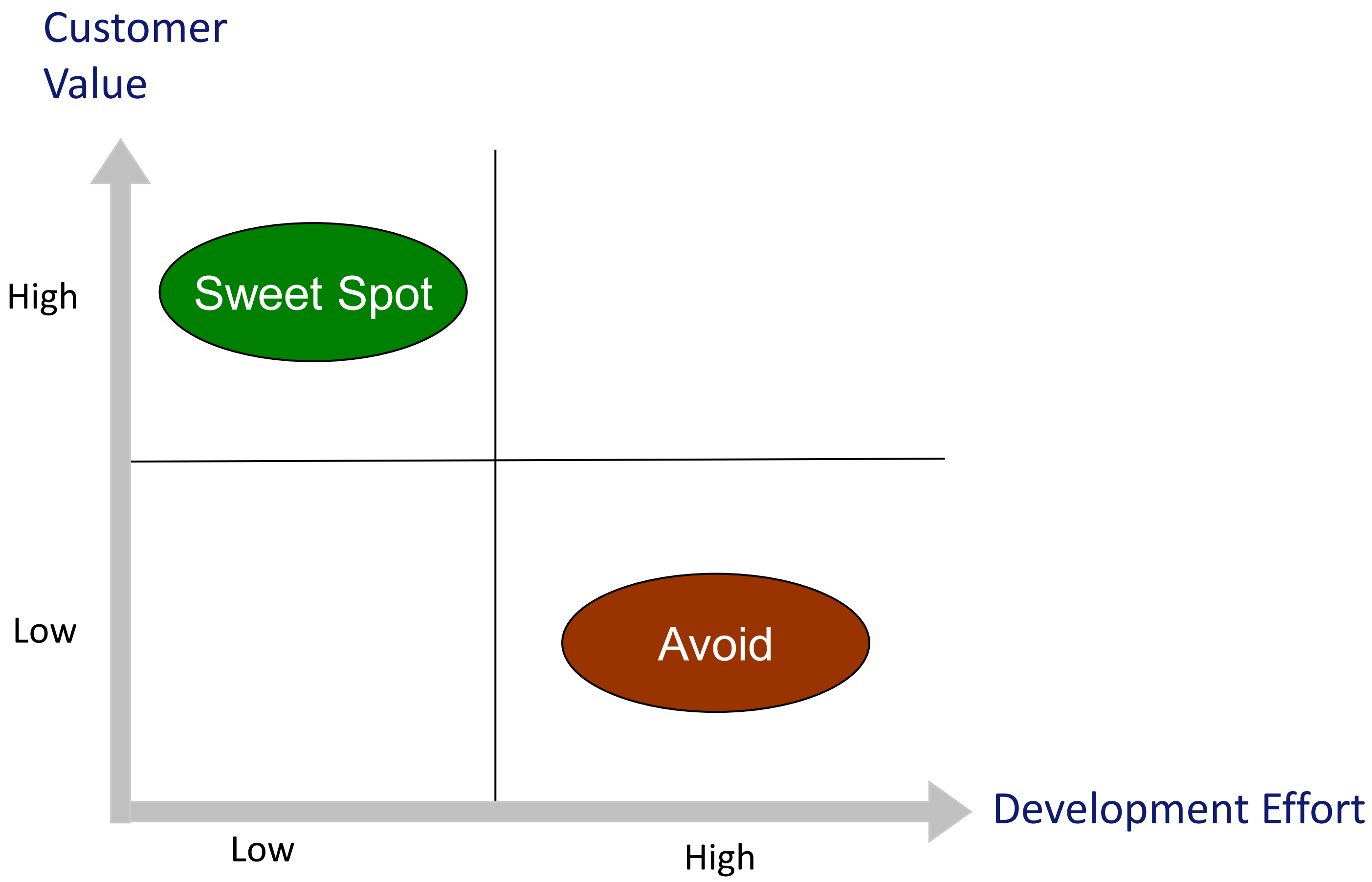


Calendar time and resources are fixed

Select the most important functions

Select quality. E.g. how general should we be?

Prioritization of requirements



Problems with iterative development

- Problem with current business contracts, especially fixed-price contracts.
- With short iterations it can be hard to map customer requirements to iterations.
- Overhead added
- Requirements selection problem
- Stressful learning period if moving from the classical waterfall model

What are the advantages of iterative development?

- Discuss in pairs

Advantages with iterative development

- Misunderstandings and inconsistency are made clear early (e.g. between requirement, design, and implementation)
- Encourage to use feedback -> elicit the *real* requirements
- Forced to focus on the most critical issues
- Continuous testing offers project assessment
- Workload is spread out over time (especially test)
- The team can get "lesson learned" and continuously improve the process
- Stakeholders get concrete evidence of progress

Agenda:

~~Definition of process~~

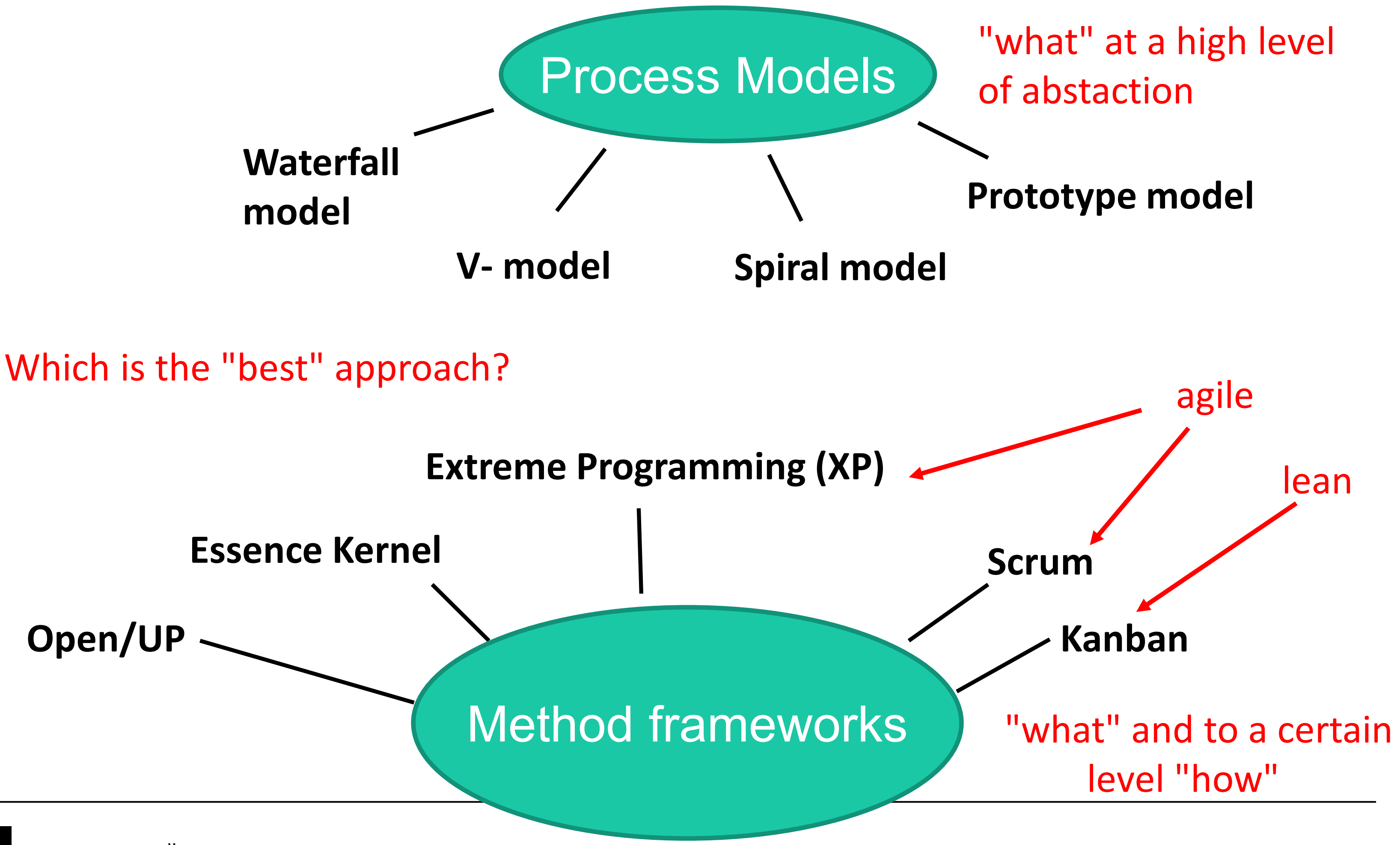
~~Life-cycle models~~

- ~~V and Waterfall~~
- ~~Incremental and Iterative~~

Method frameworks

- OpenUP
- Essence Kernel
- eXtreme Programming
- SCRUM
- KANBAN

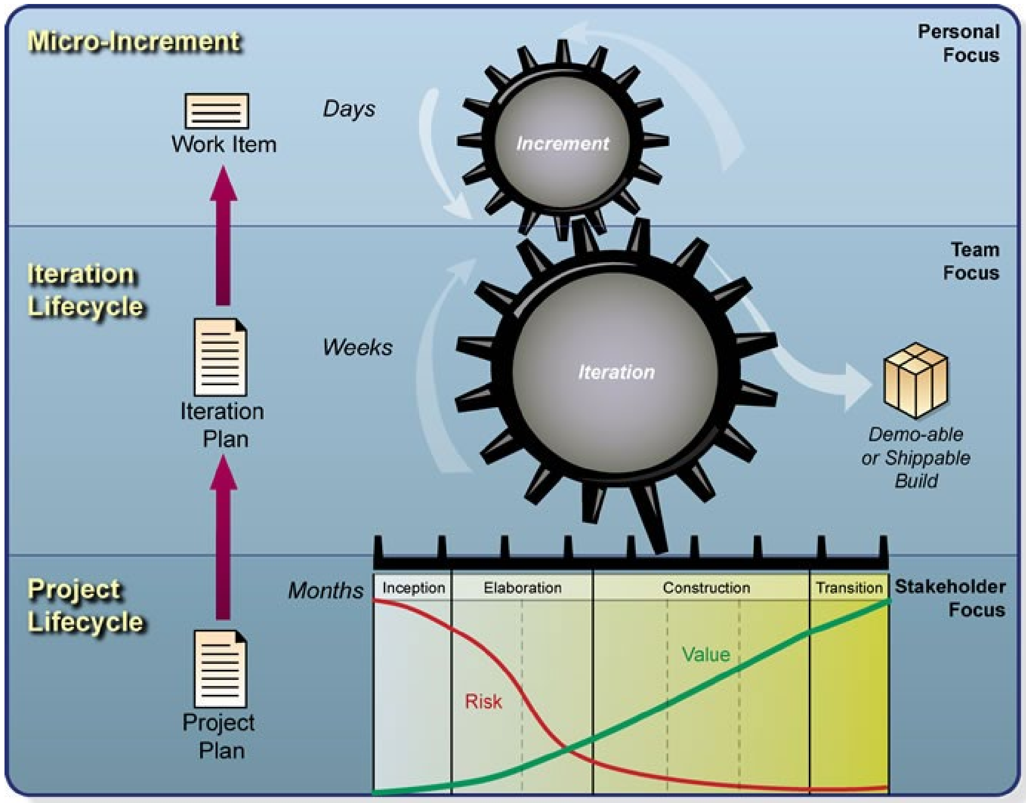
Processes, models, methodologies…



Open/UP mimics the way software is developed

- Down-scaled variant of RUP
- Mapping
  - Roles
  - Tasks
  - Workproducts

The architecture notebook



Agenda:

~~Definition of process~~

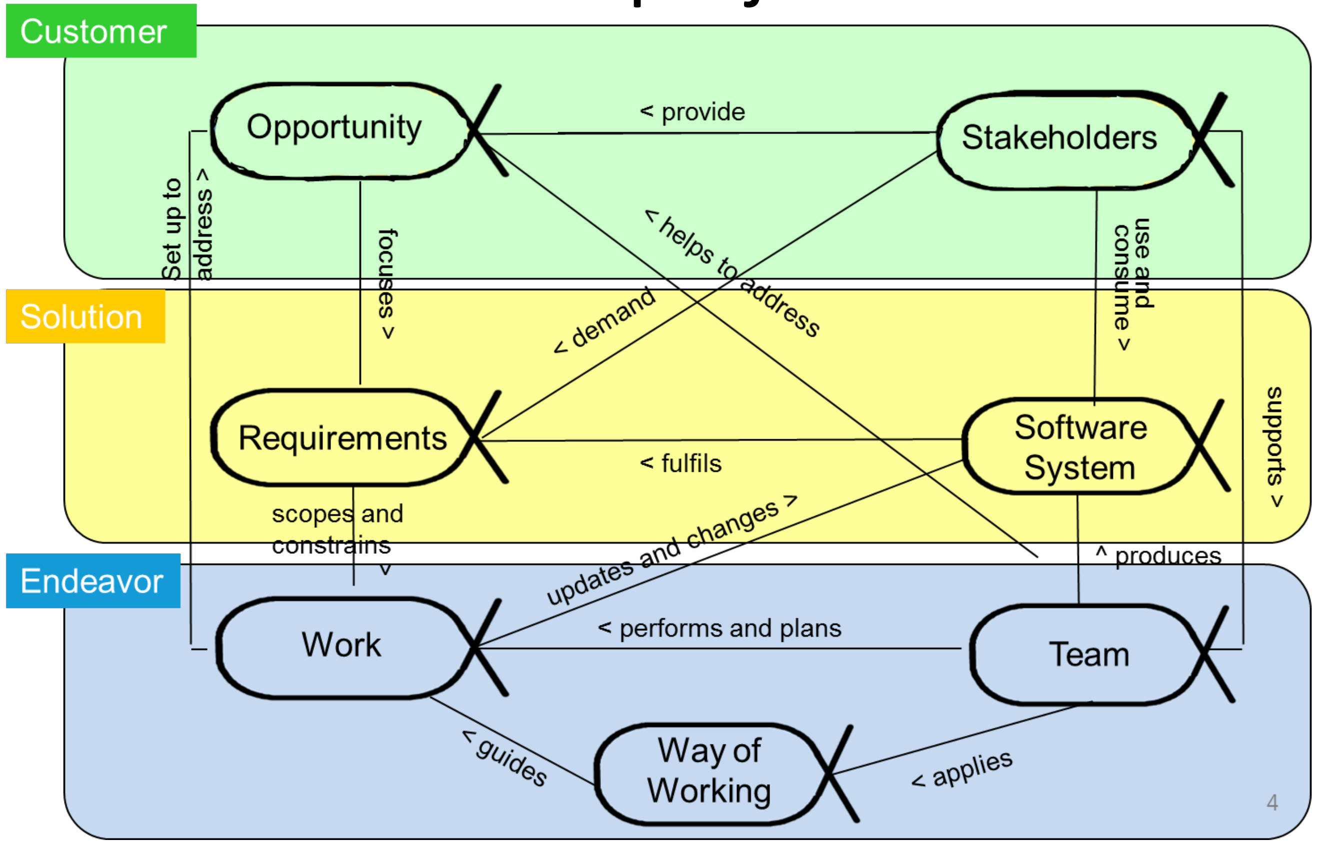
~~Life-cycle models~~

- ~~V and Waterfall~~
- ~~Incremental and Iterative~~

Method frameworks

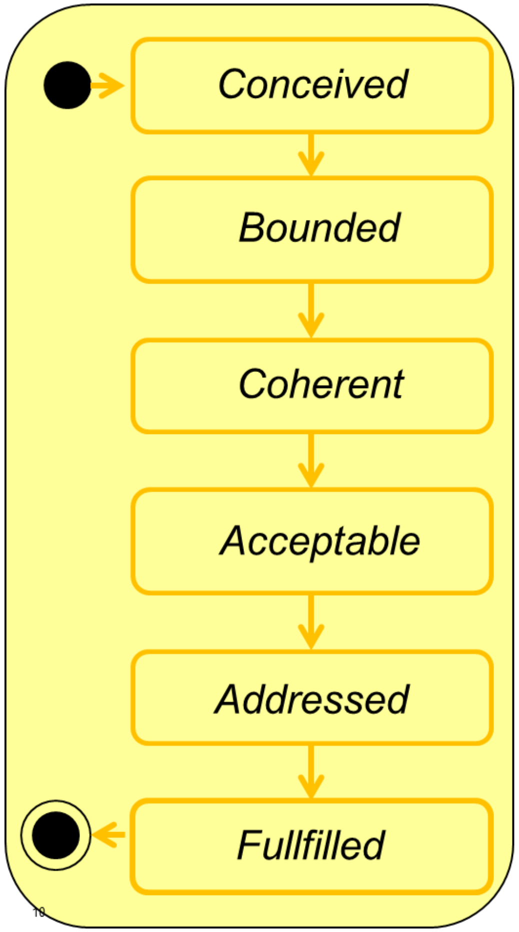
- ~~OpenUP~~
- Essence Kernel
- eXtreme Programming
- SCRUM
- KANBAN

Essence Kernel monitors the common denominator of all SE projects



Source of picture: Ivar Jacobsson International

Requirements – states



The need for a new system has been agreed.

The purpose and theme of the new system are clear.

The requirements provide a coherent description of the essential characteristics of the new system.

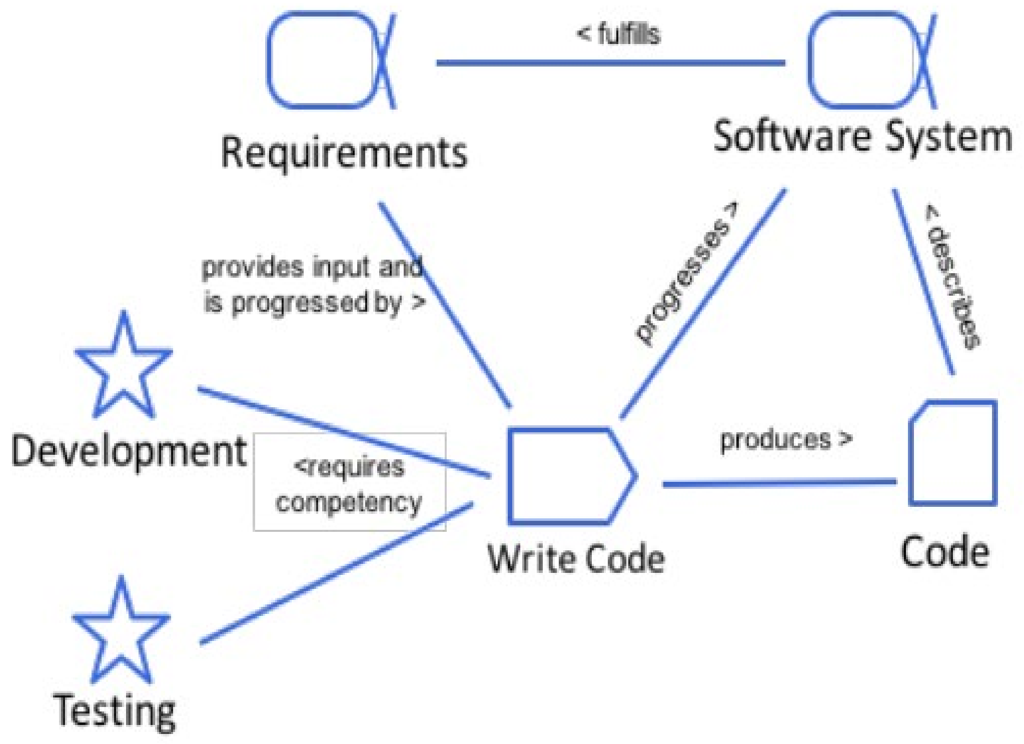
The requirements describe a system that is acceptable to the stakeholders.

Enough of the requirements have been addressed to satisfy the need for a new system in a way that is acceptable to the stakeholders.

The requirements have been addressed to fully satisfy the need for a new system.

Source of picture: Ivar Jacobsson International

Snap-shot of relations between elements – Practices



Agenda:

~~Definition of process~~

~~Life-cycle models~~

- ~~V and Waterfall~~
- ~~Incremental and Iterative~~

Method frameworks

- ~~OpenUP~~
- ~~Essence Kernel~~
- eXtreme Programming
- SCRUM
- KANBAN

Agile Approaches - Agile Alliance

Lightweight approaches to satisfy the customers with "early and continuous delivery of valuable software"

Manifesto for Agile Software Development

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

(http://agilemanifesto.org, 2001)

Extreme Programming

- Formulated in 1999 by Kent Beck
- XP is “a light-weight methodology for small to medium-sized teams developing software in the face of vague or rapidly changing requirements.”
- Driving good habits to the extreme

XP Values

- Communication
  - On-site customer, user stories, pair programming, daily standup meetings, etc.
- Simplicity
  - "Do the simplest thing that could possibly work" (DTSTTCPW) principle
- Feedback
  - Unit tests tell programmers status of the system
  - Programmers produce new releases every 2-3 weeks for customers to review
- Courage
  - Communicate and accept feedback, throw code away, refactor the architecture of a system

XP- Some Practices

Pair Programming

- Programming as a collaborative conversation
- Focus on task
- Clarify ideas
- Rotate frequently

Stories

- "requirements", but not mandatory
- a token for a piece of system capability to be implemented
- Name + short story
- On index cards (paper)

Refactoring

- Improve the design of existing code without changing its functionality
- Tool support, e.g. Eclipse

Continuous Integration

- Integrate and test often
- Automated build system
- Automated regression tests (e.g. JUnit)

Test-First Programming

- Create tests before code
- Focus on interface and "what is needed"
- Gets tests for free

Agenda:

~~Definition of process~~

~~Life-cycle models~~

- ~~V and Waterfall~~
- ~~Incremental and Iterative~~

Method frameworks

- ~~OpenUP~~
- ~~Essence Kernel~~
- ~~eXtreme Programming~~
- SCRUM
- KANBAN

Scrum

Approach public in 1995 at OOPSLA

"Scrum" strategy used in rugby for getting an out-of-play ball back into play.

Scrum in a nutshell

Small, cross-functional teams

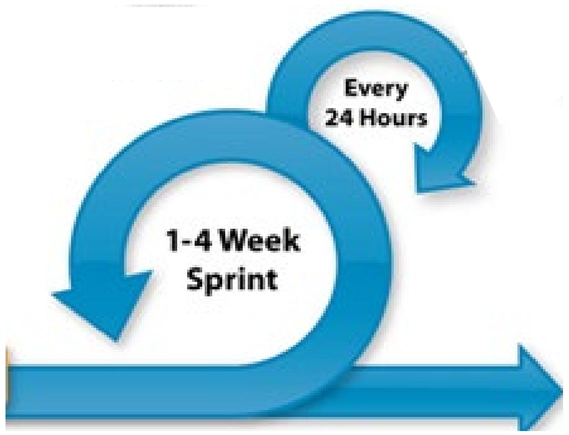
Product split into small, roughly estimated, stories

Iterations - sprints

Continuous improvement and deployment

Slides by Aseel Berglund

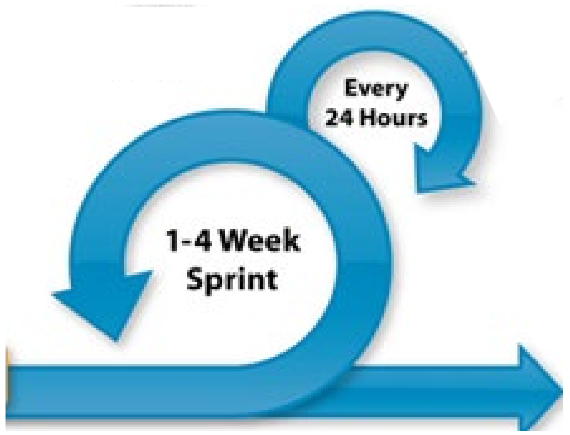
The Sprint



Sprint end date and deliverable do not change

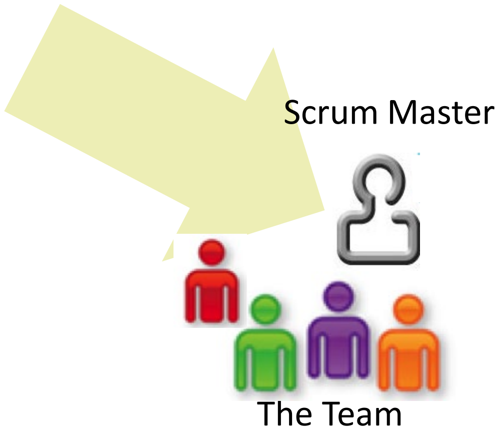
The Team

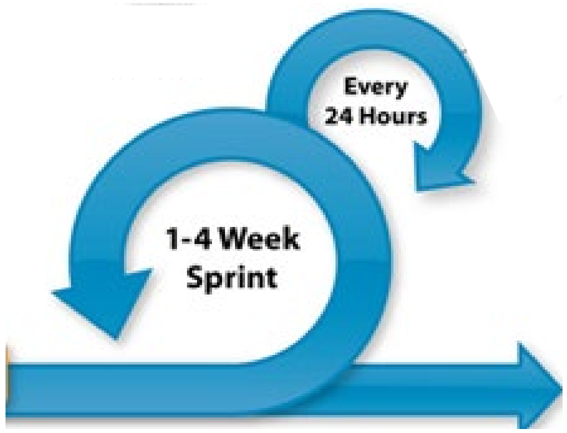
The Team



Sprint end date and deliverable do not change

The Scrum Master





Sprint end date and deliverable do not change

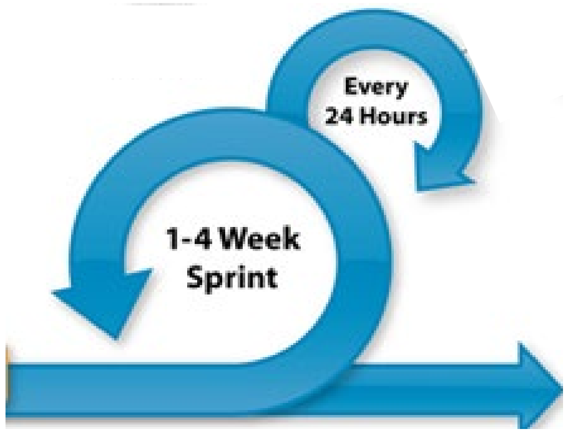
The Product Owner

Inputs from Executives, Stakeholders, Customers, Users, Team

Product Owner

Scrum Master

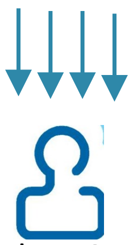
The Team



Sprint end date and deliverable do not change

The Product Backlog

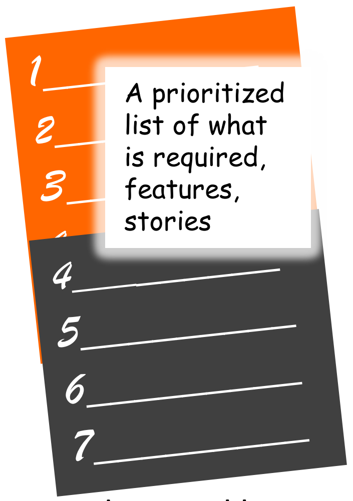
Inputs from Executives, Stakeholders, Customers, Users, Team



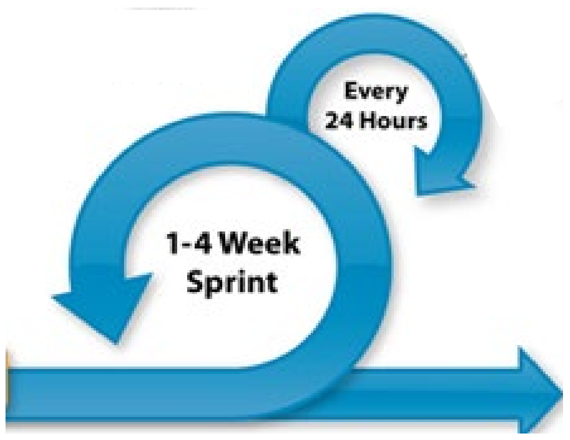
Product Owner

Scrum Master

The Team



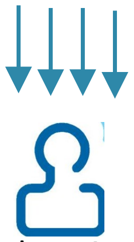
Product Backlog



Sprint end date and deliverable do not change

The Sprint Planning Meeting

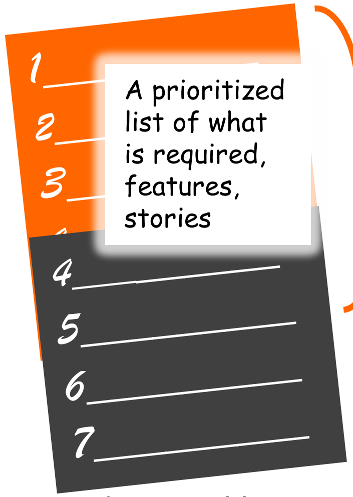
Inputs from Executives, Stakeholders, Customers, Users, Team



Product Owner

Scrum Master

The Team

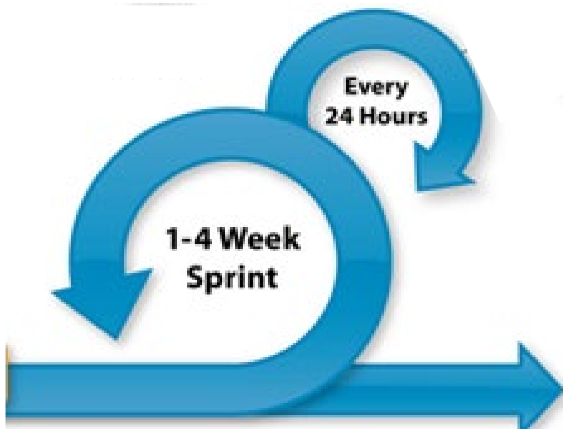


A prioritized list of what is required, features, stories

Product Backlog

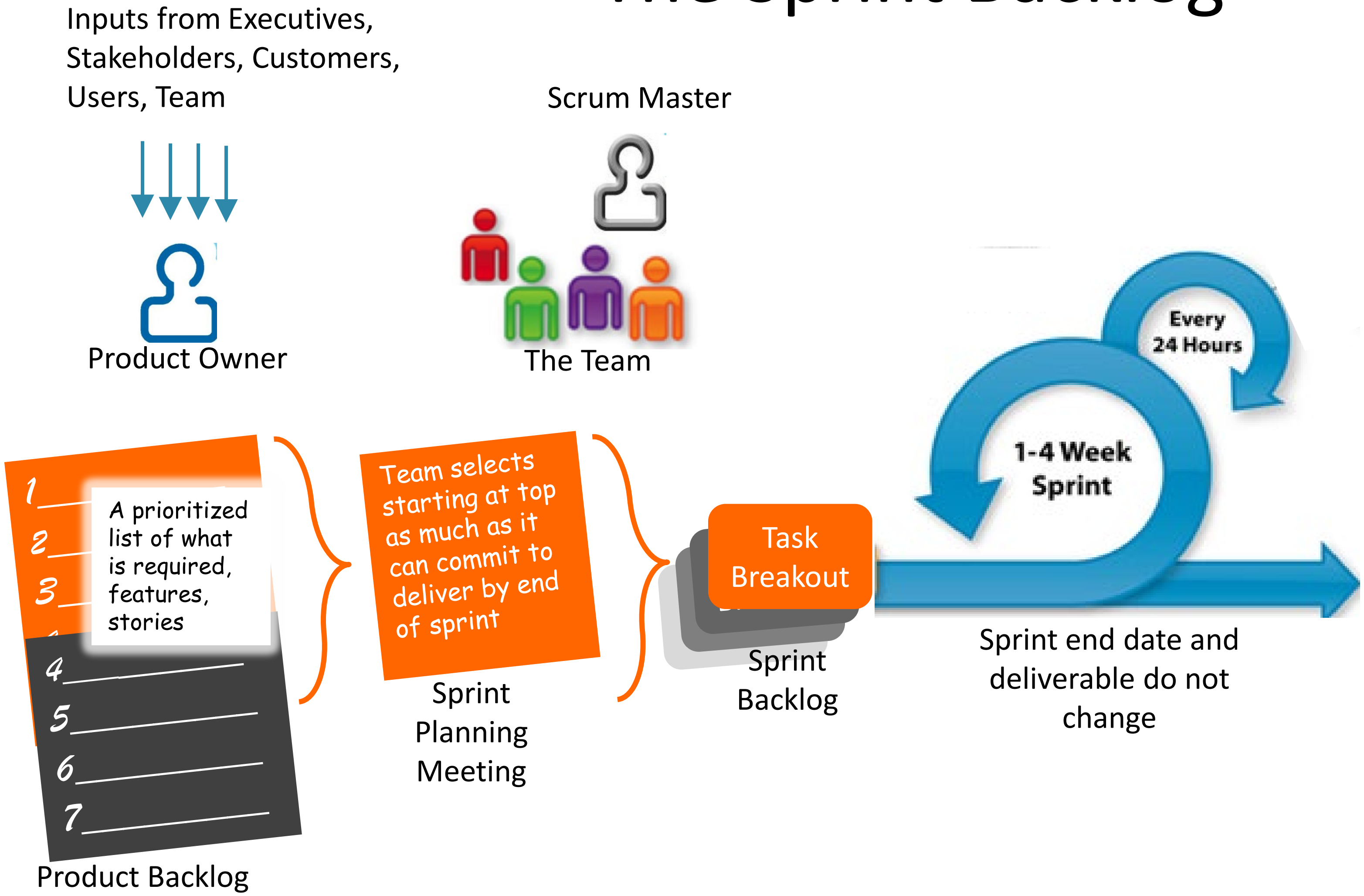
Team selects starting at top as much as it can commit to deliver by end of sprint

Sprint Planning Meeting

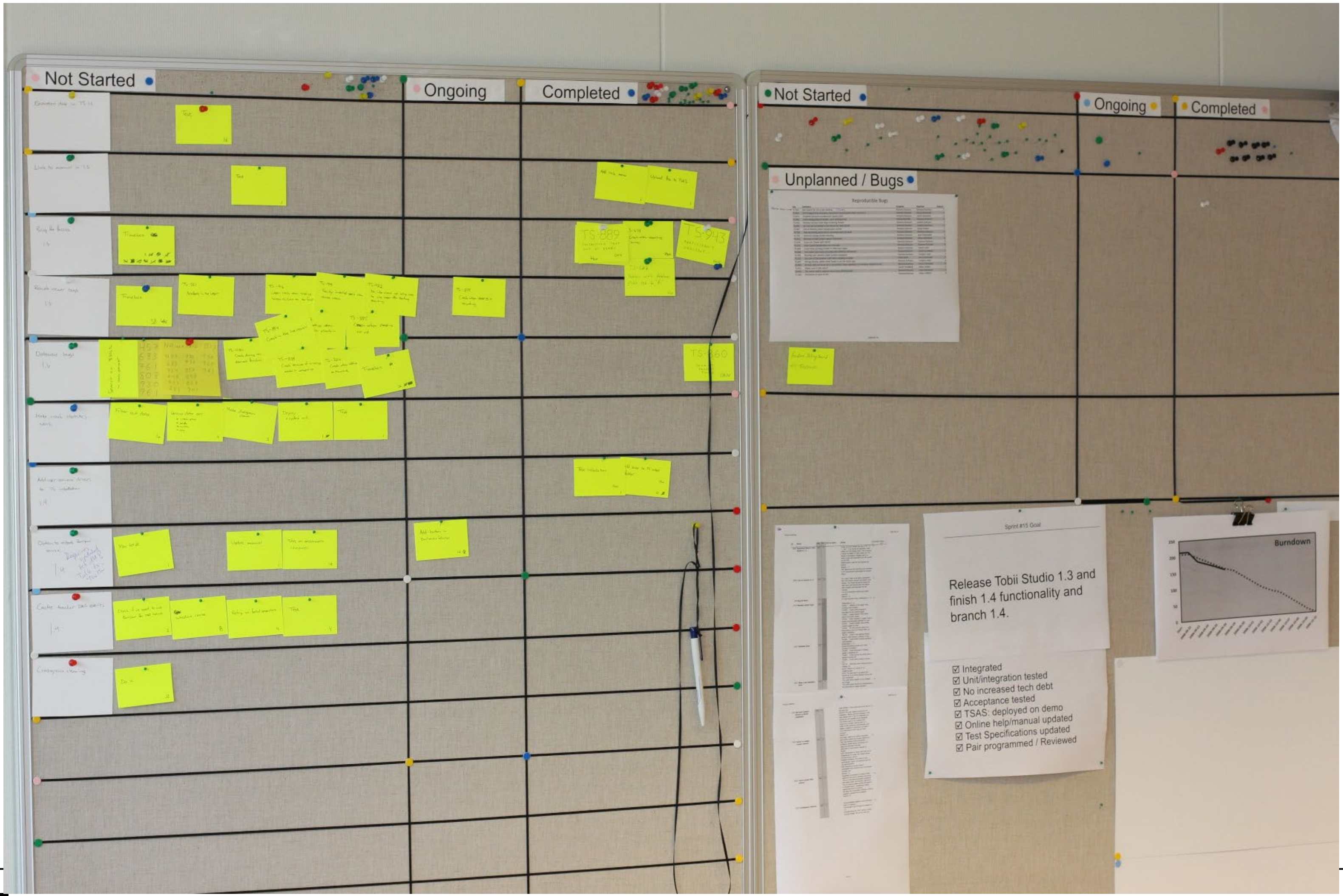


Sprint end date and deliverable do not change

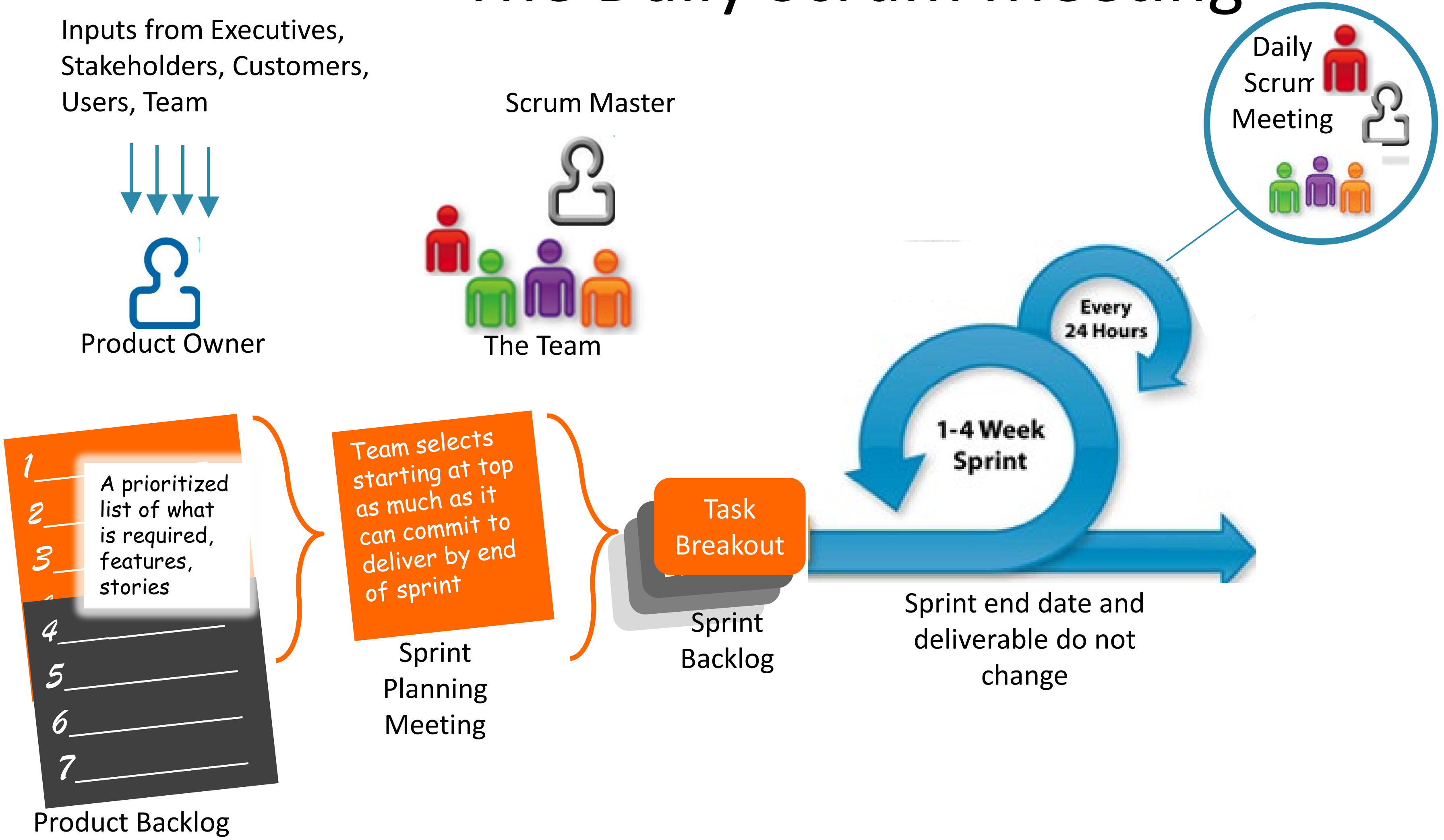
The Sprint Backlog


Inputs from Executives, Stakeholders, Customers, Users, Team
Scrum Master
Product Owner
The Team
1
2
3
4
5
6
7
A prioritized list of what is required, features, stories
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Every 24 Hours
1-4 Week Sprint
Sprint end date and deliverable do not change

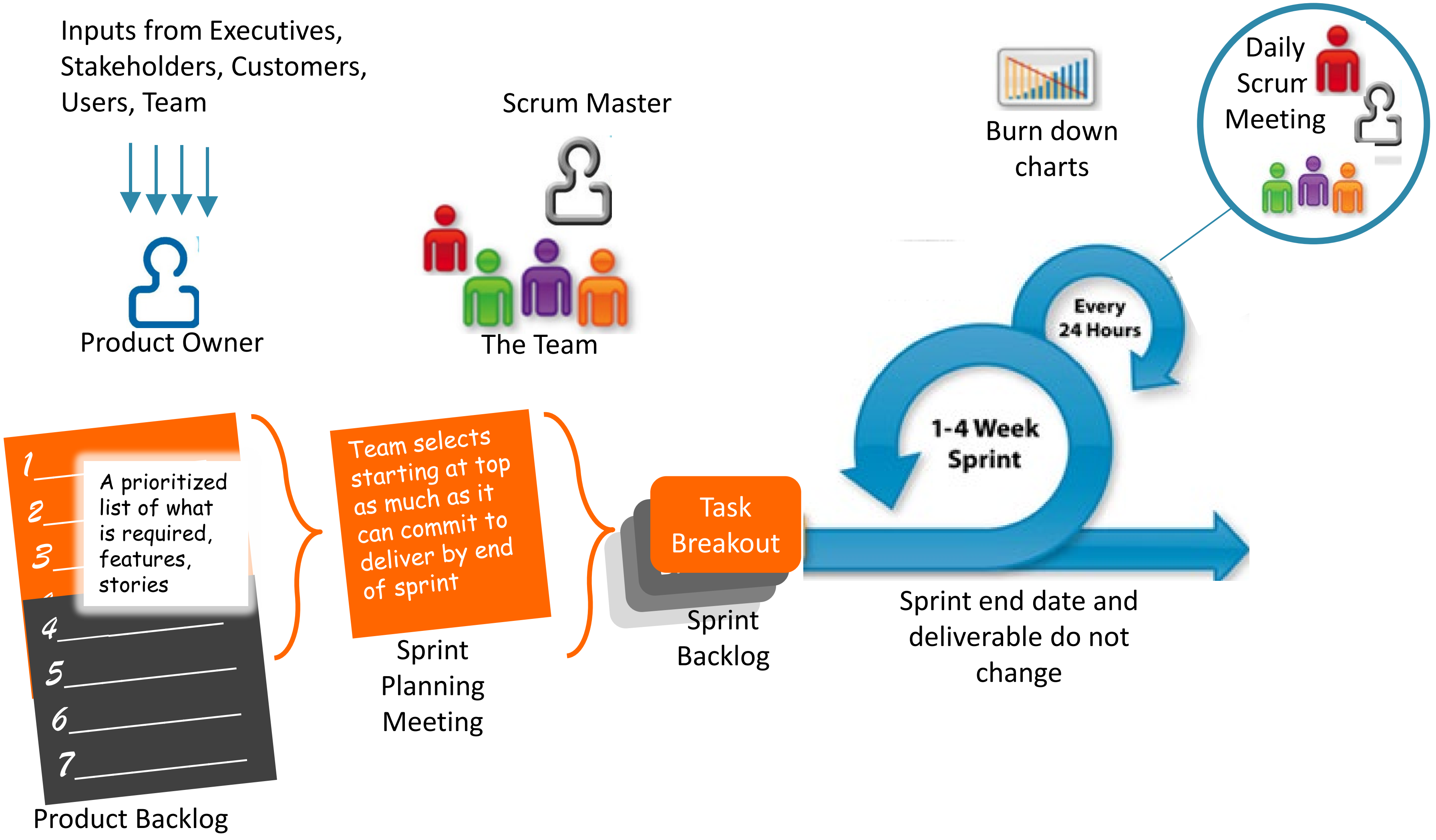
Sample Taskboard



The Daily Scrum Meeting

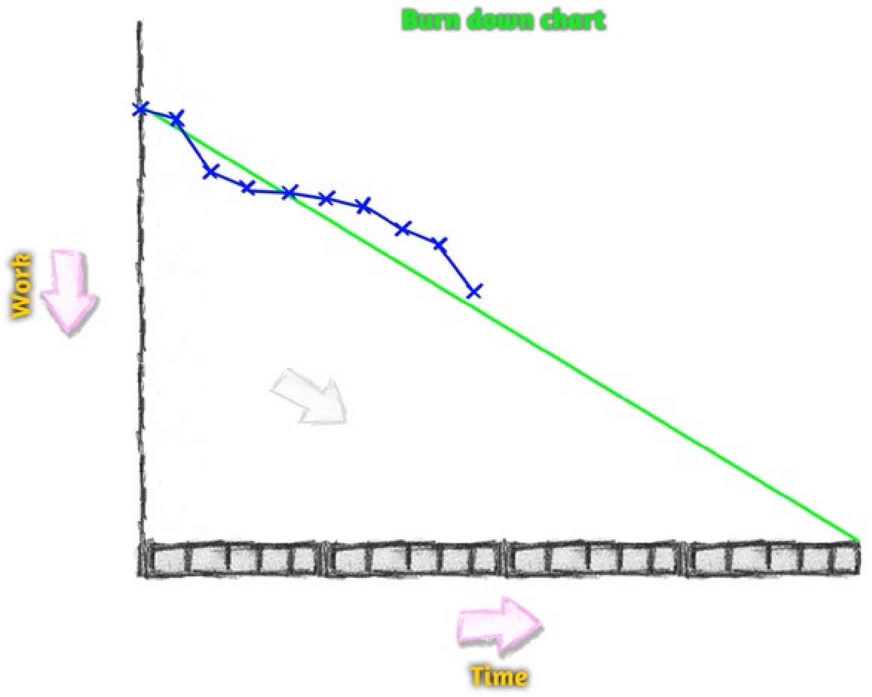

Inputs from Executives, Stakeholders, Customers, Users, Team
Product Owner
Scrum Master
The Team
Daily Scrum Meeting
1
2
3
A prioritized list of what is required, features, stories
4
5
6
7
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Every 24 Hours
1-4 Week Sprint
Sprint end date and deliverable do not change

The Burn Down Charts

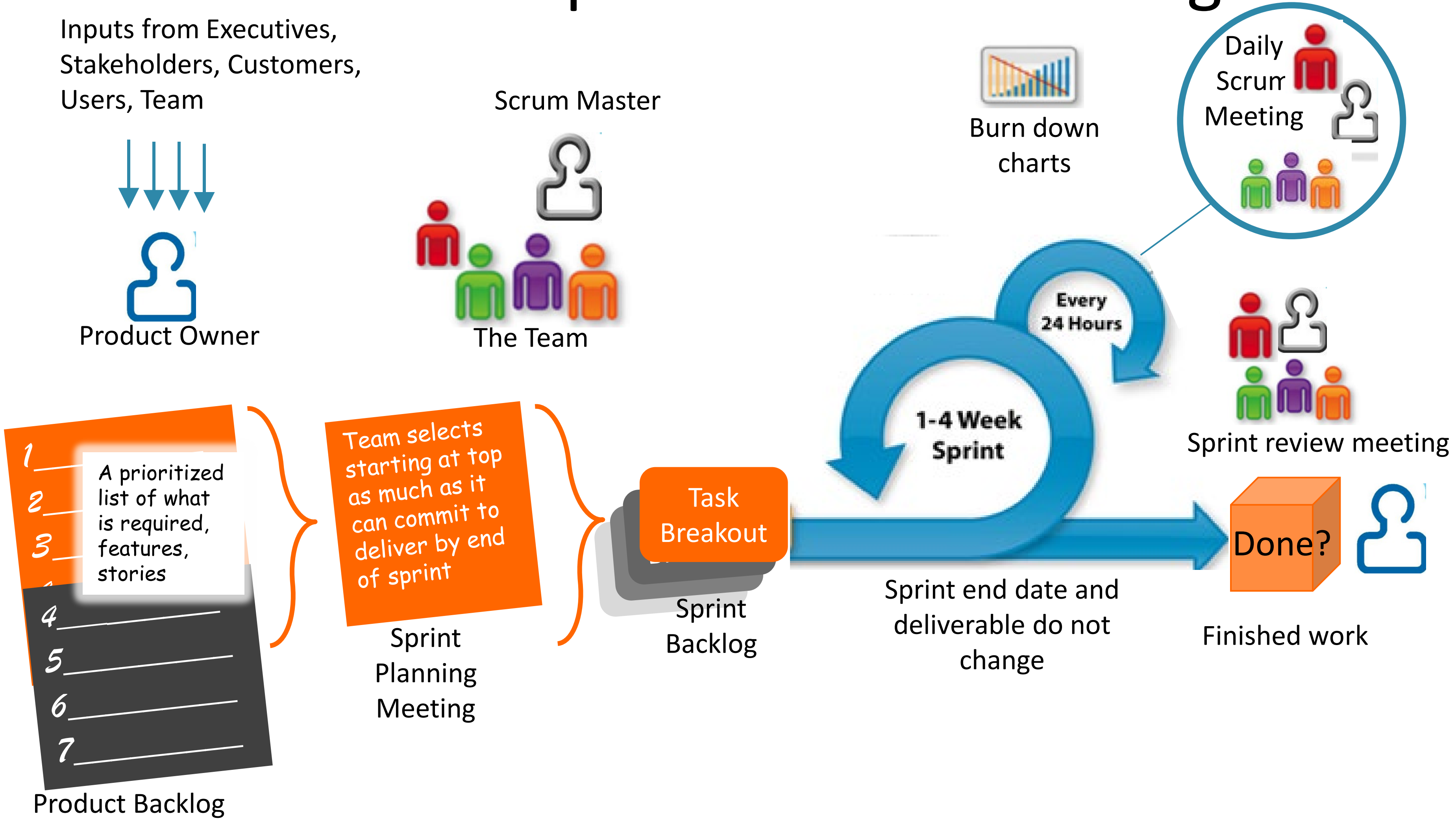

Inputs from Executives, Stakeholders, Customers, Users, Team
Product Owner
Scrum Master
The Team
Burn down charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
1
2
3
4
5
6
7
A prioritized list of what is required, features, stories
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and deliverable do not change

The burn down chart

- Only track hours remaining, not hours worked
- X – days (in Sprint)
- Y – hours remaining in **estimated** time or points
- Remove meeting time, vacation etc. from total available hours
- Update only when PBIs are DONE
- When not done – Undone PBIs



The Sprint Review Meeting


Inputs from Executives, Stakeholders, Customers, Users, Team
Product Owner
Scrum Master
The Team
Burn down charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint review meeting
1
2
3
4
5
6
7
A prioritized list of what is required, features, stories
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and deliverable do not change
Done?
Finished work

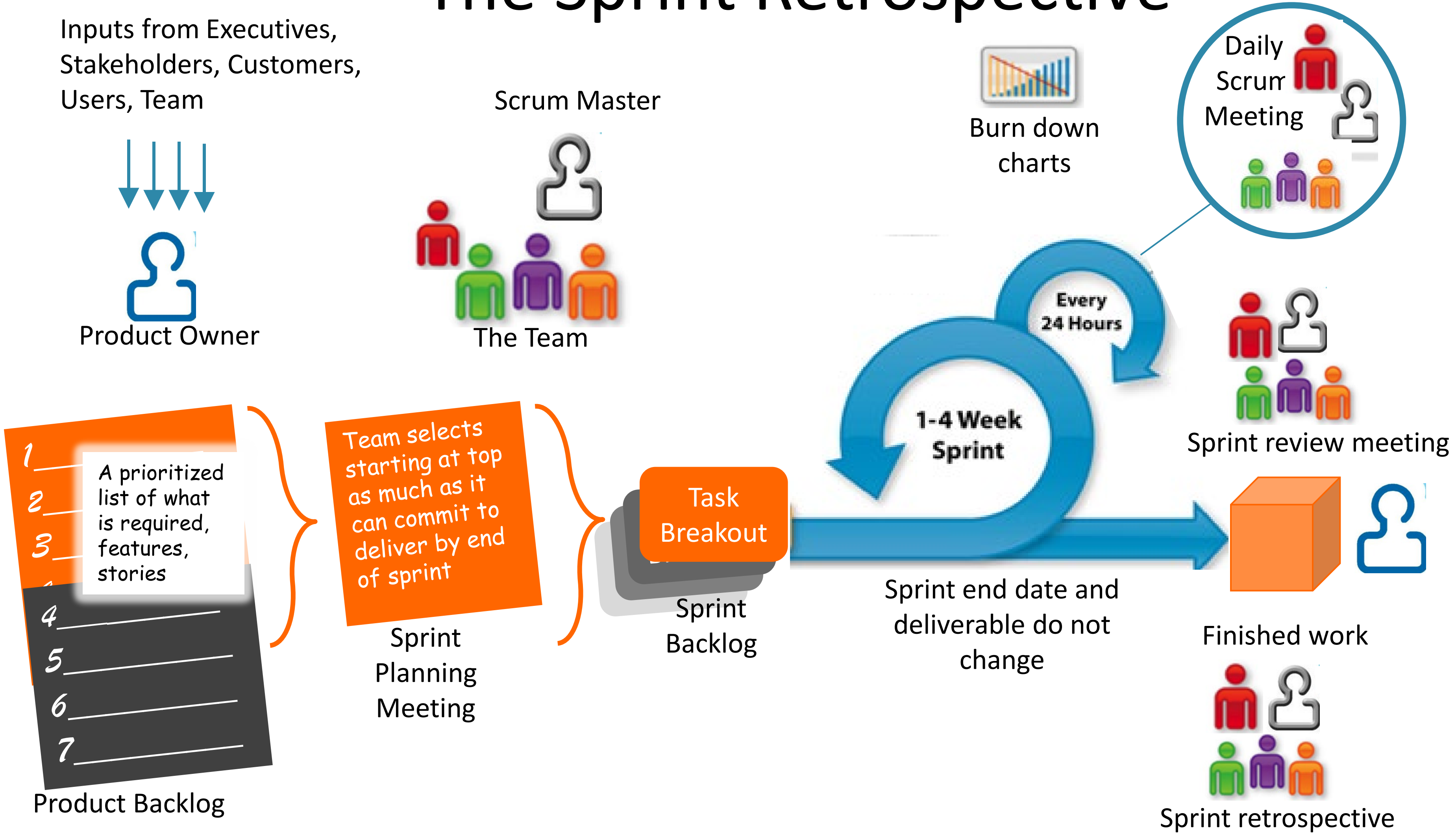
The Definition of Done!

- When are we done?

- “No more remaining work”
- Includes testing, documentation etc.
- Possible to ship after each sprint
- Everybody – understand what done means

Tools to support done
- Version handling (SCM)
- Automated build
- Automated tests (Continuous Delivery)

The Sprint Retrospective


Inputs from Executives, Stakeholders, Customers, Users, Team
Product Owner
Scrum Master
The Team
Burn down charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint review meeting
1
2
3
A prioritized list of what is required, features, stories
4
5
6
7
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and deliverable do not change
Finished work
Sprint retrospective

Agenda:

~~Definition of process~~

~~Life-cycle models~~

- ~~V and Waterfall~~
- ~~Incremental and Iterative~~

Method frameworks

- ~~OpenUP~~
- ~~Essence Kernel~~
- ~~eXtreme Programming~~
- ~~SCRUM~~
- KANBAN

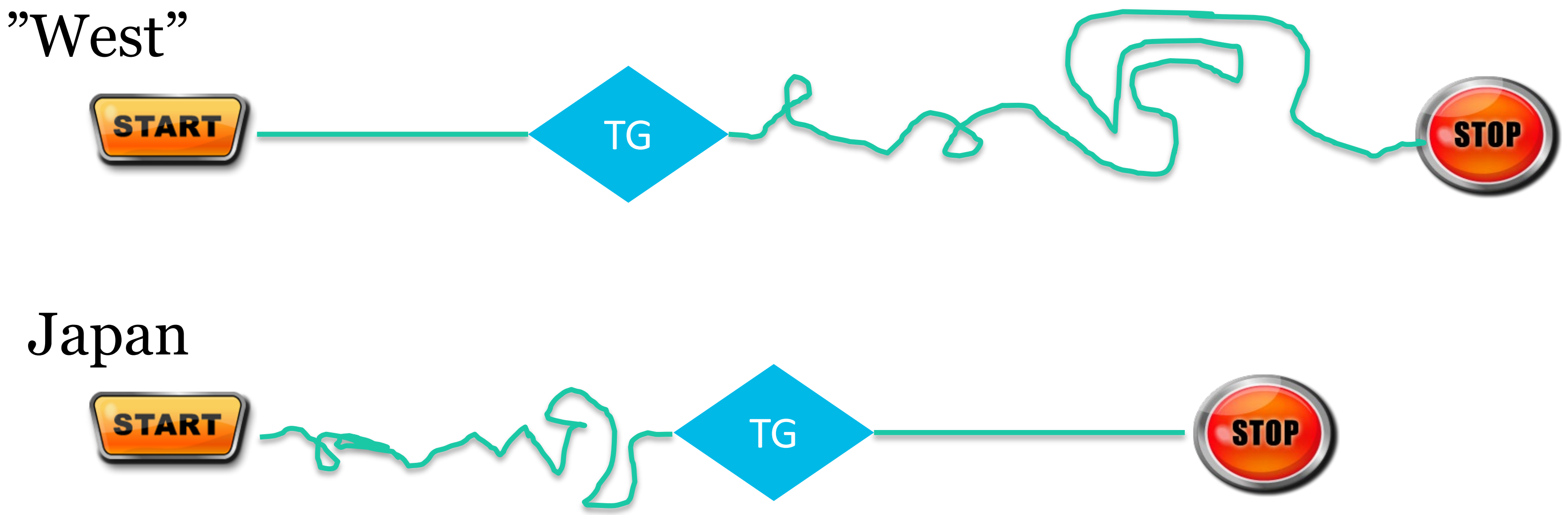
Which strategy do you prefer?

You have three books to read before the exam. Do you

1. Work ”little and often”, you study each course for three hours per day; or
2. ”Reduce multitasking”, read the first book, then the second, and then the third?

https://www.menti.com/m5eyksempa

Lean methods according to Masayuki Yamaguchi(*)



(*) Mary and Tom Poppendieck: Leading lean software development Addison-Wesley 2010

Lean principles

- Eliminate waste – don’t develop the wrong product
- Build quality in – automate tedious or error prone parts
- Create knowledge – continuous process improvement
- Defer commitment – wait until facts are known
- Deliver fast – limit queues
- Respect people – self-organized teams
- Optimize the whole – don’t just fix bugs, solve problems

(*) Mary and Tom Poppendieck:
Leading lean software development
Addison-Wesley 2010

Kanban

The two pillars of the Toyota production system are just-in-time production and automation with a human touch, or autonomation.
The tool used to operate the system is kanban.

Taiichi Ohno
Father of the Toyota Production System

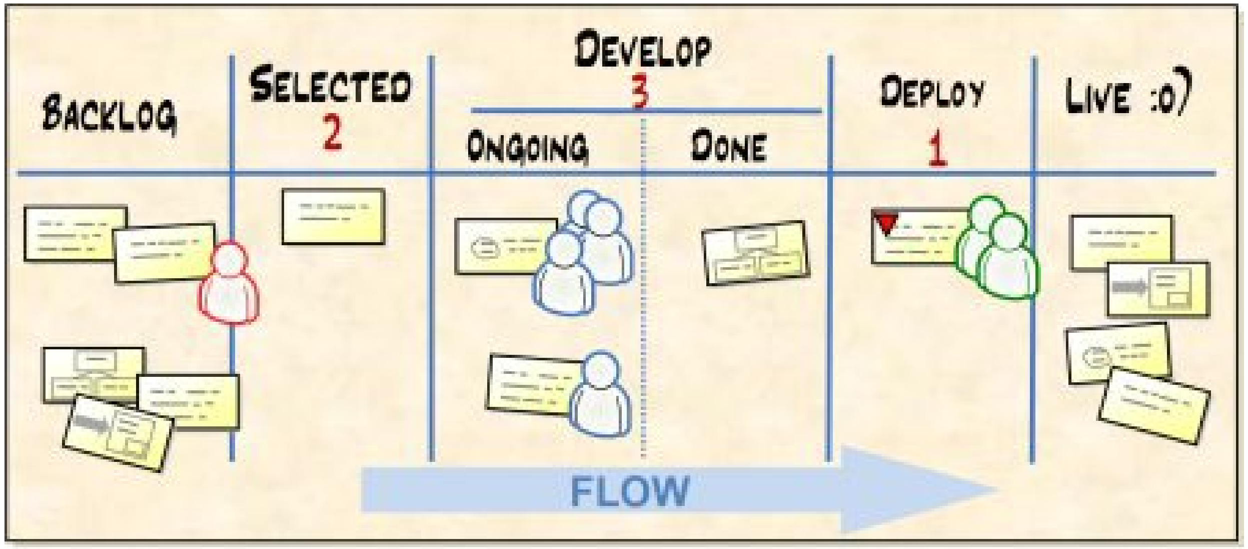
看板 - Kanban

- 看板 - Kanban is a Japanese word that means “visual card,” “signboard,” or “billboard.”
- Toyota originally used Kanban cards to limit the amount of inventory tied up in “work in progress” on a manufacturing floor
- Kanban is a **lean** approach to agile software development
- Focuses on the flow of progress

How does Kanban Work?

- **Visualize the workflow**
  - Split the work into pieces, write each item on a card and put on the wall.
  - Use named columns to illustrate where each item is in the workflow.
- **Limit WIP** (work in progress) – assign explicit limits to how many items may be in progress at each workflow state.
- **Measure the lead time** (average time to complete one item, sometimes called “cycle time”), optimize the process to make lead time as small and predictable as possible.

A simple Kanban Board



Source: http://www.crisp.se/gratis-material-och-guider/kanban
Good book: https://www.infoq.com/minibooks/kanban-scrum-minibook/

Work In Progress

Work In Progress, WIP, limits are designed to:

- reduce multitasking
- maximize throughput
- enhance teamwork

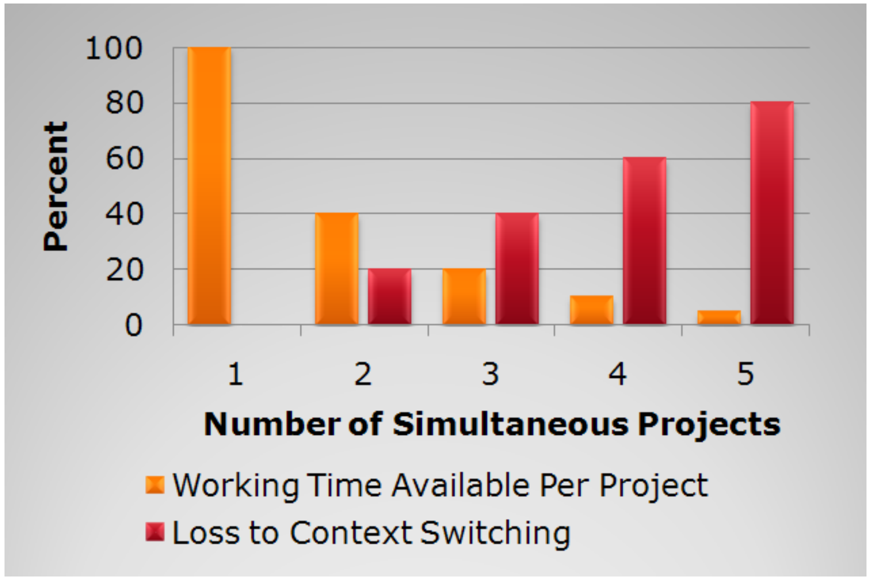
Reducing multitasking is beneficial for two primary reasons

Reducing Multitasking

1

20% time is lost to context switching per 'task', so fewer tasks means less time lost

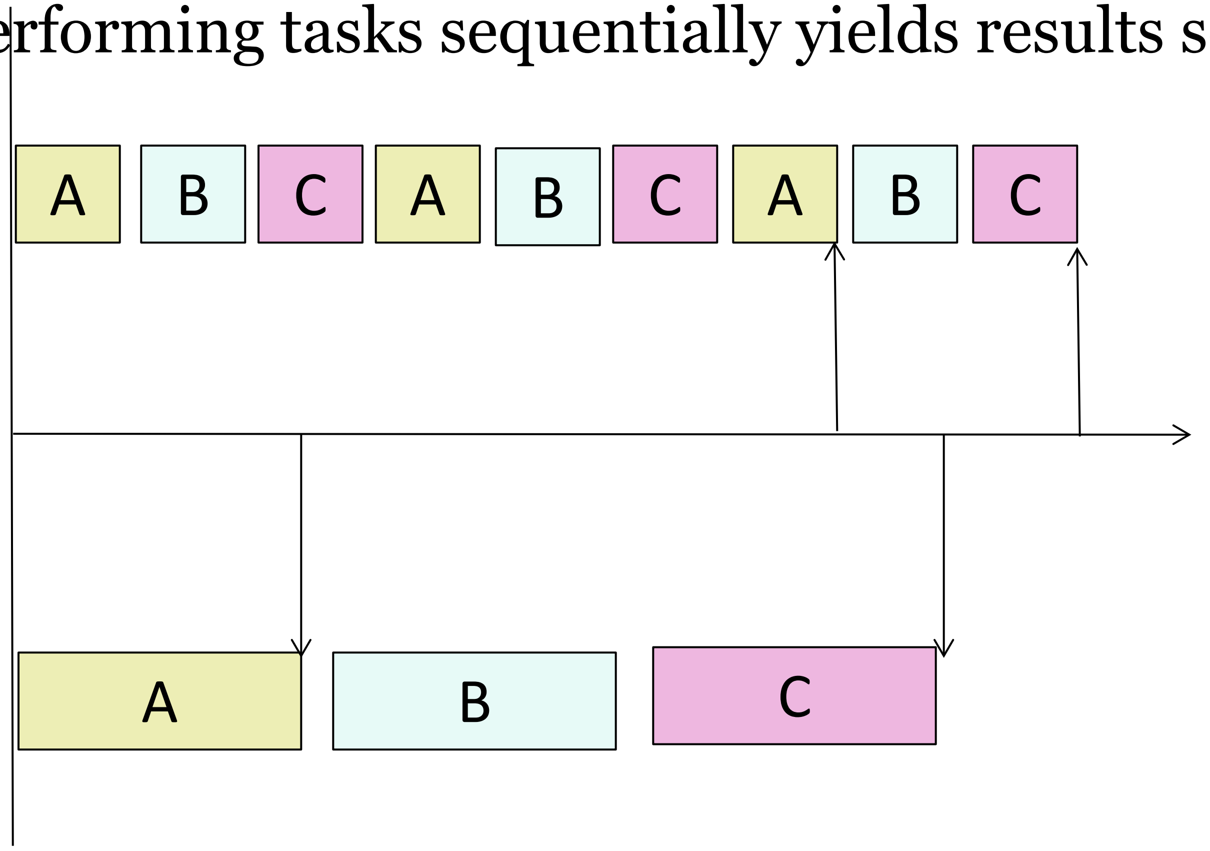
(from Gerald Weinberg, Quality Software Management: Systems Thinking)



2

Reducing Multitasking

Performing tasks sequentially yields results sooner.



multi-tasking A, B and C (on the top), delivers A much later, and even C slightly later, than sequentially (on the bottom).

Typical Measurements

- **Cycle time** – Measured from when you started working on it
- **Lead time** – Measured form when the customer ordered
- **Quality** – Time spent fixing bugs per iteration
- **WIP** – Average number of “stories” in progress
- **Throughput** – Number of “stories” completed per iteration (when using fixed iterations)

Benefits of Kanban

- Eliminate over-production, the #1 waste
- Produce only what is ordered, when ordered, & quantity ordered
- Increase flexibility to meet customer demand
- Competitive advantage by sequencing shipments to customers (what they want, when they want it, in the order they want it!)
- Several things are optional: sprints, estimation, agile practices. Even iterations!

Good things with agile

- With waterfall you only did specification
- It is easier to make changes
- Collaboration and cross-functional teaming
- Works well under uncertainties - focus on high-risk development,
- Small teams protected from the organization (stop management to interfere)
- Works in some contexts, small teams, engaged customer

(From group discussion in a seminar lead by Jan Bosch, Chalmers and Helena Holmström-Olsson, MU)

Bad things with agile

- Hard to connect with hardware development.
- It was a stressful transition from waterfall
- Missing on architecture perspective or larger picture
- Agile at scale hard to get working example:17 teams in one room, teams do their planning in beforehand anyway
- So many prescribed things, such as meetings
- Irrelevant measures, velocity, story points – inhumane
- Impact from team changes
- Reactive
- Knowledge development of co-workers

Beyond Agile

- Work architecture-driven
- Decouple teams from components and features
- Track the behavior of software in real-time
- Predict stakeholder needs
- Hypothesize and experiment

(Jan Bosch, Chalmers)

www.liu.se

LINKÖPING UNIVERSITY