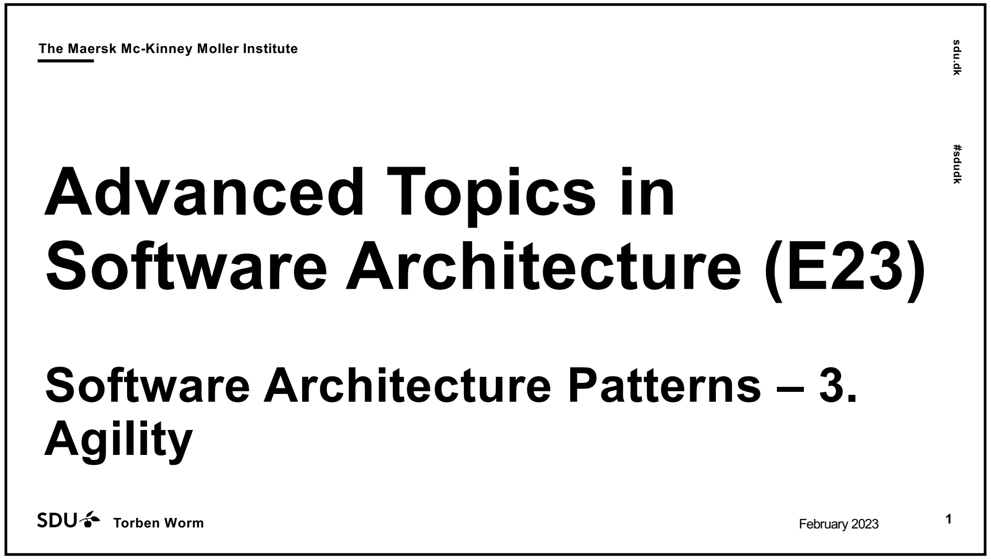


The Maersk Mc-Kinney Moller Institute

sdu.dk

#sduk

Advanced Topics in Software Architecture (E23)

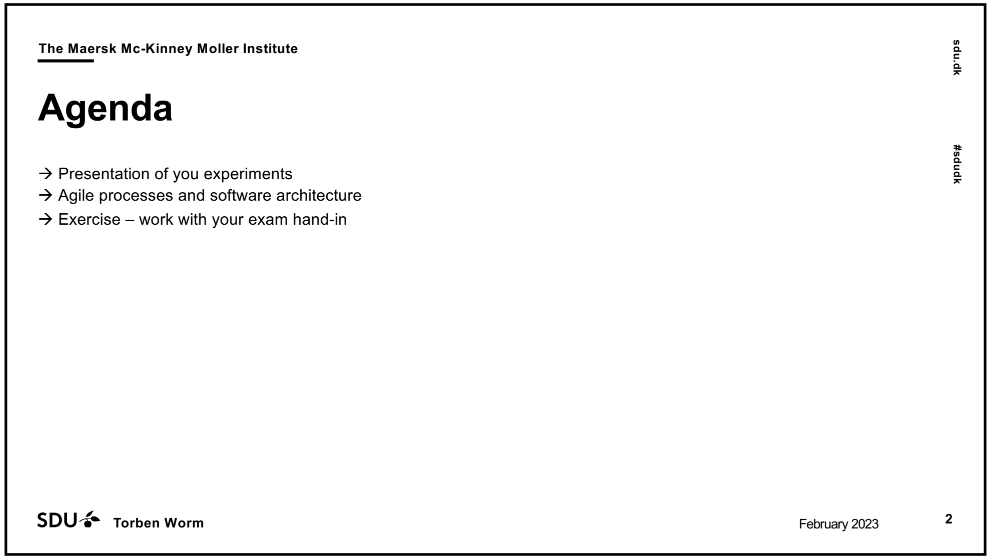
Software Architecture Patterns – 3. Agility

SDU Torben Worm

February 2023

1

1



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sduk

Agenda

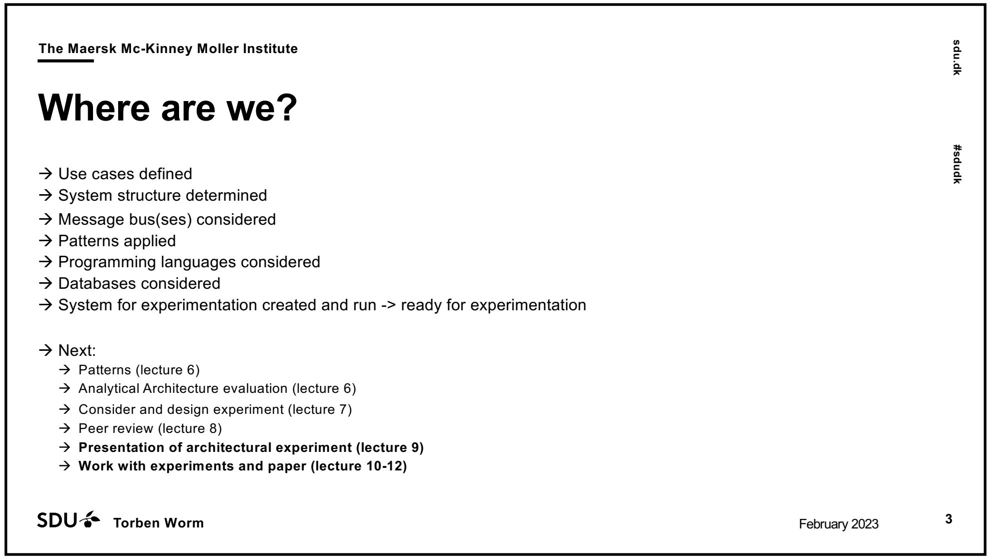
→ Presentation of you experiments
→ Agile processes and software architecture
→ Exercise – work with your exam hand-in

SDU Torben Worm

February 2023

2

2



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

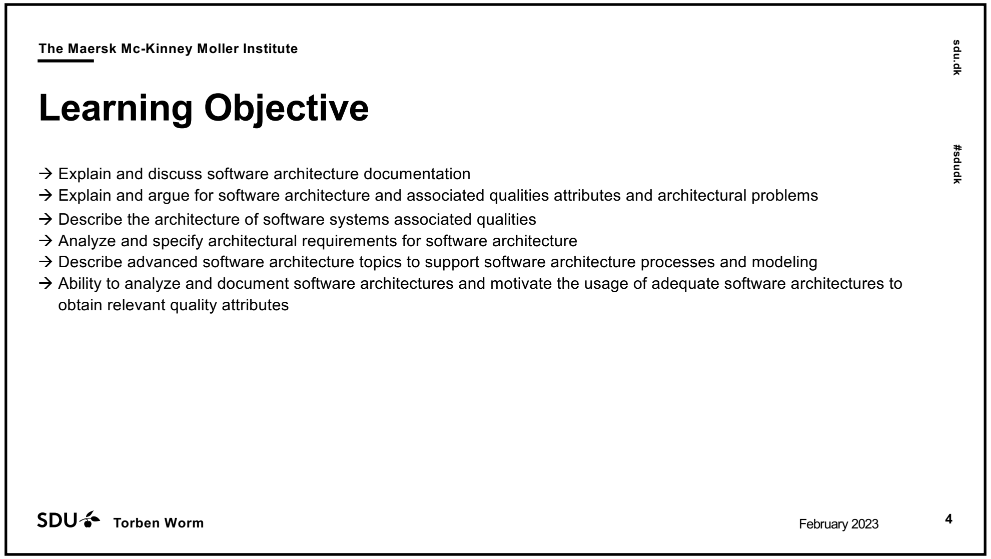
Where are we?

- → Use cases defined
- → System structure determined
- → Message bus(ses) considered
- → Patterns applied
- → Programming languages considered
- → Databases considered
- → System for experimentation created and run -> ready for experimentation

- → Next:
  - → Patterns (lecture 6)
  - → Analytical Architecture evaluation (lecture 6)
  - → Consider and design experiment (lecture 7)
  - → Peer review (lecture 8)
  - → **Presentation of architectural experiment (lecture 9)**
  - → **Work with experiments and paper (lecture 10-12)**

SDU Torben Worm

February 2023



The Maersk Mc-Kinney Moller Institute

sdu.dk

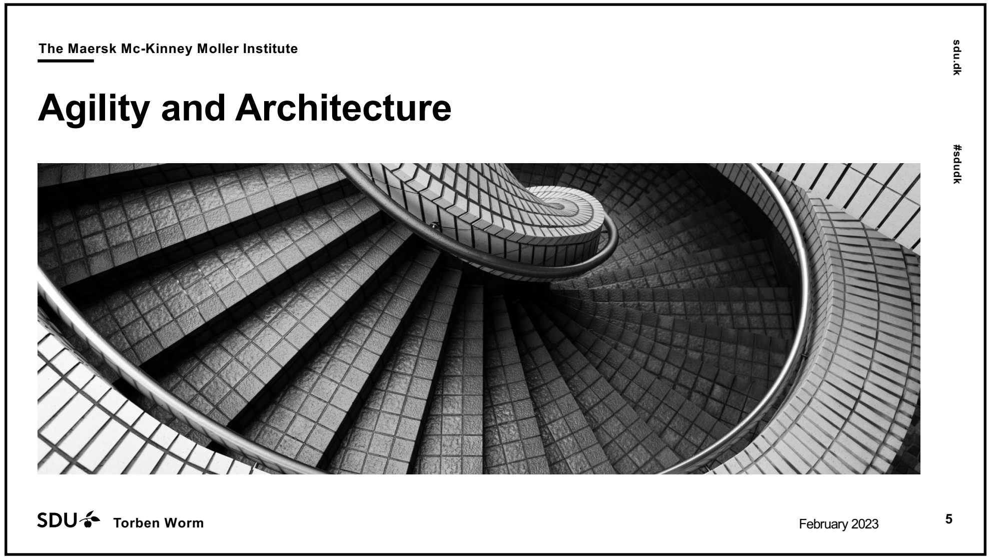
#sdudk

Learning Objective

- → Explain and discuss software architecture documentation
- → Explain and argue for software architecture and associated qualities attributes and architectural problems
- → Describe the architecture of software systems associated qualities
- → Analyze and specify architectural requirements for software architecture
- → Describe advanced software architecture topics to support software architecture processes and modeling
- → Ability to analyze and document software architectures and motivate the usage of adequate software architectures to obtain relevant quality attributes

SDU Torben Worm

February 2023



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

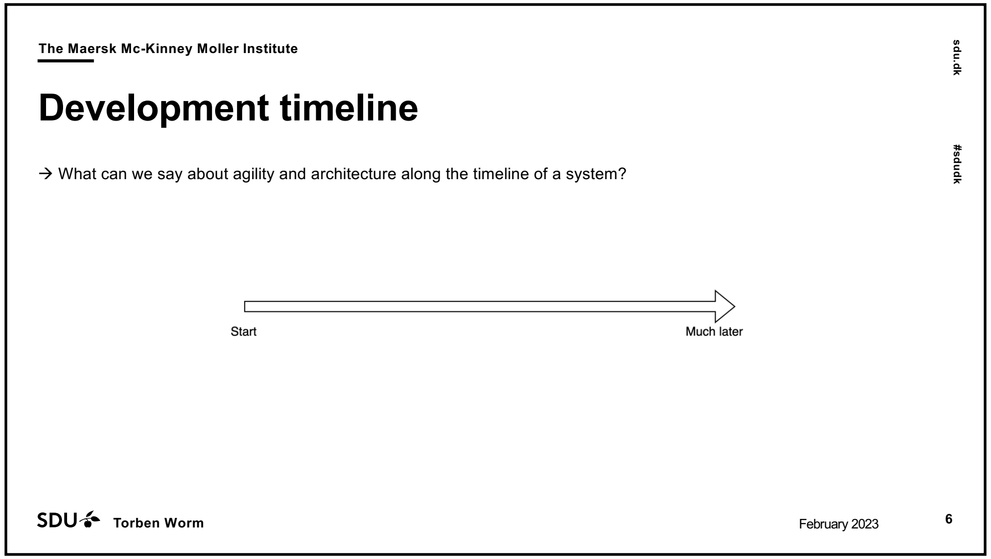
Agility and Architecture

SDU Torben Worm

February 2023

5

5



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

Development timeline

→ What can we say about agility and architecture along the timeline of a system?

SDU Torben Worm

February 2023

6

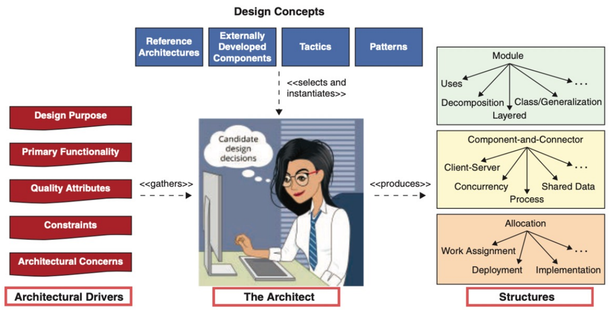
6

Characteristics of systems

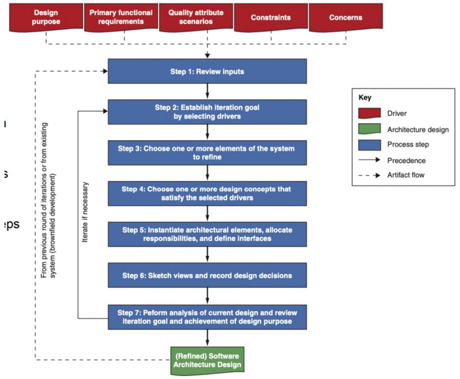
→ What can we say about architure and agility if we look at systems from these perspectives?

		Lifespan	
		Short	Long
Size	Large		
	Small		

Architecture Design Activity



Steps and Artifacts of ADD


Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
From previous round of iterations or from existing system (brownfield development)
Iterate if necessary
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

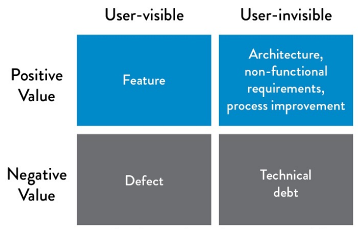
What is Agile?

- → A set of concepts/approaches to avoid ”software bureaucracy”
  - → Code-centered (usually)
  - → Continuous testing
  - → Refactoring

- → Based on the Agile Manifesto (http://agilemanifesto.org):
- → We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
  - → Individuals and interactions over processes and tools ▶ software over comprehensive documentation
  - → Customer collaboration over contract negotiation
  - → Responding to change over following a plan
  - → That is, while there is value in the items on the right, we value the items on the left more.

- → What are the implications on architecture (agile ←→ architecture)?

(In)visible value



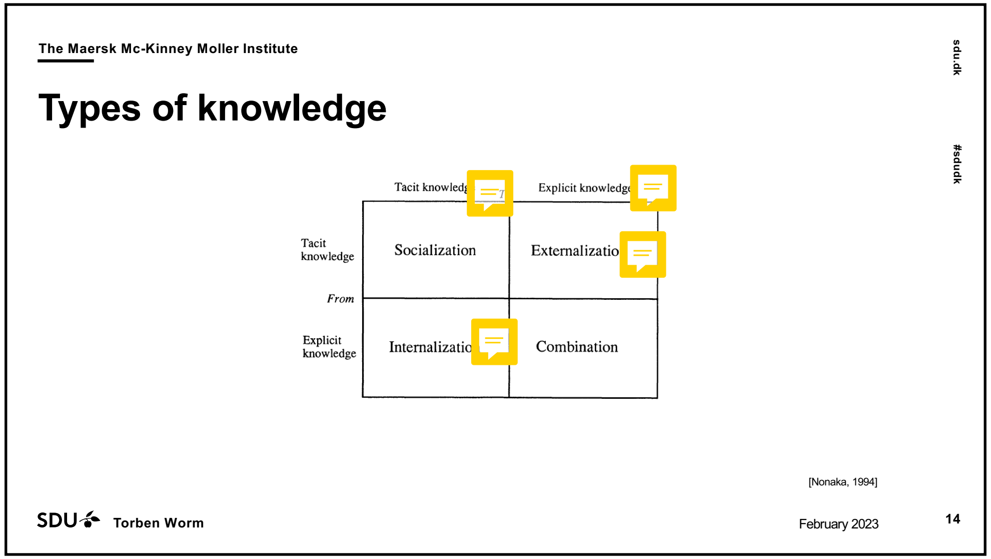
	User-visible	User-invisible
Positive Value	Feature	Architecture, non-functional requirements, process improvement
Negative Value	Defect	Technical debt

Figure 11: *Invest 20% of cycles on those that create positive, user-invisible value (Source: "Machine Learning and Technical Debt with D. Sculley,"* Software Engineering Daily *podcast, November 17, 2015, http://softwareengineeringdaily.com/2015/11/17/machine-learning-and-technical-debt-with-d-sculley/.)*

Architecture and Knowledge

→ Architecture is knowledge (about the structure of a system)
→ Conveying the architecture to different stakeholders is about conveying knowledge

→ What is knowledge?
→ How can we exchange knowledge?



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sduk

Types of knowledge

From

	Tacit knowledg	Explicit knowledge
Tacit knowledge	Socialization	Externalizatio
Explicit knowledge	Internalizatio	Combination

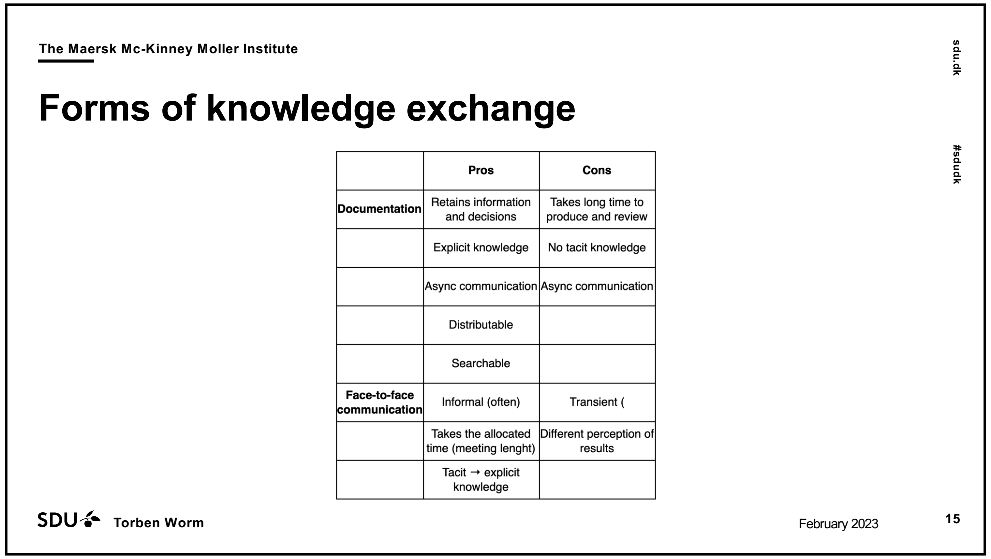
[Nonaka, 1994]

SDU Torben Worm

February 2023

14

14



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sduk

Forms of knowledge exchange

	Pros	Cons
Documentation	Retains information and decisions	Takes long time to produce and review
	Explicit knowledge	No tacit knowledge
	Async communication	Async communication
	Distributable	
	Searchable	
Face-to-face communication	Informal (often)	Transient (
	Takes the allocated time (meeting lenght)	Different perception of results
	Tacit → explicit knowledge	

SDU Torben Worm

February 2023

15

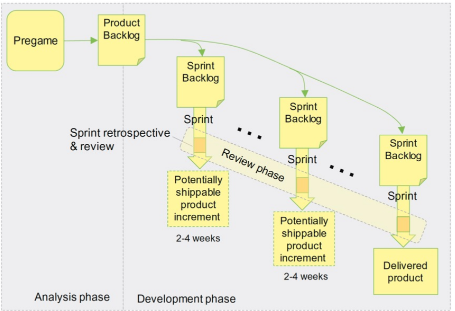
15

The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

SCRUM



[Babar, 2013]

SDU Torben Worm

February 2023

16

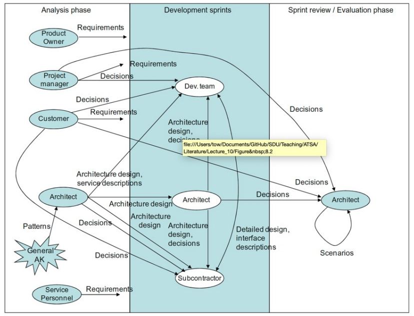
16

The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

Big-up-front and sprint-zero - results



[Babar, 2013]

SDU Torben Worm

February 2023

17

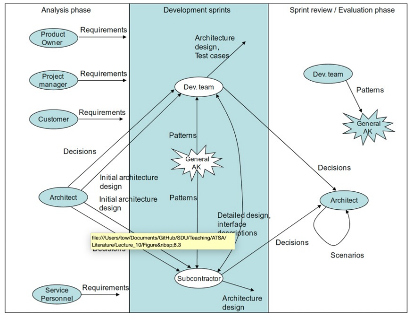
17

The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

In-sprints - results



[Babar, 2013]

SDU Torben Worm

February 2023

18

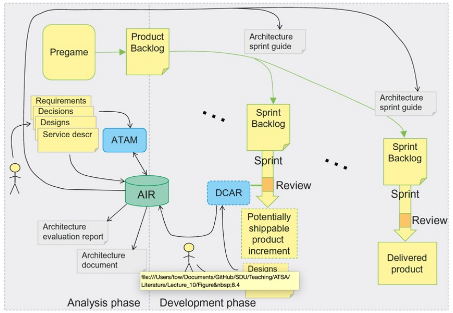
18

The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

Big-up-front and sprint-zero


Pregame
Product Backlog
Architecture sprint guide
Requirements
Decisions
Designs
Service descr
ATAM
AIR
Architecture evaluation report
Architecture document
...
Sprint Backlog
Sprint
Review
DCAR
Potentially shippable product increment
Designs
file:///Users/tow/Documents/GitHub/SDU/Teaching/ATSA/Literature/Lecture_10/Figure 8.4
...
Architecture sprint guide
Sprint Backlog
Sprint
Review
Delivered product
Analysis phase
Development phase

[Babar, 2013]

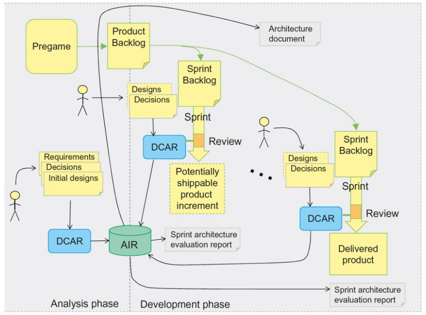
SDU Torben Worm

February 2023

19

19

In-sprints

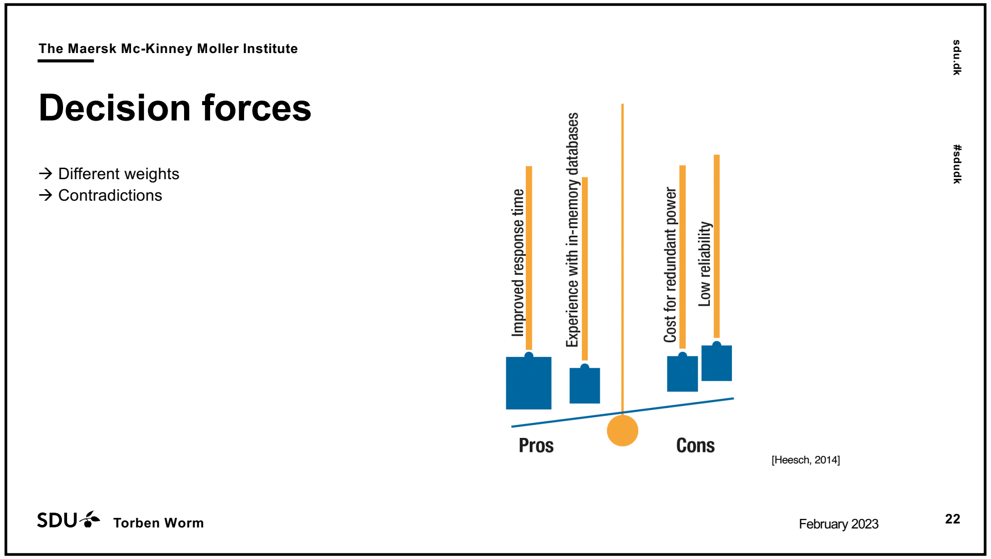


[Babar, 2013]

DCAR

→ **Evaluation objectives**: determine the soundness of architectural decisions that were made
→ **Inputs for evaluation**: informal description of requirements, business drivers, and architectural design
→ **Knowledge of evaluators**: general knowledge about software architecture
→ **Output**: risks, issues, and thorough documentation of the evaluated decisions and their decision forces
→ **Priority setting of decisions**: during the review
→ **Project phase**: within or after the architectural design is finalized
→ **Reviewers**: company-internal or external reviewers
→ **Schedule**: half a day preparation and postprocessing and half a day review session
→ **Scope**: a set of specific architecture decisions
→ **Social interaction**: face-to-face meeting between reviewers, architect, developers, and business representative
→ **Tools or automation**: templates, wiki, and UML too

[Heesch, 2014]



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

Decision forces

→ Different weights
→ Contradictions

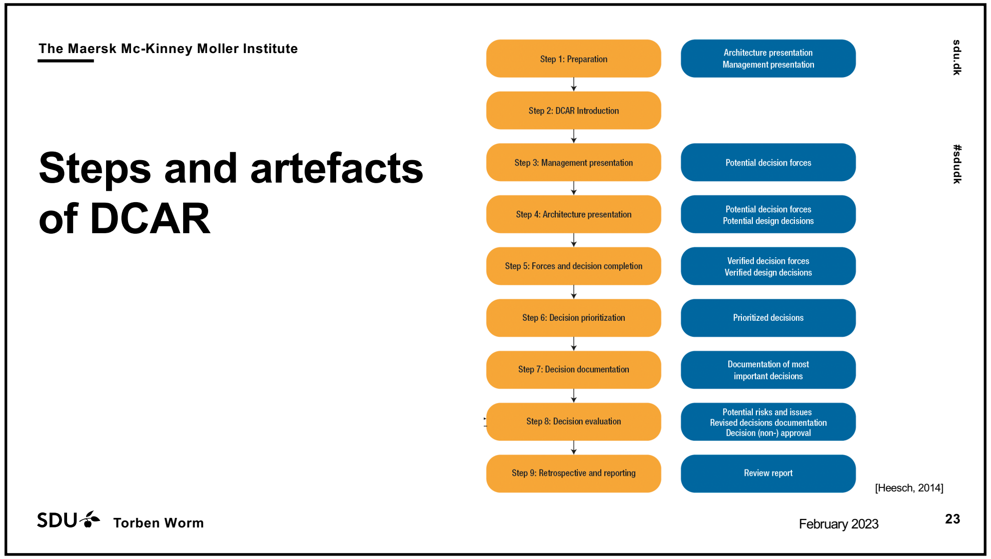
[Heesch, 2014]

SDU Torben Worm

February 2023

22

22



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

Steps and artefacts of DCAR

Step 1: Preparation — Architecture presentation
Management presentation

Step 2: DCAR Introduction

Step 3: Management presentation — Potential decision forces

Step 4: Architecture presentation — Potential decision forces
Potential design decisions

Step 5: Forces and decision completion — Verified decision forces
Verified design decisions

Step 6: Decision prioritization — Prioritized decisions

Step 7: Decision documentation — Documentation of most important decisions

Step 8: Decision evaluation — Potential risks and issues
Revised decisions documentation
Decision (non-) approval

Step 9: Retrospective and reporting — Review report

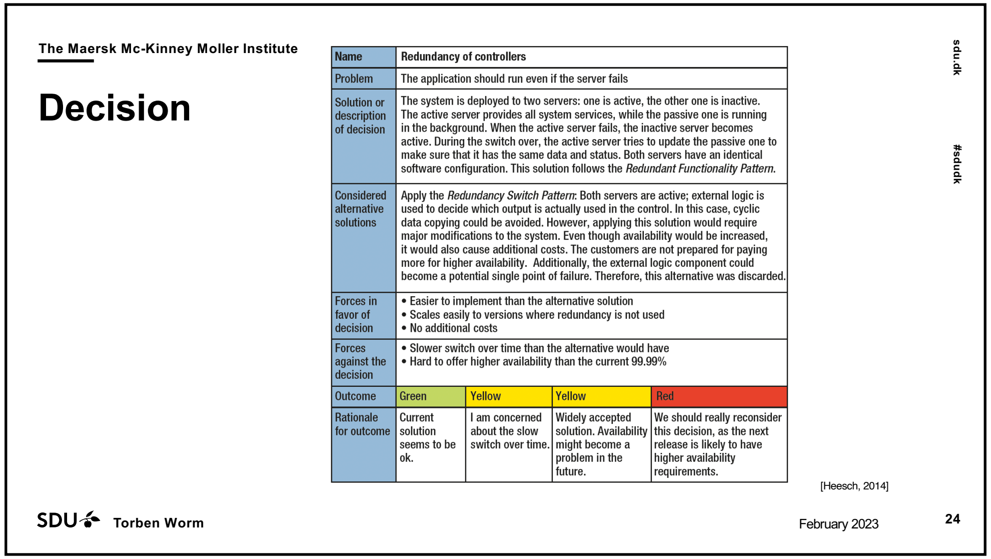
[Heesch, 2014]

SDU Torben Worm

February 2023

23

23



The Maersk Mc-Kinney Moller Institute

Decision

Name	**Redundancy of controllers**			
Problem	The application should run even if the server fails			
Solution or description of decision	The system is deployed to two servers: one is active, the other one is inactive. The active server provides all system services, while the passive one is running in the background. When the active server fails, the inactive server becomes active. During the switch over, the active server tries to update the passive one to make sure that it has the same data and status. Both servers have an identical software configuration. This solution follows the *Redundant Functionality Pattern*.			
Considered alternative solutions	Apply the *Redundancy Switch Pattern*: Both servers are active; external logic is used to decide which output is actually used in the control. In this case, cyclic data copying could be avoided. However, applying this solution would require major modifications to the system. Even though availability would be increased, it would also cause additional costs. The customers are not prepared for paying more for higher availability. Additionally, the external logic component could become a potential single point of failure. Therefore, this alternative was discarded.			
Forces in favor of decision	• Easier to implement than the alternative solution • Scales easily to versions where redundancy is not used • No additional costs			
Forces against the decision	• Slower switch over time than the alternative would have • Hard to offer higher availability than the current 99.99%			
Outcome	Green	Yellow	Yellow	Red
Rationale for outcome	Current solution seems to be ok.	I am concerned about the slow switch over time.	Widely accepted solution. Availability might become a problem in the future.	We should really reconsider this decision, as the next release is likely to have higher availability requirements.

[Heesch, 2014]

sdu.dk

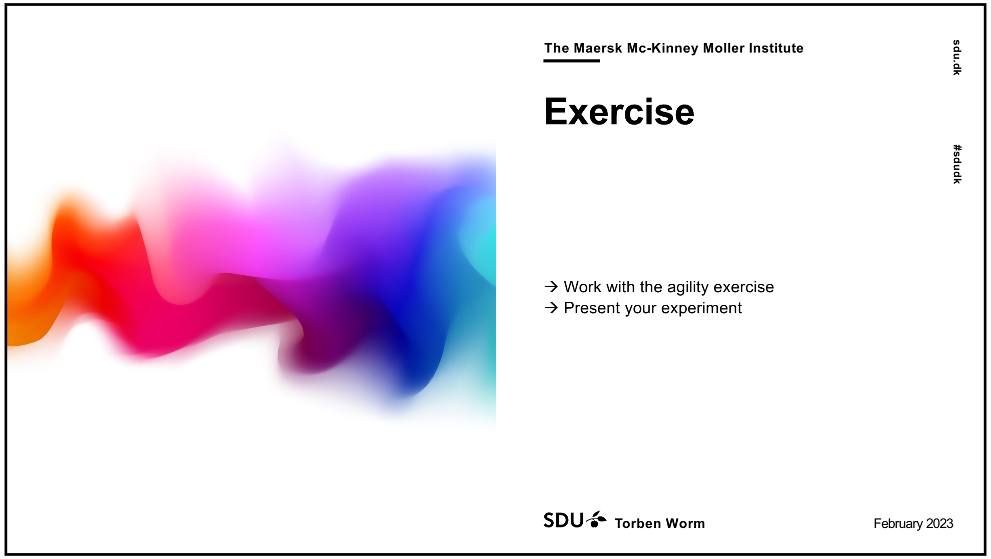
#sdudk

SDU Torben Worm

February 2023

24

24



The Maersk Mc-Kinney Moller Institute

Exercise

→ Work with the agility exercise
→ Present your experiment

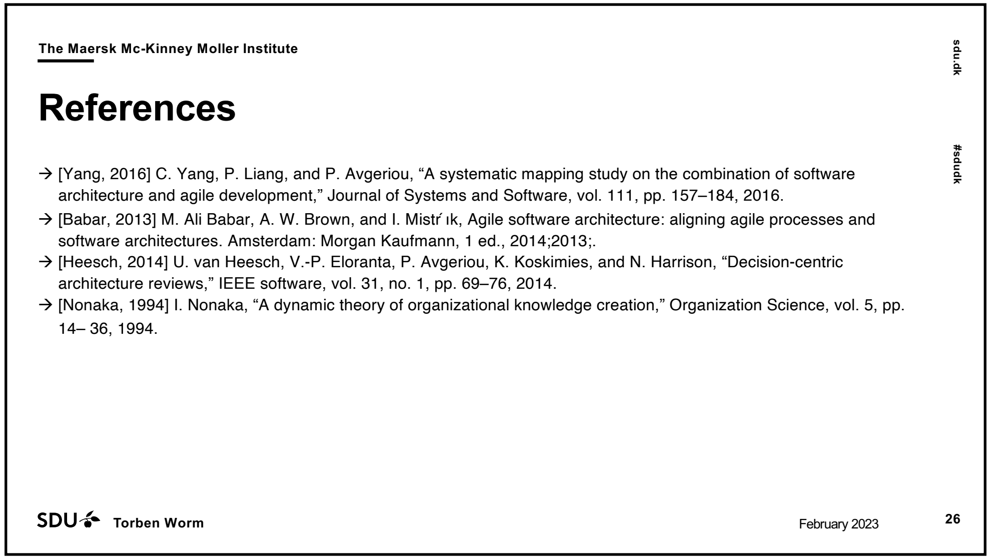
sdu.dk

#sdudk

SDU Torben Worm

February 2023

25



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sduck

References

→ [Yang, 2016] C. Yang, P. Liang, and P. Avgeriou, “A systematic mapping study on the combination of software architecture and agile development,” Journal of Systems and Software, vol. 111, pp. 157–184, 2016.

→ [Babar, 2013] M. Ali Babar, A. W. Brown, and I. Mistŕ ık, Agile software architecture: aligning agile processes and software architectures. Amsterdam: Morgan Kaufmann, 1 ed., 2014;2013;.

→ [Heesch, 2014] U. van Heesch, V.-P. Eloranta, P. Avgeriou, K. Koskimies, and N. Harrison, “Decision-centric architecture reviews,” IEEE software, vol. 31, no. 1, pp. 69–76, 2014.

→ [Nonaka, 1994] I. Nonaka, “A dynamic theory of organizational knowledge creation,” Organization Science, vol. 5, pp. 14– 36, 1994.

The Maersk Mc-Kinney Moller Institute

sdu.dk

#sduck

End of Presentation