

IN5140

Smart processes and agile methods in Software Engineering

Group session 7

Agenda

First hour:

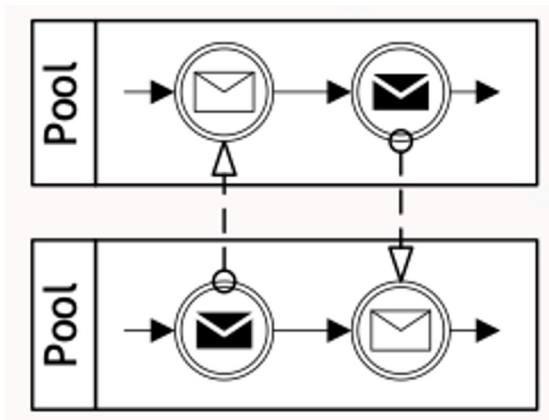
- Delivery 2
- Large-Scale Agile
- Weekly tasks

Second hour:

- Weekly tasks
- Retrospective

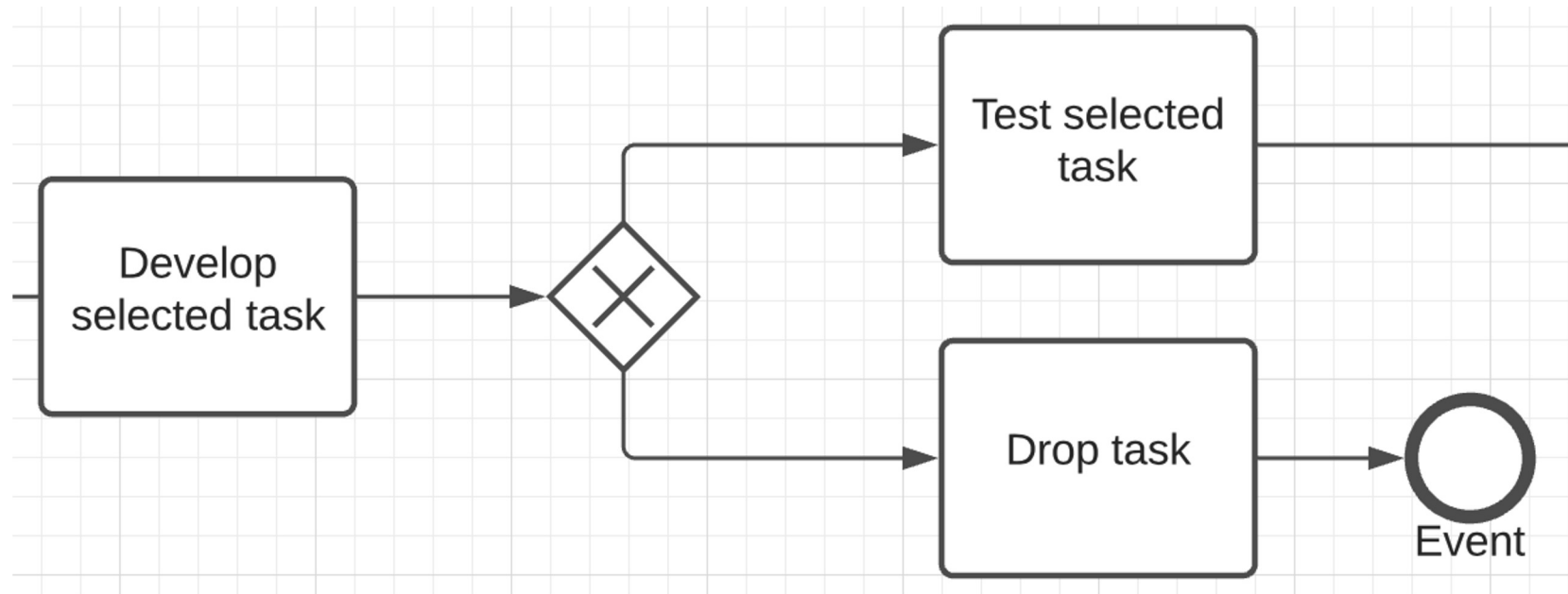
Deliverable 2 - Points to consider

- All pools should have a start-event **and** end-event **with descriptions**
 - Important to visualize what initiates the process
 - Not necessary in swimlanes (but can be included)
 - Remember to name your pools!
- Dotted lines between **pools**, sequence flow between **swimlanes**.



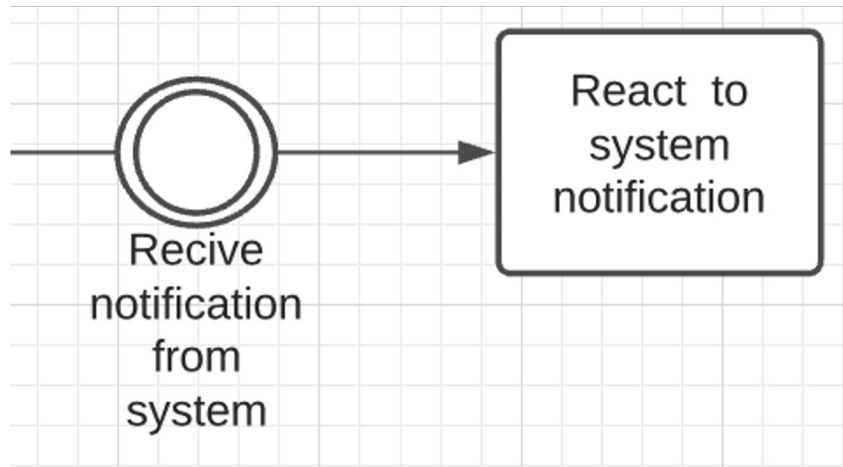
Deliverable 2 - Points to consider

- **All** gateways and events must have a description



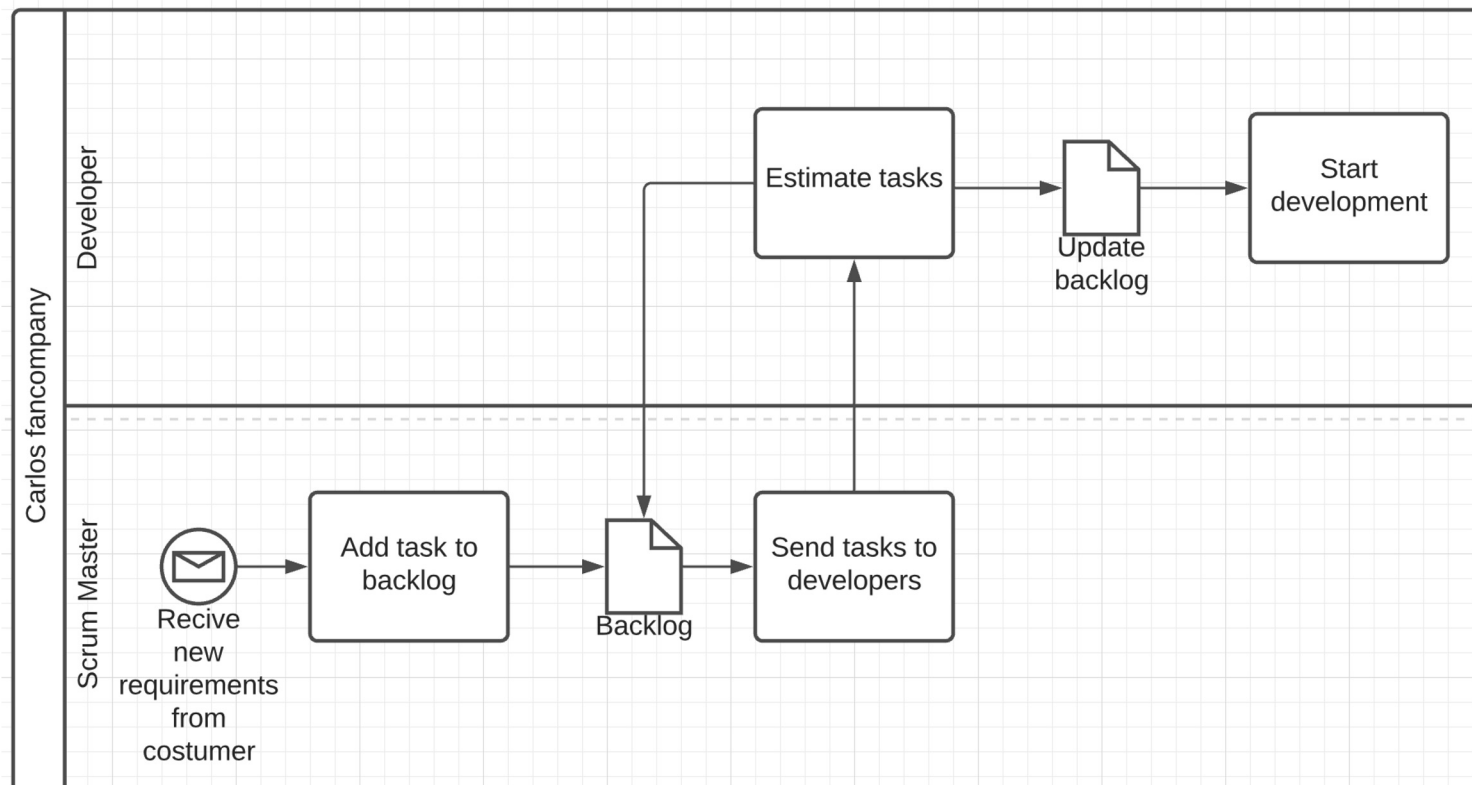
Deliverable 2 - Points to consider

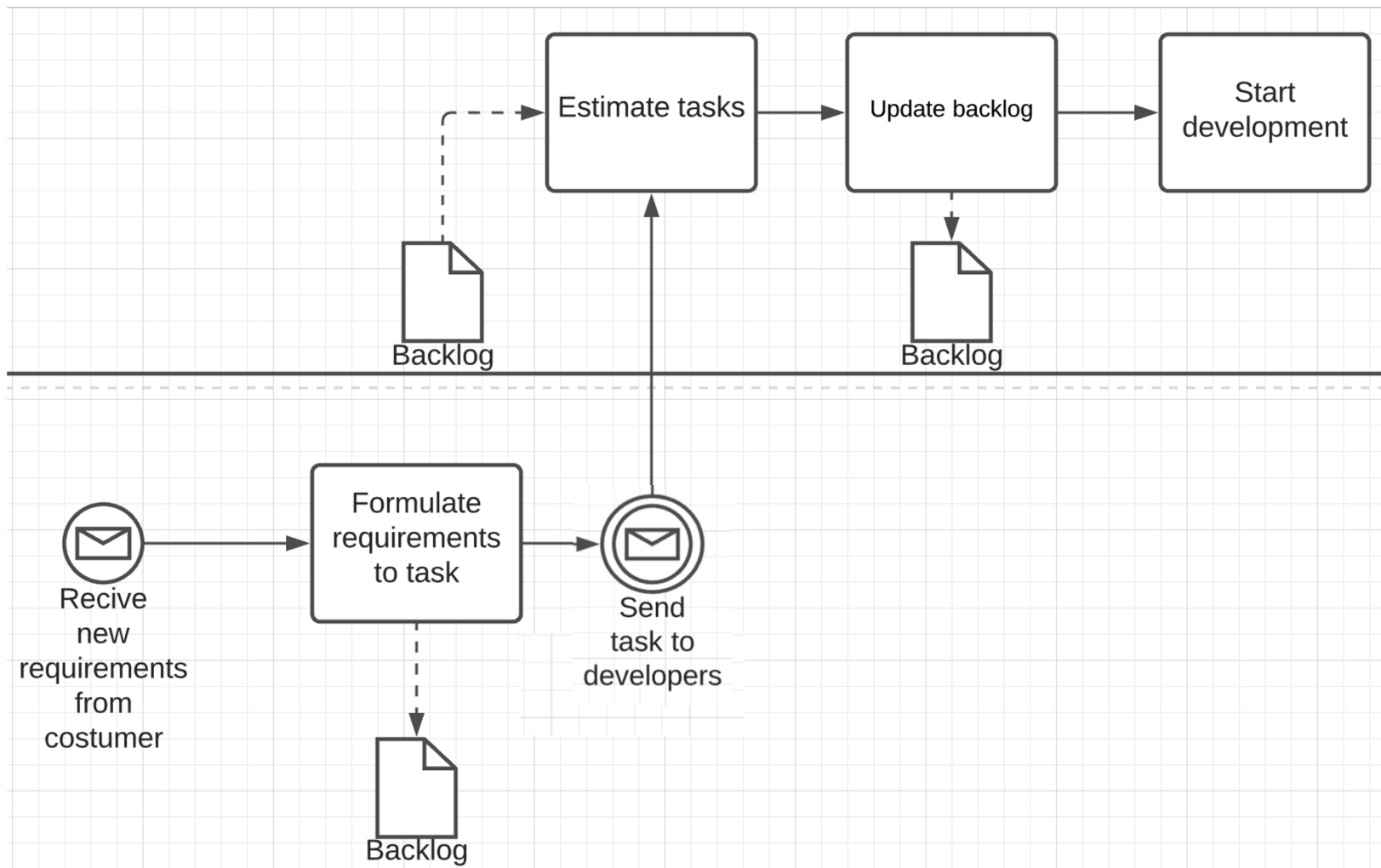
- Connect to data types using Data Association (dotted lines)
- In practise data and systems is used in most activities, you do not have to model all of them everytime
- Data should not “push” new flow directly
 - Alternative solutions to this:

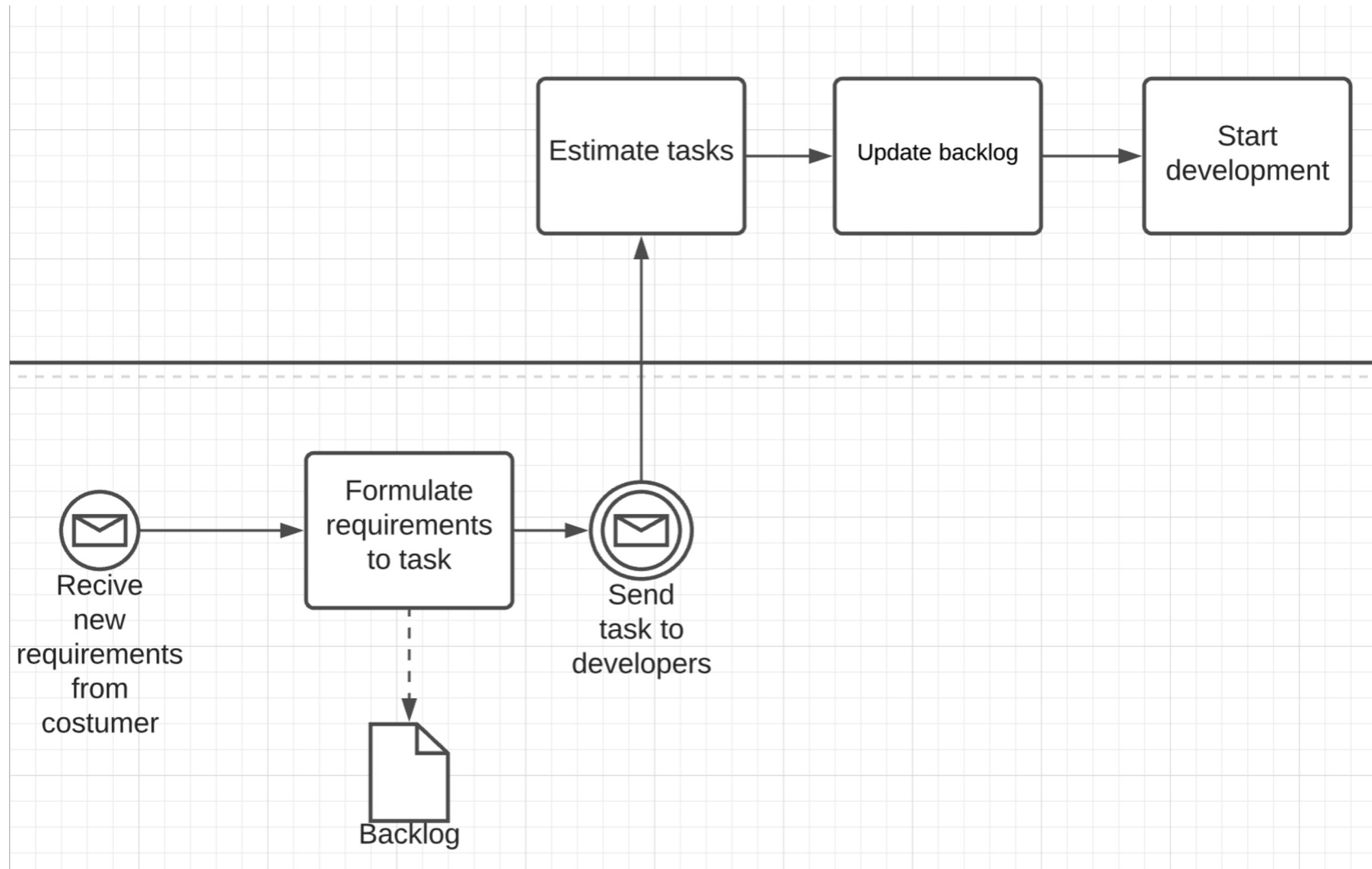


Deliverable 2 - Points to consider

- Data mistakes





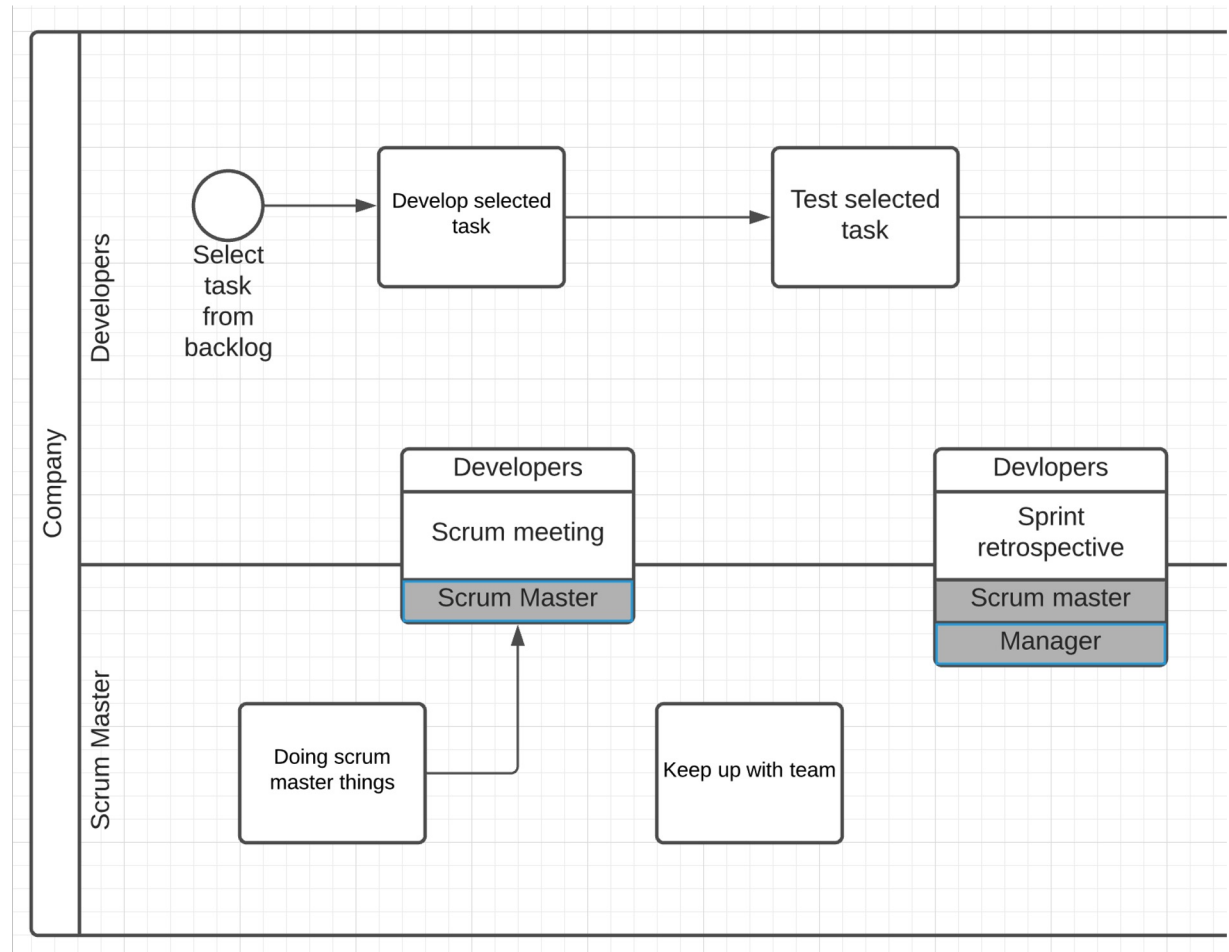


Deliverable 2 - Points to consider

- The goal of a BPMN diagram is to visualize a process (simplified)
 - Different stakeholders with different knowledge and experience
- It is therefore important to have a **readable** and **understandable** diagram.
Keep it simple!
- Keep in mind “Are we adding **valuable** details or **unnecessary** clutter?”
- Examples of “over-complicating”
 - Unnecessary sub-processes (sub-processes within subprocesses)
 - Focus on your process and the problems! Adding detail is great, but make sure that the main process/problem is not lost (especially important for the exam)
 - The “flow” should make sense through the whole diagram.

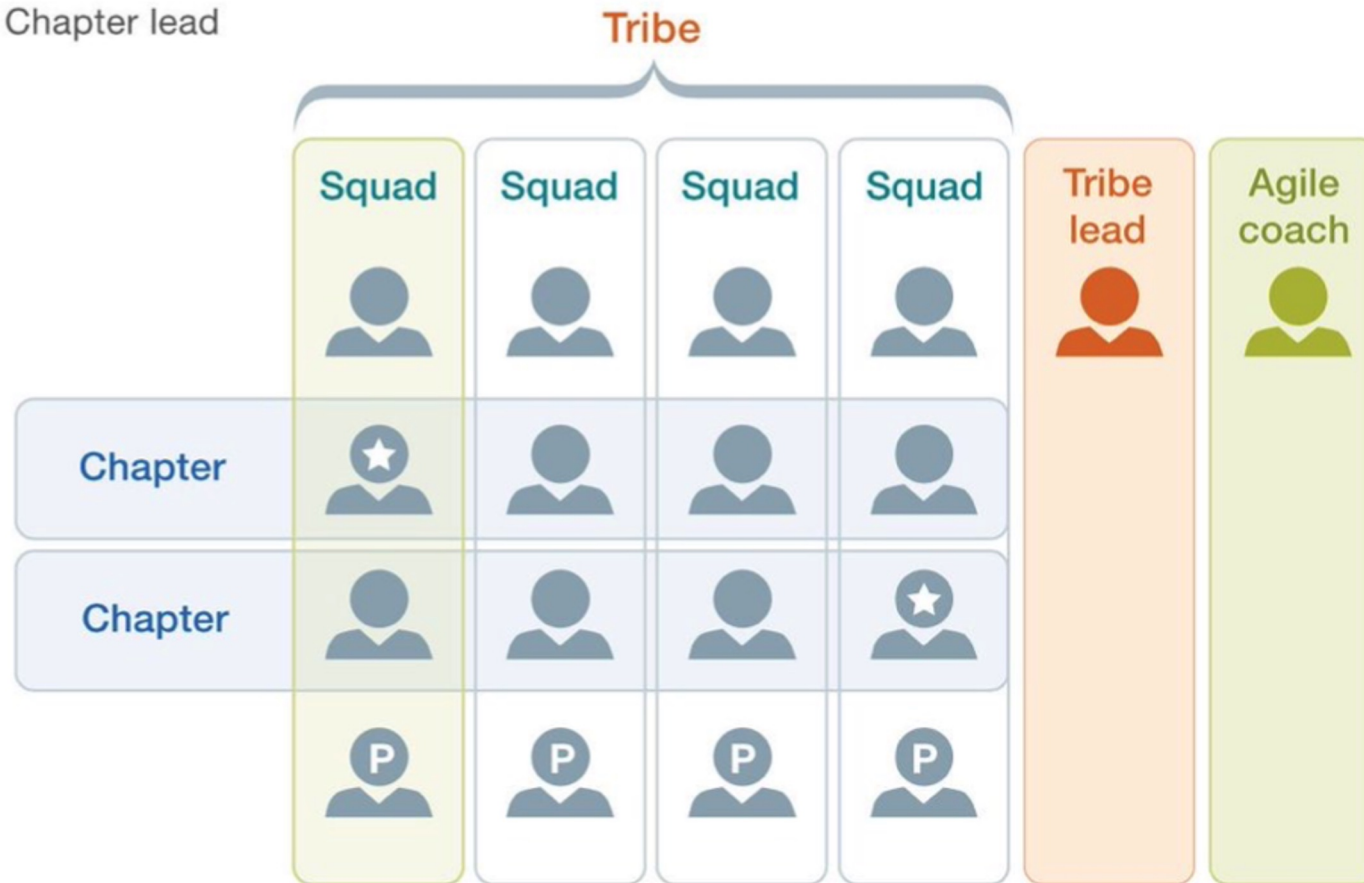
Deliverable 2 - Points to consider

Example of flow mistakes and unnecessary activities



Large-Scale Agile

- P Product owner
- ★ Chapter lead



Tribe

(collection of squads with interconnected missions)

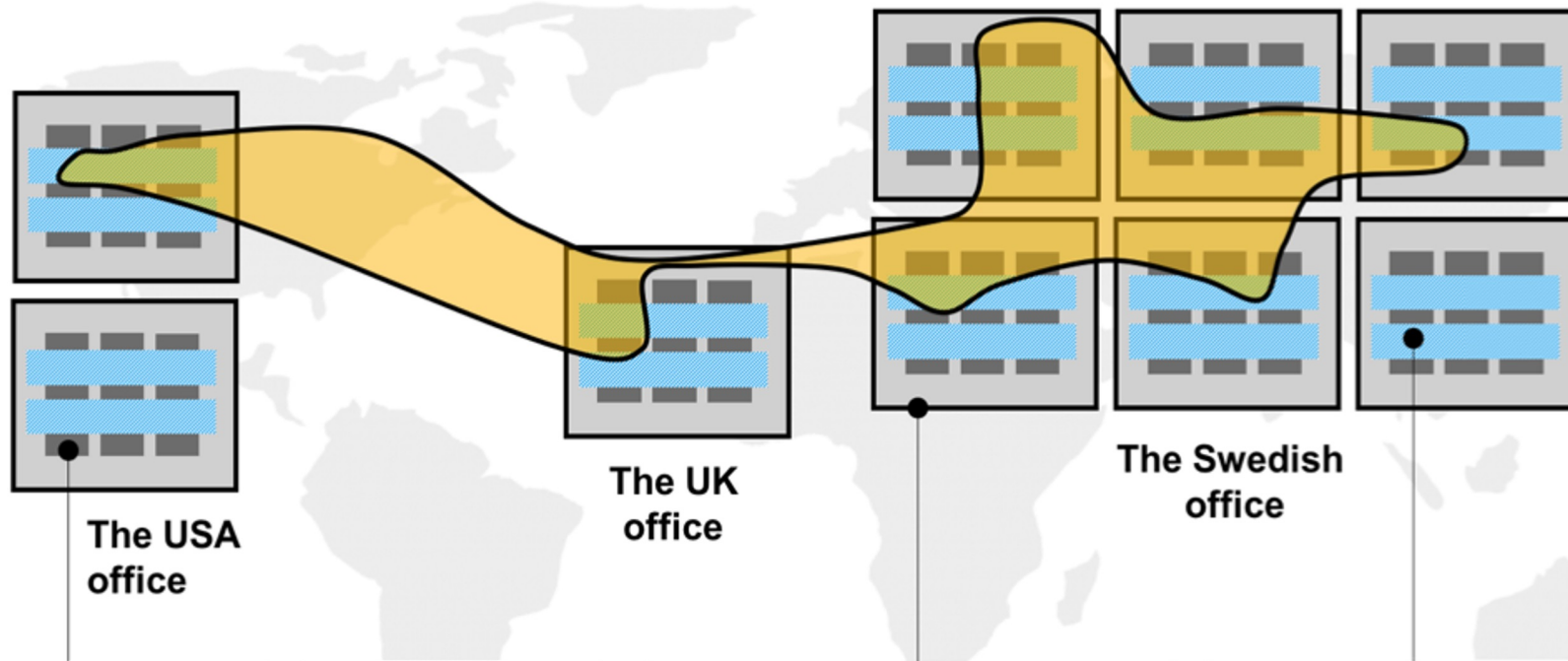
Squad

(basis of new agile organization)

Chapter

(develops expertise and knowledge across squads)

Basic organizational structures: ■ Squads ■ Chapters ■ Tribes ■ Guilds



The USA office

The UK office

The Swedish office

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

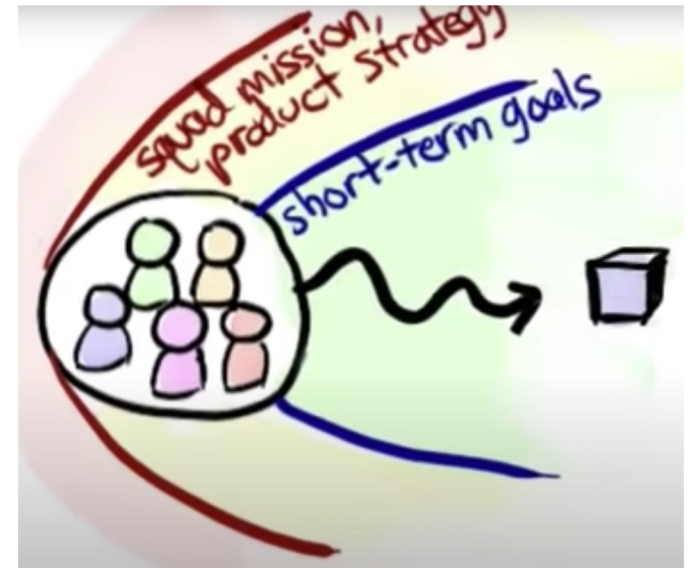
A **guild** is a group of people with similar skills and interests that share knowledge, tools or code across Spotify.

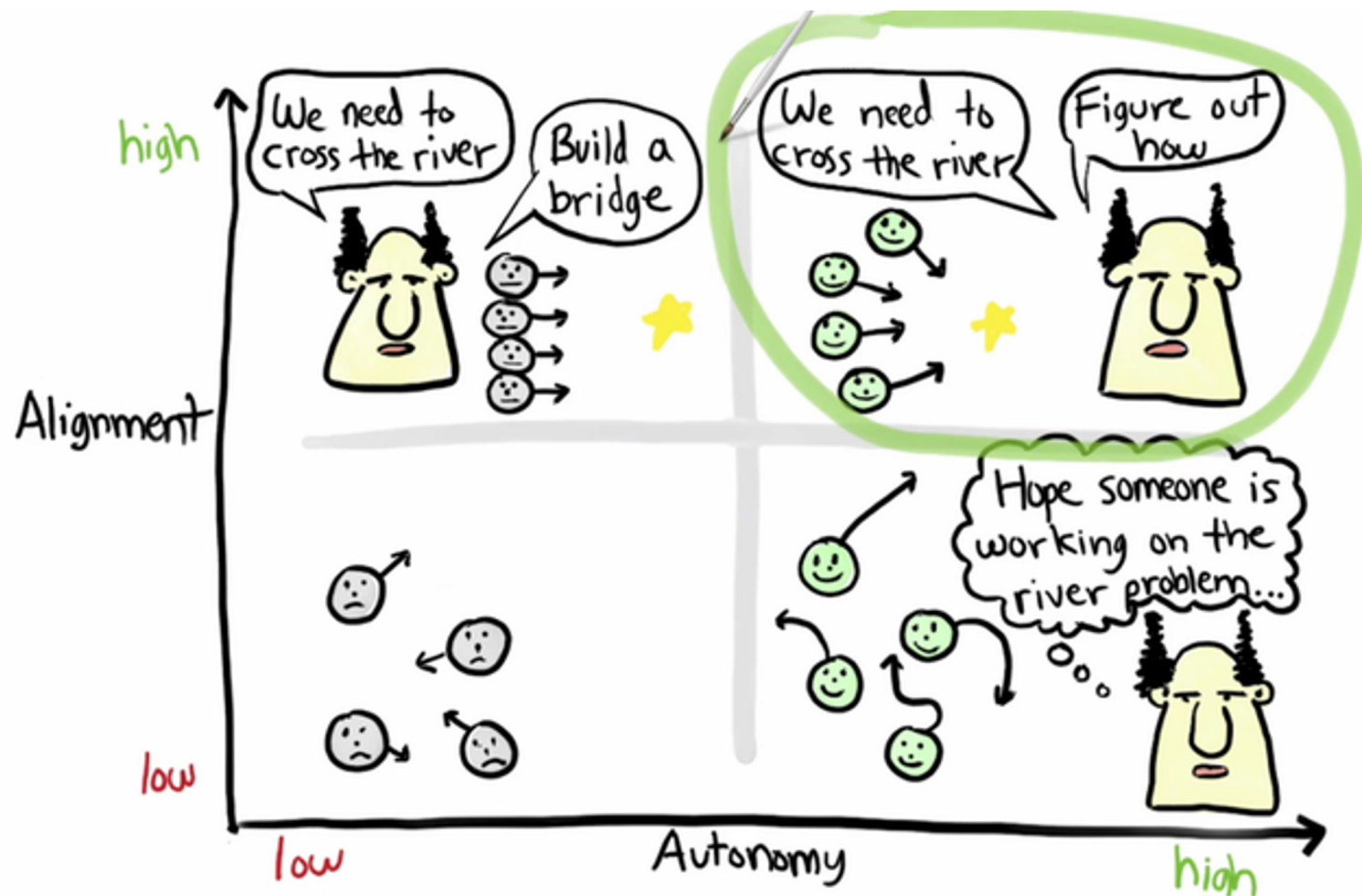
All squads are organized into **tribes** containing 30-200 people each. Tribes have a clear mission, set of principles, a senior experienced leader, and all skills needed to engineer working software features end-to-end.

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.

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





Weekly tasks

Weekly tasks

- Main challenges/issues when scaling up Agile?
- When should you make decisions about architecture?
- What drives the architecture?
- What does “continuous delivery” mean?
- What is Lean vs Agile, same or different?

Weekly tasks

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	Different approaches emerge in a multi-team environment 9 (21%) Interpretation of agile differs between teams Using old and new approaches side by side
Lack of investment 13 (31%) Lack of coaching Lack of training Too high workload Old commitments kept Challenges in rearranging physical spaces	Hierarchical management and organizational boundaries 14 (33%) Middle managers' role in agile unclear Management in waterfall mode Keeping the old bureaucracy Internal silos kept
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	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
Coordination challenges in multi-team environment 13 (31%) Interfacing between teams difficult Autonomous team model challenging Global distribution challenges Achieving technical consistency	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)

Team-related challenges

Table 11
Challenges.

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

Weekly tasks

When should you make decisions about architecture?

- Emergent architecture vs upfront architecture
- Postponing architectural decisions can increase the probability of being correct
- More information can change our perception of the risks

Weekly tasks

What drives the architecture?

- Business goals!
- Fancy architecture have little value if it does not help with the business goals



Weekly tasks

What does “continuous delivery” mean?

Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, safely and quickly in a sustainable way.”

15 min break



The Sailboat Retrospective



Group Session Retrospective!

We want your feedback!

... and what better way to do it, than a Group Session Retrospective!

Project Report Help-session