Summary lecture

IN5140

Practical information

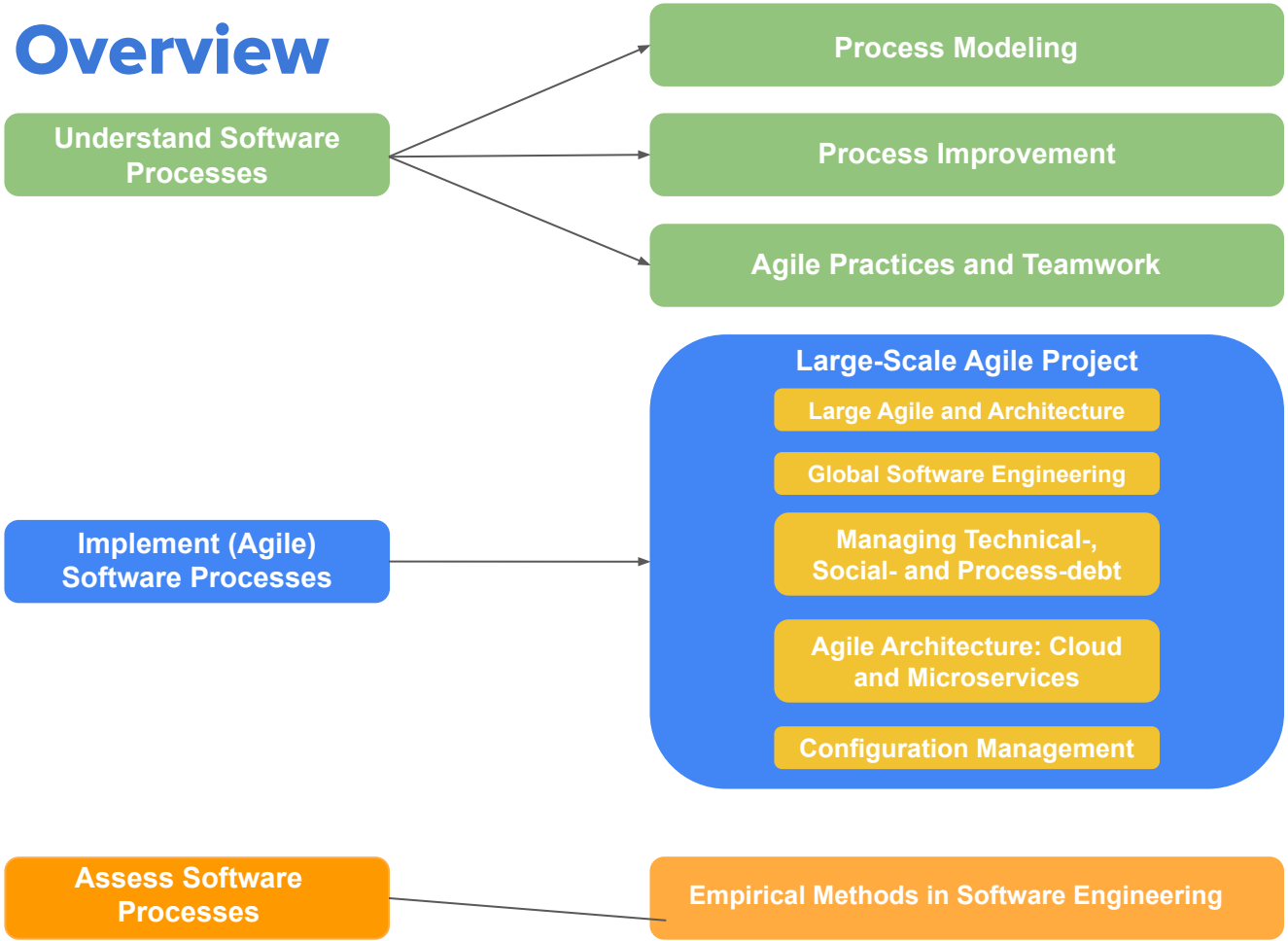
- When: November 28 at 9:00 AM
- Silurveien 2, 4 hours + 30 minutes
- No materials permitted
- Platform: Inspera

- All topics covered in the course is relevant for the exam (including guest lectures)

Questions asked

- Will BPMN models be in the exams?
  - Yes
- How much details do we need to know? Every icon and symbol there is or just the most important ones?
  - Only the most important ones. Lanes, actors, events, activities

Course Overview



How to answer on the exam?

The student must be able to show that they **understand** the curriculum and are able to **apply** it to different cases and **discuss** the results

In general:

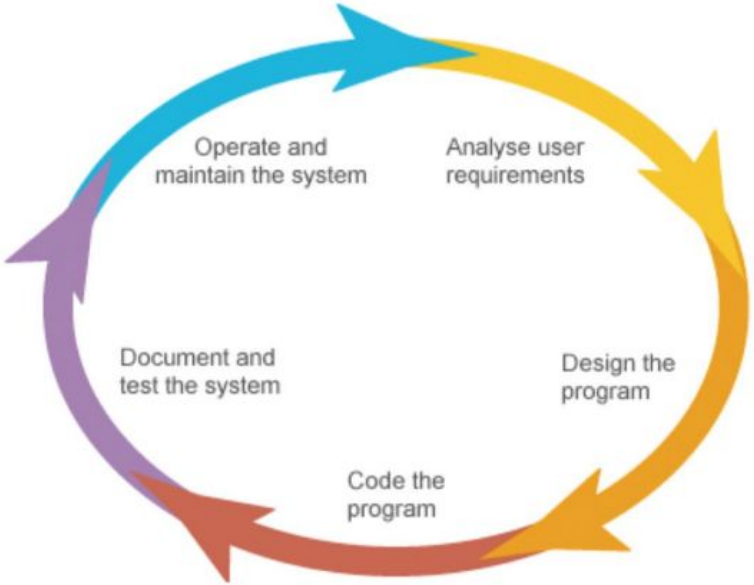
- Answers are mostly correct and concise, no irrelevant text
- Answers are well structured
- The student reasoned about the topics and gives rationale
- Content is related to the material in the lectures and slides

Summary

Slides and figures from group sessions and lectures

Software processes

Systematic approach to SPI
(Software Process Improvement)

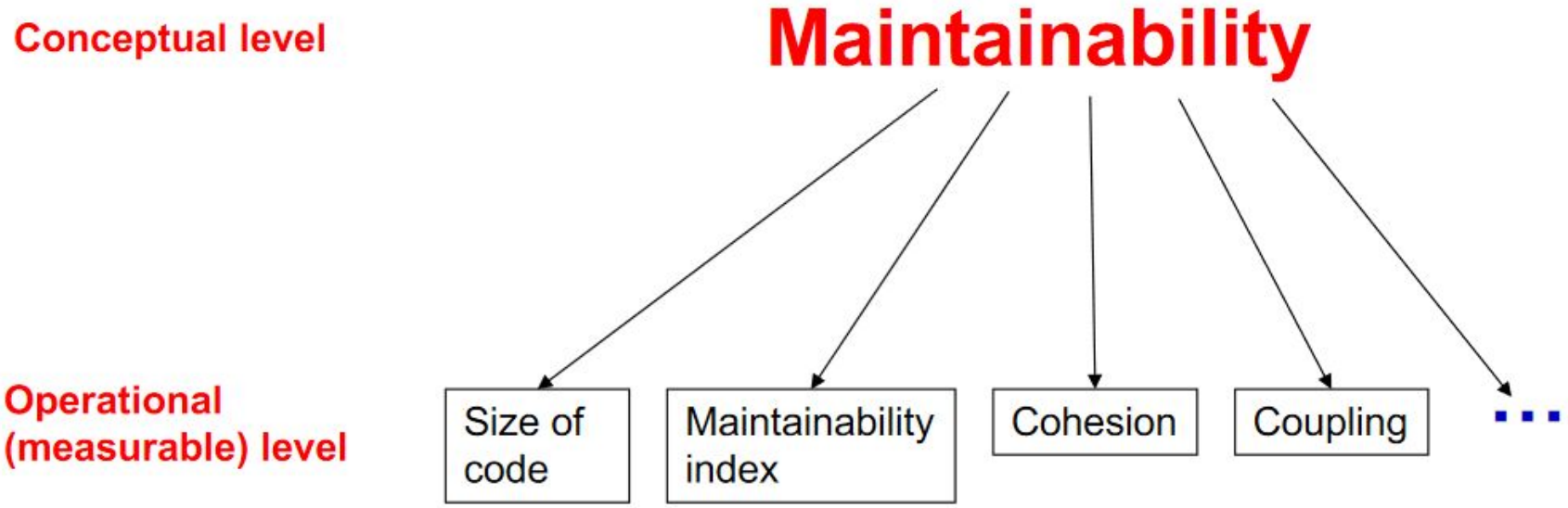


- **PLAN** what you want to accomplish over a period of time and what you might do, or need to do, to get there
- **DO** what you planned to do
- **CHECK** the results of what you did to see if the objective was achieved
- **ACT** on the information – standardize or plan for further improvement

Types of maintenance

- Corrective maintenance - Fixing bugs
- Preventive maintenance - Improving structure (re-engineering refactoring)
- Adaptive maintenance - Adapt to changing environments
- Perfective maintenance - New functionality

Maintainability



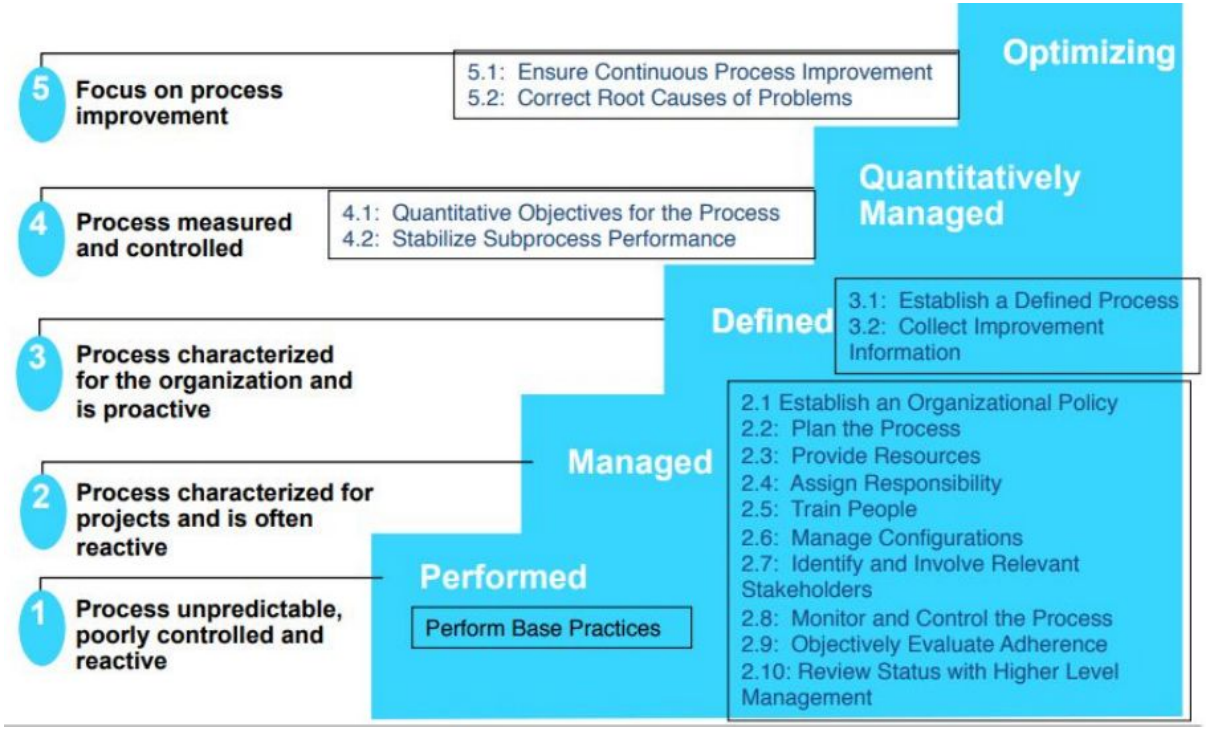
Formal and real process

- Formal process: An abstract representation of a process. The model describes the process from a certain perspective such as formal process is presenting what we say we do or **what we should do**. e.g. "Cooking recipe", "Delivery route".

- Real process: Activities that are carried out in a development project such as execution of processes that describes **what we actually do**.

Software maturity levels

CMMI (Capability Maturity Model Integrated)



BPMN

Activities

Task — A **Task** is a unit of work, the job to be performed. When marked with a [+] symbol it indicates a **Sub-Process**, an activity that can be refined.

Transaction — A **Transaction** is a set of activities that logically belong together; it might follow a specified transaction protocol.

Event Sub-Process — An **Event Sub-Process** is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can interrupt the higher level process context or run in parallel (non-interrupting) depending on the start event.

Call Activity — A **Call Activity** is a wrapper for a globally defined Task or Process reused in the current Process. A call to a Process is marked with a [+] symbol.

Activity Markers

Markers indicate execution behavior of activities:

- Sub-Process Marker
- Loop Marker
- Parallel MI Marker
- Sequential MI Marker
- Ad Hoc Marker
- Compensation Marker

Task Types

Types specify the nature of the action to be performed:

- Send Task
- Receive Task
- User Task
- Manual Task
- Business Rule Task
- Service Task
- Script Task

Sequence Flow — defines the execution order of activities.

Default Flow — is the default branch to be chosen if all other conditions evaluate to false.

Conditional Flow — has a condition assigned that defines whether or not the flow is used.

Gateways

Exclusive Gateway — When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

Event-based Gateway — Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

Parallel Gateway — When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.

Inclusive Gateway — When splitting, one or more branches are activated. All active incoming branches must complete before merging.

Exclusive Event-based Gateway (instantiate) — Each occurrence of a subsequent event starts a new process instance.

Complex Gateway — Complex merging and branching behavior that is not captured by other gateways.

Parallel Event-based Gateway (instantiate) — The occurrence of all subsequent events starts a new process instance.

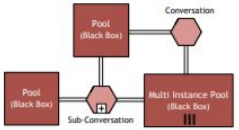
Conversations

A **Conversation** defines a set of logically related message exchanges. When marked with a [+] symbol it indicates a **Sub-Conversation**, a compound conversation element.

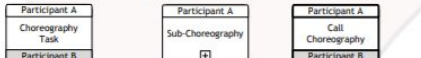
A **Call Conversation** is a wrapper for a globally defined Conversation or Sub-Conversation. A call to a Sub-conversation is marked with a [+] symbol.

A **Conversation Link** connects Conversations and Participants.

Conversation Diagram



Choreographies



A **Choreography Task** represents an Interaction (Message Exchange) between two Participants.

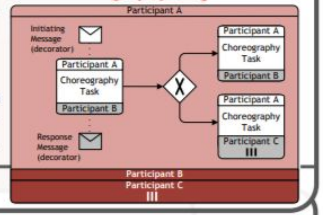
A **Sub-Choreography** contains a refined choreography with several Interactions.

A **Call Choreography** is a wrapper for a globally defined Choreography Task or Sub-Choreography. A call to a Sub-Choreography is marked with a [+] symbol.

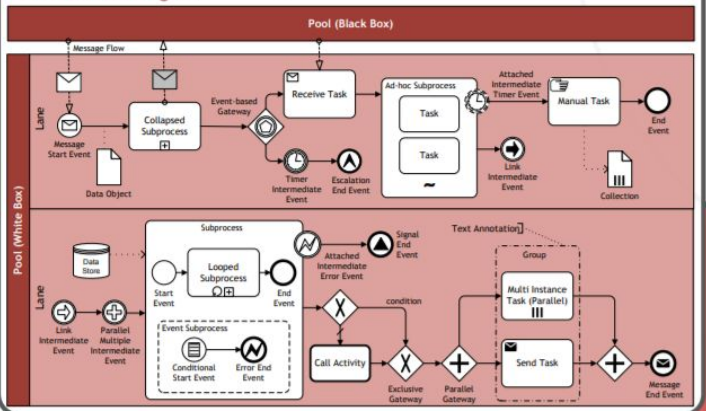
Multiple Participants Marker — denotes a set of Participants of the same kind.

Message — a decorator depicting the content of the message. It can only be attached to Choreography Tasks.

Choreography Diagram



Collaboration Diagram



Events

	Start: Standard	Start: Event Sub-Process Interrupting	Start: Event Sub-Process Non-Interrupting	Intermediate: Catching	Intermediate: Boundary Interrupting	Intermediate: Boundary Non-Interrupting	Intermediate: Throwing	End: Standard
None: Untyped events, indicate start point, state changes or final states.	○						○	○
Message: Receiving and sending messages.	✉	✉	✉	✉	✉	✉	✉	✉
Timer: Cyclic timer events, points in time, time spans or timeouts.	🕒	🕒	🕒	🕒	🕒	🕒		
Escalation: Escalating to an higher level of responsibility.		▲	▲		▲	▲	▲	▲
Conditional: Reacting to changed business conditions or integrating business rules.	≡	≡	≡	≡	≡	≡		
Link: Off-page connectors. Two corresponding link events equal a sequence flow.				⇨			⇨	
Error: Catching or throwing named errors.		⚡			⚡			⚡
Cancel: Reacting to cancelled transactions or triggering cancellation.					⊗			⊗
Compensation: Handling or triggering compensation.		⏪			⏪		⏪	⏪
Signal: Signalling across different processes. A signal thrown can be caught multiple times.	△	△	△	△	△	△	▲	▲
Multiple: Catching one out of a set of events. Throwing all events defined	⬠	⬠	⬠	⬠	⬠	⬠	⬟	⬟
Parallel Multiple: Catching all out of a set of parallel events.	✚	✚	✚	✚	✚	✚		
Terminate: Triggering the immediate termination of a process.								●

Data

A **Data Object** represents information flowing through the process, such as business documents, e-mails, or letters.

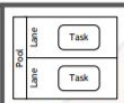
A **Collection Data Object** represents a collection of information, e.g., a list of order items.

Input — A **Data Input** is an external input for the entire process.A kind of input parameter.

Output — A **Data Output** is data result of the entire process. A kind of output parameter.

A **Data Association** is used to associate data elements to Activities, Processes and Global Tasks.

Data Store — A **Data Store** is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.



Pools (Participants) and **Lanes** represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.

Swimlanes

Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events. The Message Flow can be decorated with an envelope depicting the content of the message.

The order of message exchanges can be specified by combining message flow and sequence flow.

BPM Offensive Berlin

Hasso Plattner Institut

camunda

inubit

SIGNAVIO

Pools and Swimlanes

- **A pool** represents major participants in a process. A different pool may be in a different company or department but still involved in the process.

- **Swimlanes** within a pool show the activities and flow for a certain role or participant, defining who is accountable for what parts of the process.

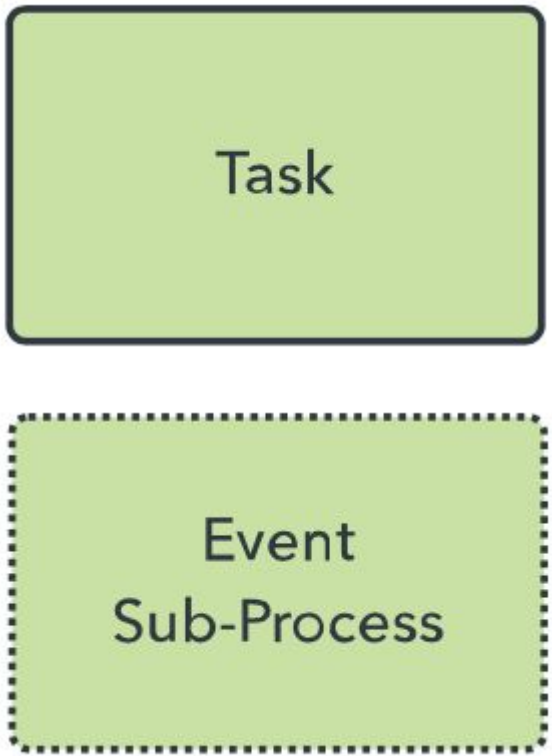
(Taken from Lucidchart)

Events

- A trigger that starts, modifies or completes a process.
- They are shown by circles containing other symbols based on event type.

Activities

- A particular activity or task performed by a person or system. It’s shown by a rectangle with rounded corners.
- They can become more detailed with for example sub-processes.



Gateways

- Decision point that can adjust the path based on conditions or events. They are shown as diamonds.
- Gateways can also be an empty diamond

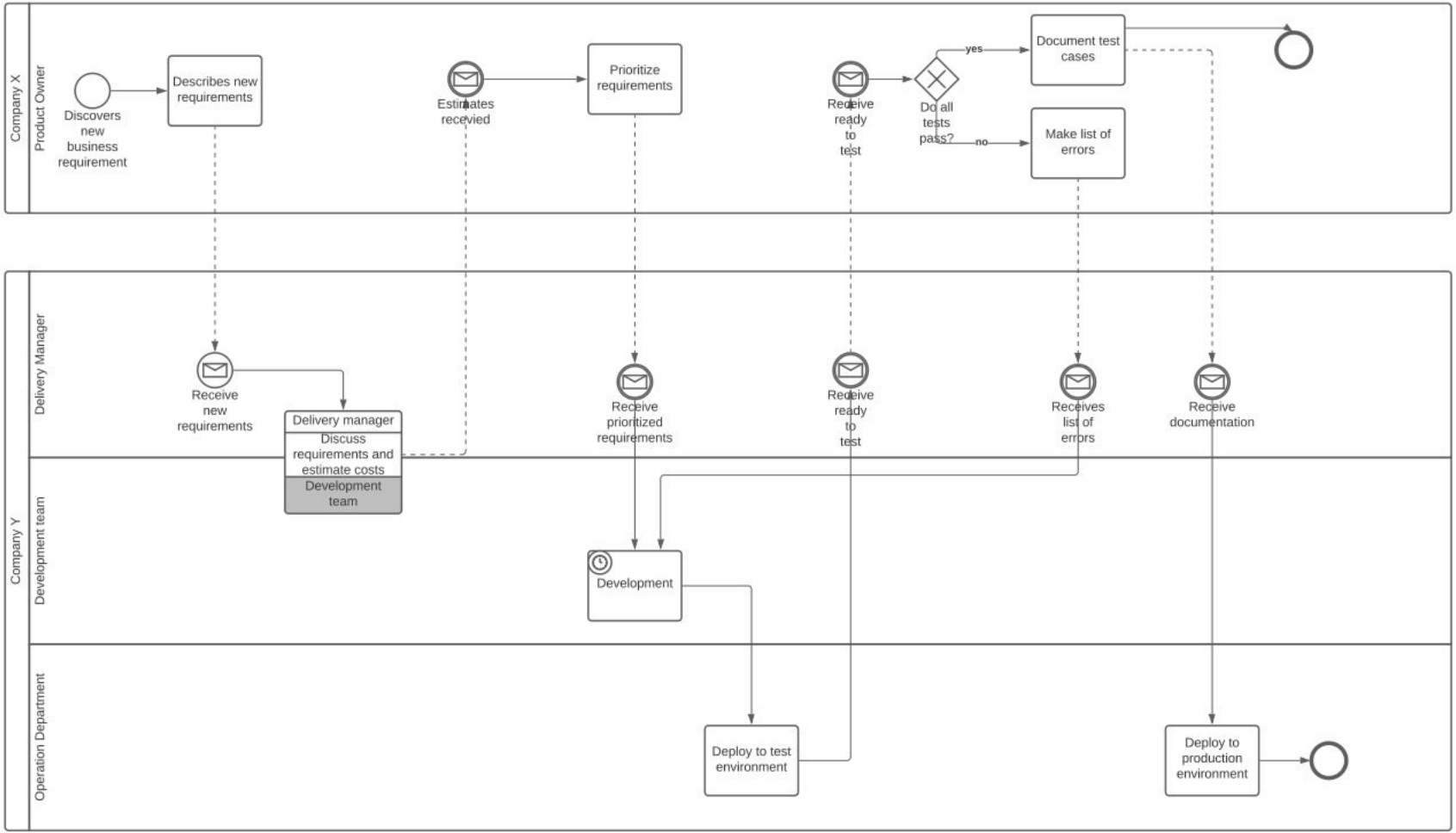
Flow

- **Sequence flow:** shows the order of activities to be performed

- **Message flow:** depicts messages that flow across “pools,” or organization boundaries such as departments. It shouldn’t connect events or activities within a pool.

How to engage BPMN

- **Skim through** the textual description
  - Try not to drown in the text
- Note potential **pools**
- Identify the **roles** of each pool
  - Swim lanes
  - What are their tasks
  - What are their dependencies
- Model the **tasks** with metadata
- Check that you are still answering the exercise
- Do not make the BPMN diagram too extensive and complicated


Company X
Product Owner
Discovers new business requirement
Describes new requirements
Estimates recevied
Prioritize requirements
Receive ready to test
Do all tests pass?
yes
no
Document test cases
Make list of errors
Company Y
Delivery Manager
Receive new requirements
Delivery manager
Discuss requirements and estimate costs
Development team
Receive prioritized requirements
Receive ready to test
Receives list of errors
Receive documentation
Development team
Development
Operation Department
Deploy to test environment
Deploy to production environment

Empirical methods and measurements

Directly or indirectly measurement

- Directly: Measuring that exact variable.

- Indirectly: Measuring the variable through measuring something else.

Objective or subjective

- Objective:
  - Based on facts rather than feelings, opinions, prejudices or interpretations [Merriam-Webster].

- Subjective:
  - Related to the way people experience things in their own mind
  - Based on feelings or opinions rather than facts, modified or affected by personal views, experience or background [Merriam-Webster]

Quantitative or qualitative

- Quantitative:
  - Data expresses quantity
  - Data expressed as numbers
  - Used in statistics

- Qualitative:
  - Data expresses quality in some sense
  - Data expressed as text, images, audio or video but not numbers
  - Not used in statistics

Nominal, ordinal, interval or ratio scale measurement

- Nominal: Divides the set of objects into categories, with no particular ordering among them. E.g. labeling, classification, defect type.
- Ordinal: Divides the set of entities into categories that are ordered. E.g. ranking, difficulty, failure severity, complexity.
- Interval: Comparing the differences between values is meaningful. E.g. temperature, start and end date of activities.
- Ratio scale: There is a meaningful “zero” value, and ratios between values are meaningful. E.g. Length, weight, lines of code, number of errors.

Lean

Lean principles (some of them)

- Satisfying the customer
- Flow
- Visualization
- Avoiding waste
- Supporting changes

Satisfying the customer

- Create value for the customer (what the customer is willing to pay for)
- Produce what the customer **actually** want
- Produce high quality and valuable software

Flow

- How items or people in a process move from the first step to the last
- Move as quick as possible, but without any risk to quality and customer satisfaction
- Optimized and planned, stable and with minimal waste

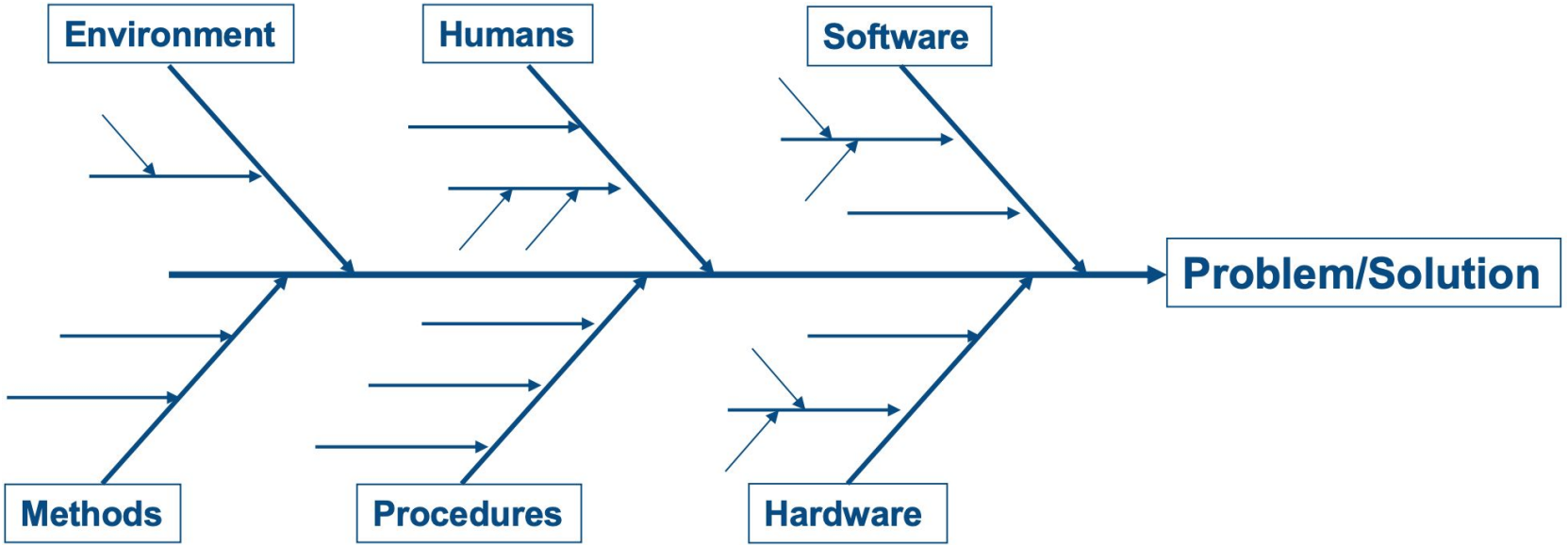
Visualization

- Gives an overview which is helpful for:
  - common understanding for all stakeholders
  - planning
- Examples: Obeya room, Kanban board

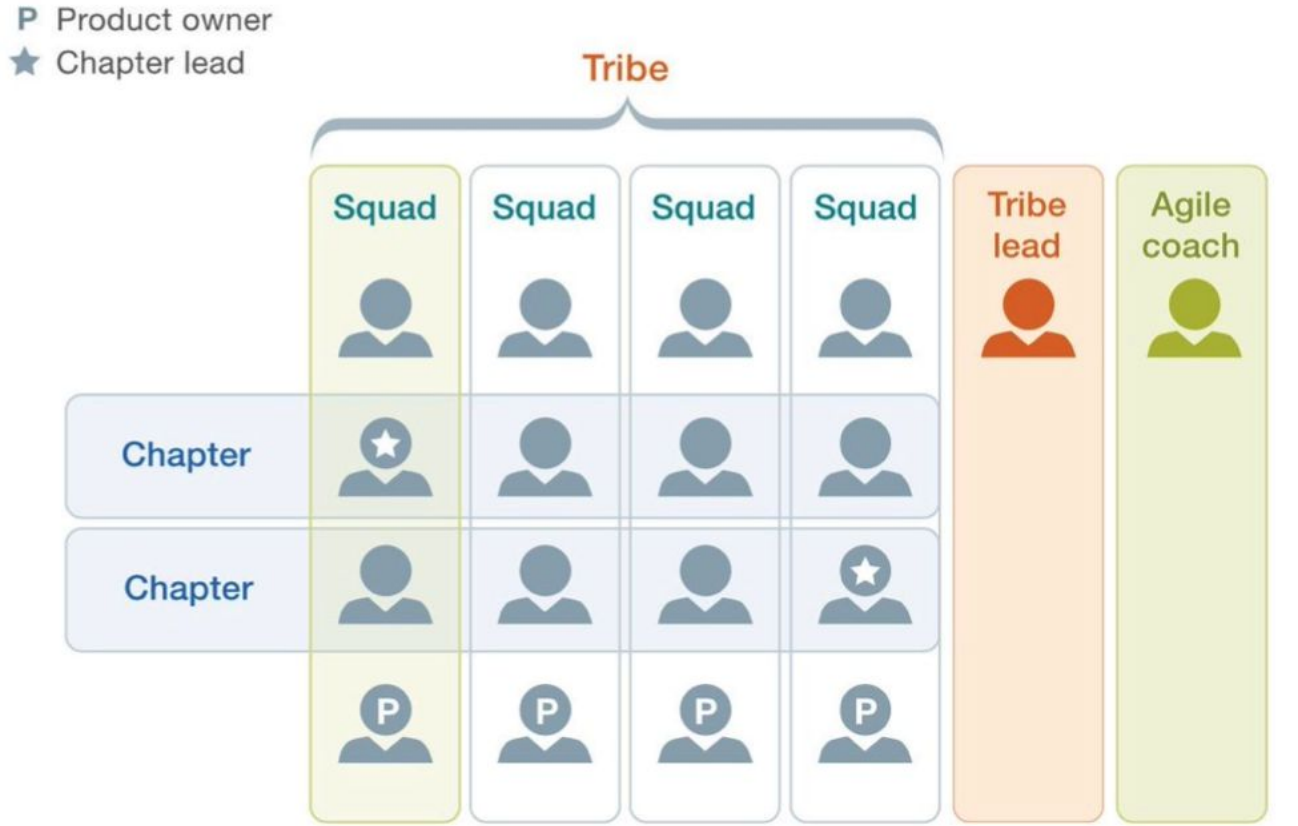
Waste

- Everything that requires resources and does not give value to the customer (time, work effort, space, money, equipment)
- Waste increase costs, not value
- Minimal waste ensures flow
- Unnecessary features, excessive documentation, partly done work, duplication of data, etc.

Fishbone diagram



Large-Scale Agile

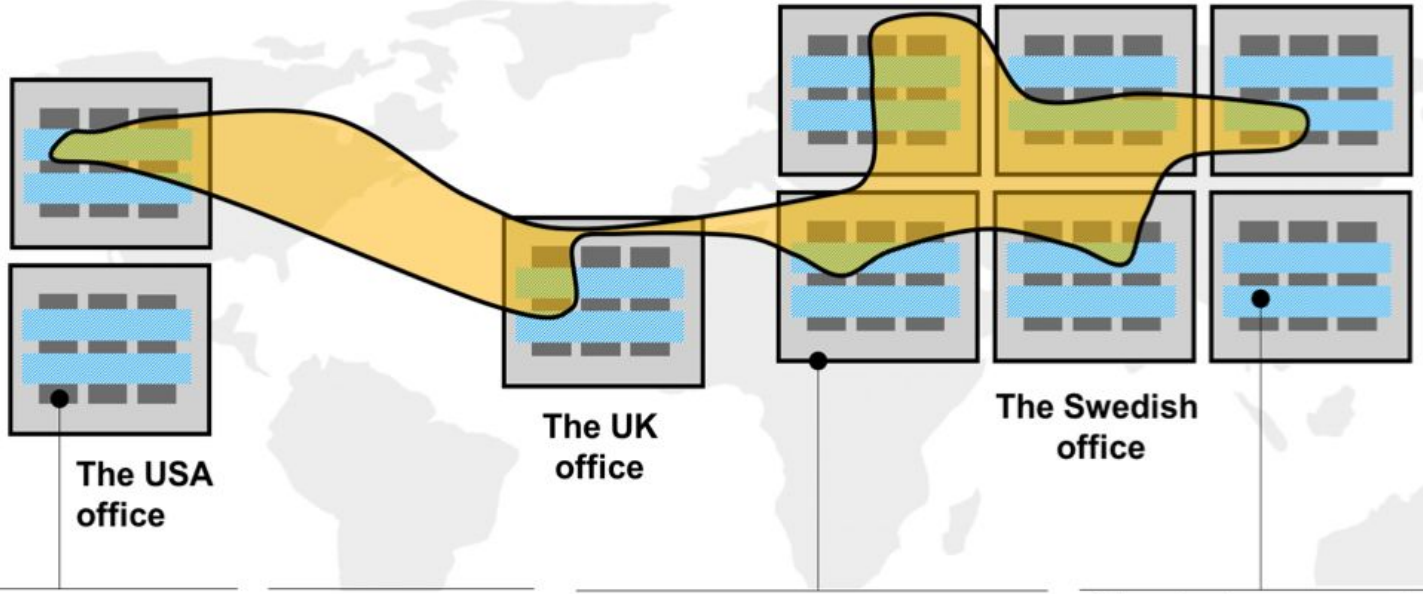


Tribe
(collection of squads with interconnected missions)

Squad
(basis of new agile organization)

Chapter
(develops expertise and knowledge across squads)

Basic organizational structures: Squads Chapters Tribes Guilds



Teams at Spotify are called **squads**, which should "feel like mini-startups", be self-organized and cross-functional, and ideally consist of 5-7 people

A **guild** is a group of people with similar skills and interests that share knowledge, tools or code across Spotify.

All squads are organized into **tribes** containing 30-200 people each. Tribes have a clear mission, set of principles, a senior experienced leader, and all skills needed to engineer working software features end-to-end.

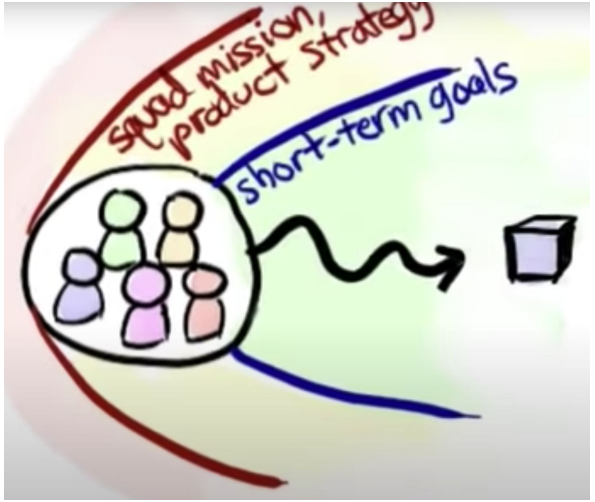
Chapter is a group of engineers who have the same manager (Chapter Lead) and is focused on personal growth and skills development. Engineers in chapters share knowledge, learn from each other, and discuss common challenges.

F

The Spotify Model

- Spotify Engineering Culture
- Key driving factor: Autonomous squad (cross functional, end to end responsibility)
    - Loosely coupled, tightly aligned squads
- Knowledge sharing

- Agile > Scrum
- Principles > Practises
- Cross-pollination > standardization
- Community > Structure



Main challenges

Main challenges/issues when scaling up Agile?

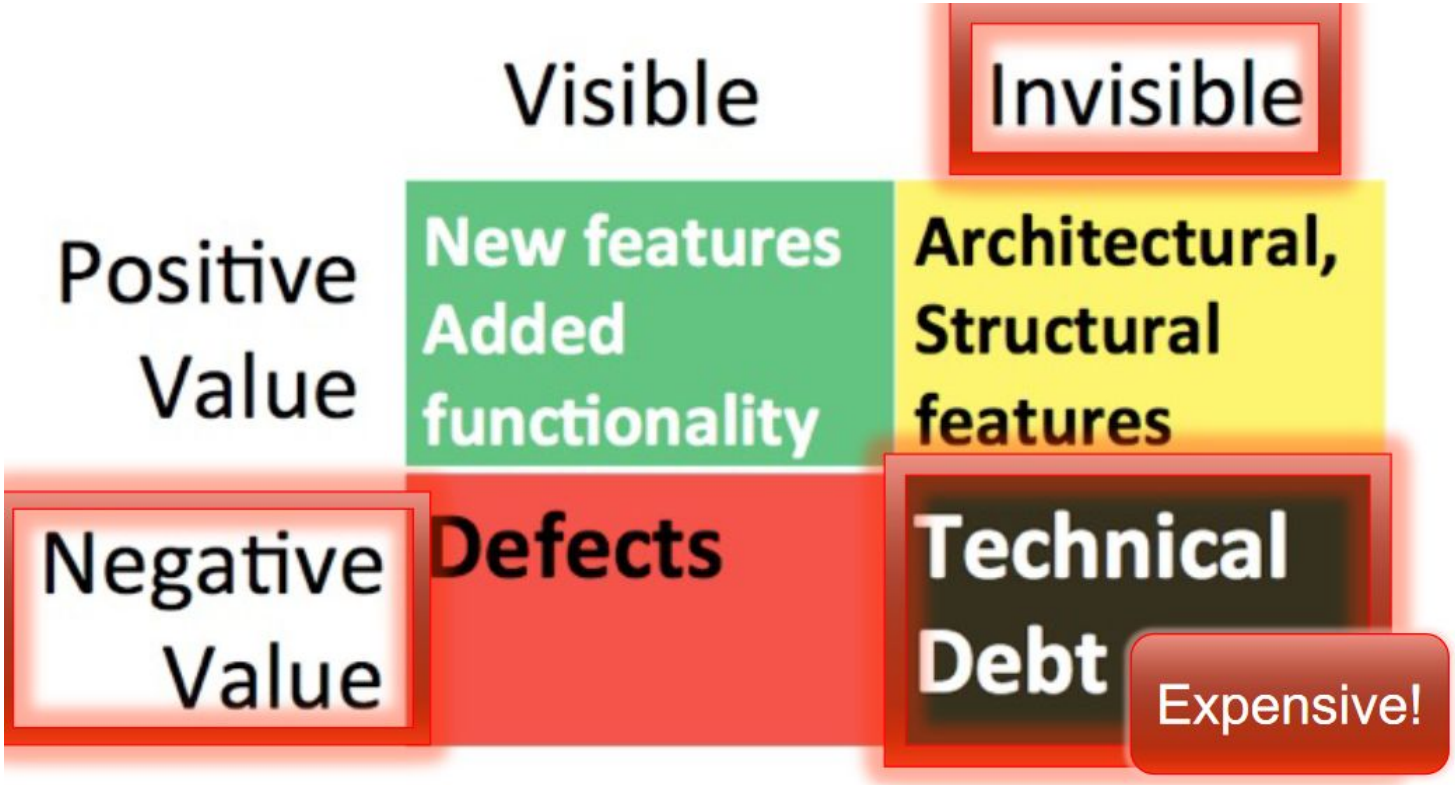
Challenge type
Change resistance 16 (38%)
General resistance to change
Skepticism towards the new way of working
Top down mandate creates resistance
Management unwilling to change
Lack of investment 13 (31%)
Lack of coaching
Lack of training
Too high workload
Old commitments kept
Challenges in rearranging physical spaces
Agile difficult to implement 20 (48%)
Misunderstanding agile concepts
Lack of guidance from literature
Agile customized poorly
Reverting to the old way of working
Excessive enthusiasm
Coordination challenges in multi-team environment 13 (31%)
Interfacing between teams difficult
Autonomous team model challenging
Global distribution challenges
Achieving technical consistency
Different approaches emerge in a multi-team environment 9 (21
Interpretation of agile differs between teams
Using old and new approaches side by side
Hierarchical management and organizational boundaries 14 (33%
Middle managers' role in agile unclear
Management in waterfall mode
Keeping the old bureaucracy
Internal silos kept
Requirements engineering challenges 16 (38%)
High-level requirements management largely missing in agile
Requirement refinement challenging
Creating and estimating user stories hard
Gap between long and short term planning
Quality assurance challenges 6 (14%)
Accommodating non-functional testing
Lack of automated testing
Requirements ambiguity affects QA
Integrating non-development functions 18 (43%)
Other functions unwilling to change
Challenges in adjusting to incremental delivery pace
Challenges in adjusting product launch activities
Rewarding model not teamwork centric

INTEF

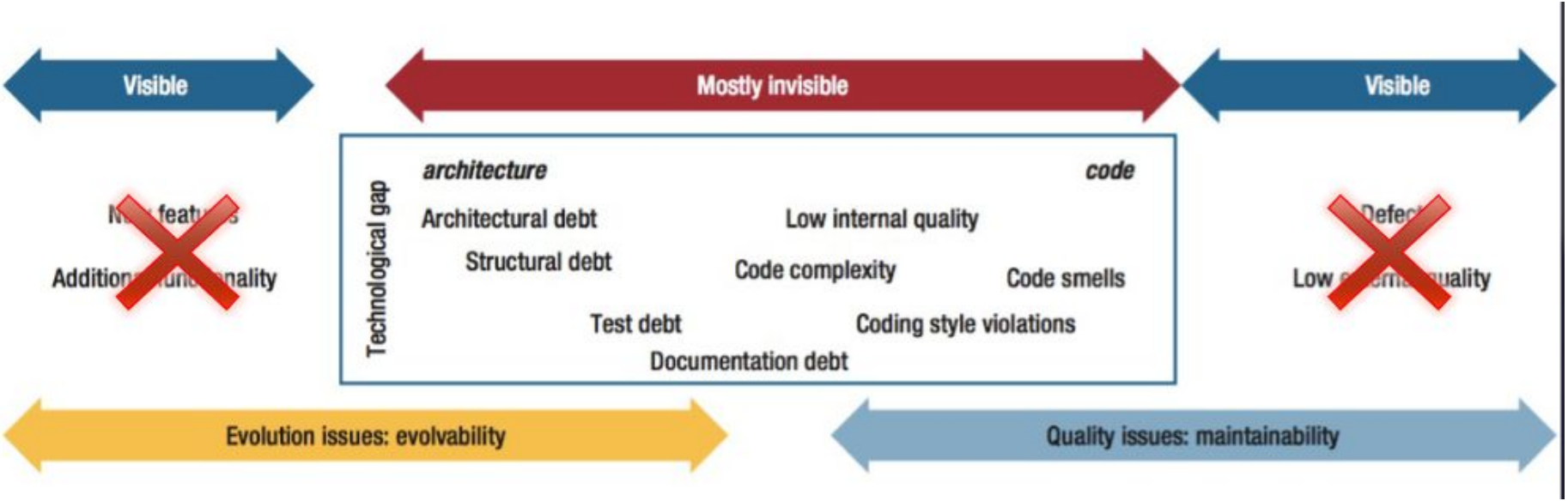
(Dikert et al., 2016)

Technical-, social- and process debt

Technical debt

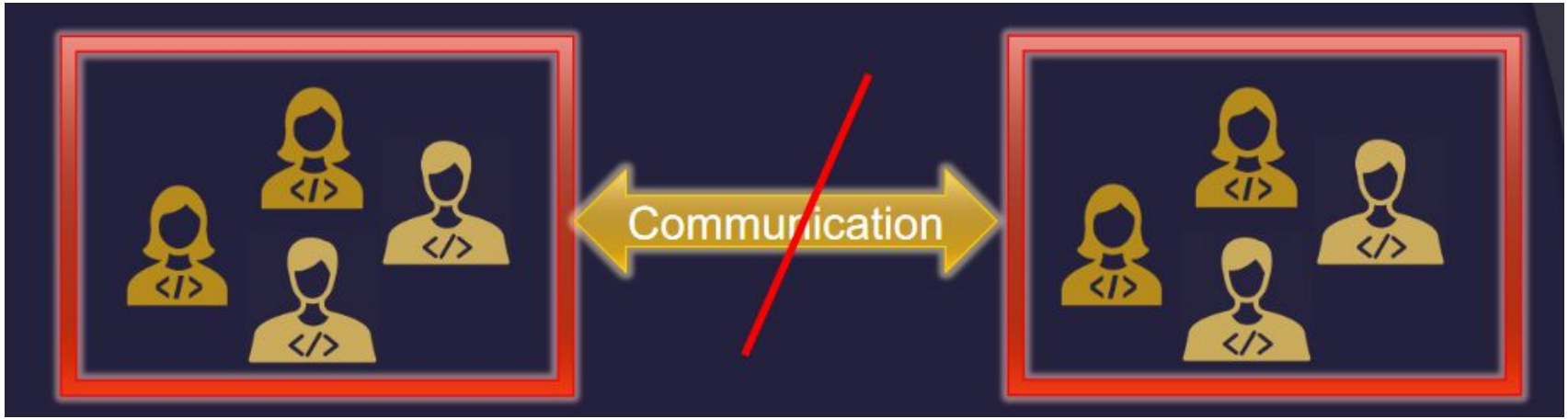


Technical debt



Social debt

- “The presence of sub-optimality in the development community, which causes a negative effect”



Process debt

- Sub-optimal process design
- Divergence from optimal formal process
- Deficiencies in the infrastructure that might be beneficial in the short term
- Negative long term effects

How do we fix technical debt?

Proactive approaches:

- Education
- Culture
- Organization
- Process
- Guidelines
- Visualization

Continuous approaches:

- Semi-automatic identifications
- Code reviews
- Retrospectives
- Technical leadership
- Dedicated refactoring sprints

Reactive approaches:

- Impact map
- Roadmap evaluation
- Resources to remove TD
- TD information used in planning and budget

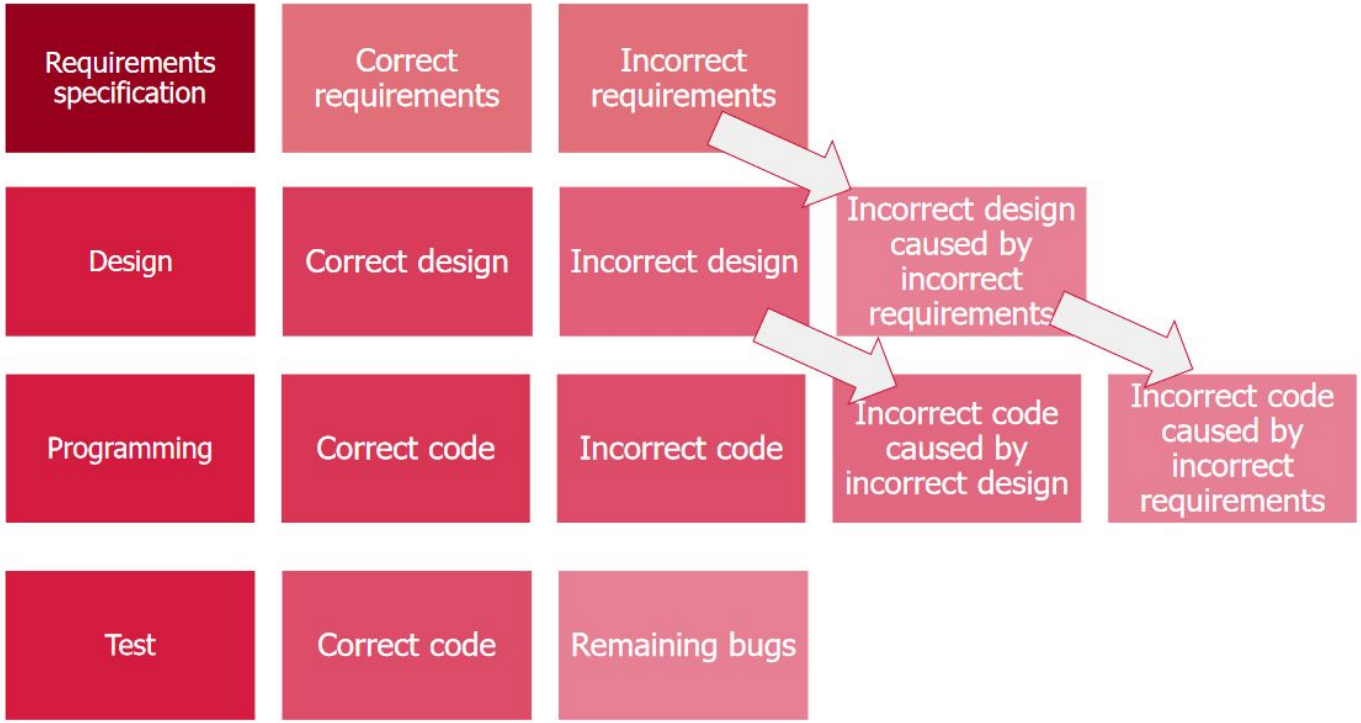
Quality Assurance

Quality

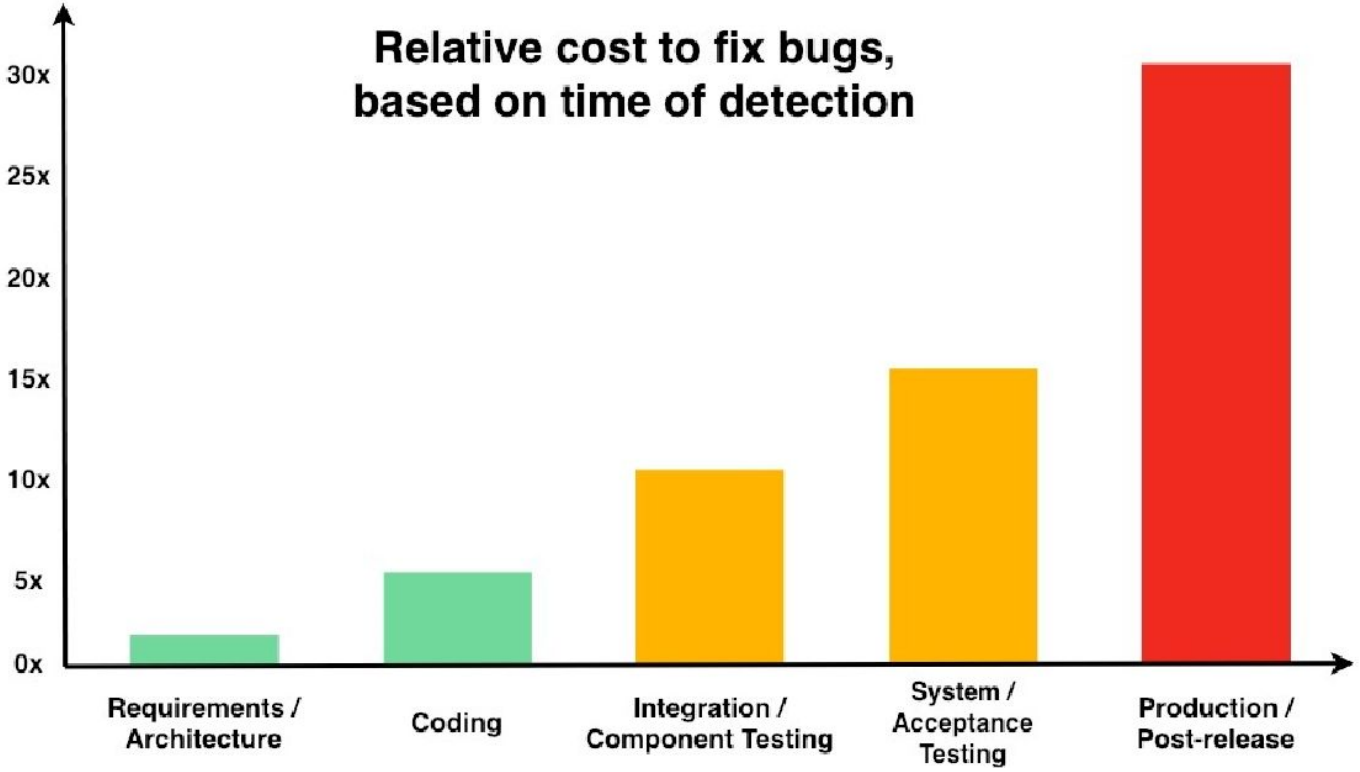
Could for example be:

- Correctness
- Maintainability
- Reliability
- Robustness

When do bugs occur?



Cost of bugs


Relative cost to fix bugs,
based on time of detection
30x
25x
20x
15x
10x
5x
0x
Requirements / Architecture
Coding
Integration / Component Testing
System / Acceptance Testing
Production / Post-release

DevOps

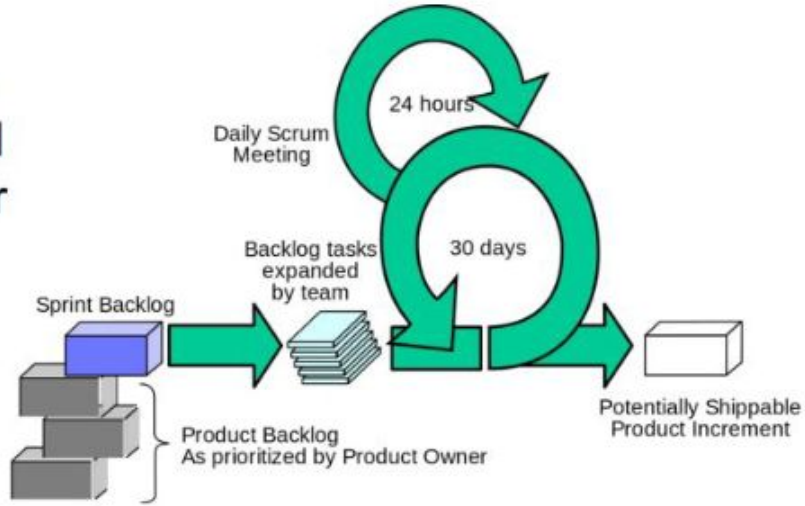

CODE
PLAN
BUILD
TEST
Dev
RELEASE
DEPLOY
OPERATE
MONITOR
Ops

Agile practices

Agile development

What is Agile software development?

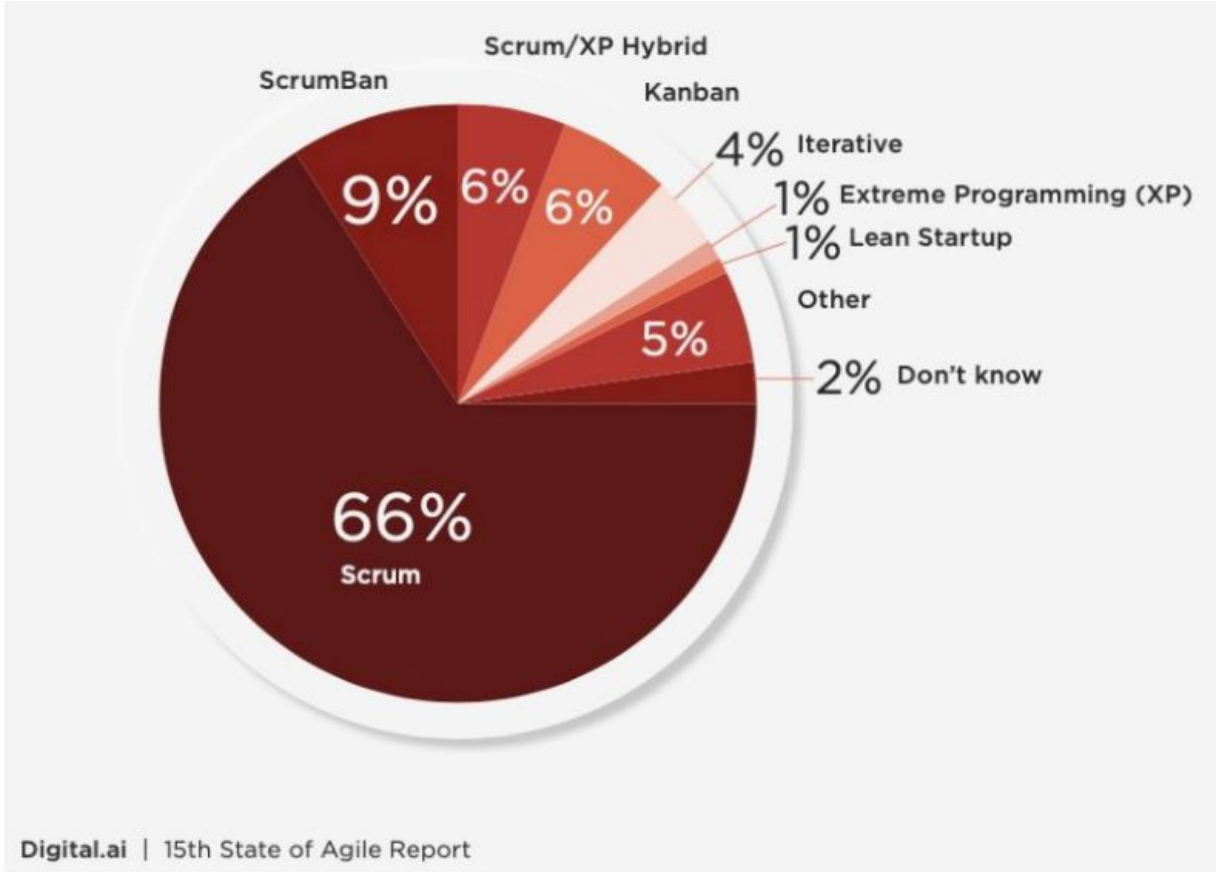
- **«Agile»** refers to a set of **iterative** approaches to software delivery that builds software **incrementally** from the start of the project, instead of trying to deliver it all at once near the end.
  - An **iteration** is a cycle in the development
    - Popularly known as a sprint
  - An **increment** is a subset of requirements for the system
    - For example, backlog items, user stories



Agile methodologies

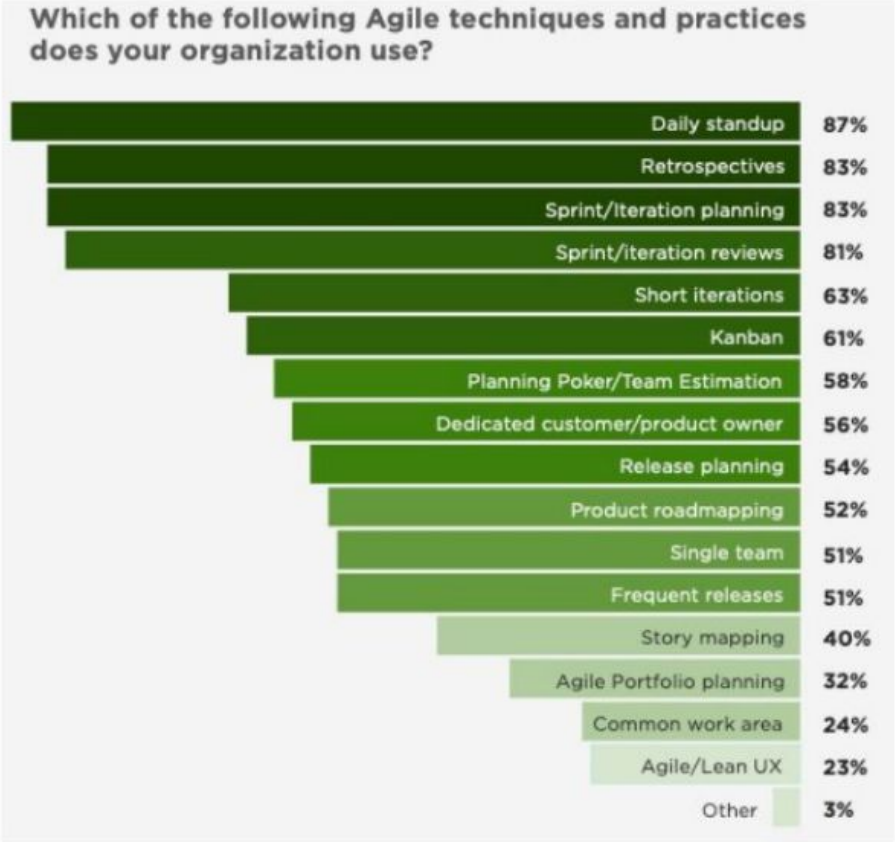
Which agile methodology do you think most teams use?

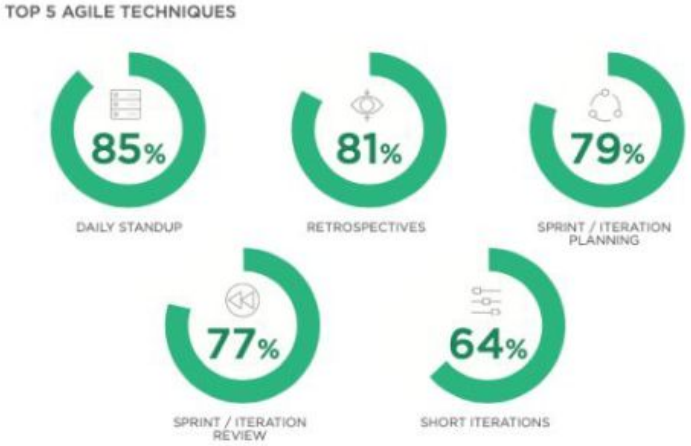
Results from a large international survey →



Digital.ai, "Annual State of Agile Report"
https://digital.ai/resource-center/analyst-reports/state-of-agile-report/

Popular agile practices





Digital.ai, “Annual State of Agile Report”
https://digital.ai/resource-center/analyst-reports/state-of-agile-report/

Questions?

Good luck on the exam! :)