

The Maersk Mc-Kinney Møller Institute

Advanced Topics in Software Architecture (E23)

Software Architecture Patterns 2 Evaluating Software Architectures 1

SDU  Torben Worm

February 2023

1


sdumk
#sdumk

1

The Maersk Mc-Kinney Møller Institute

Agenda

- Follow-up on exercise from last lecture
- (Architectural) Patterns
- Evaluating Software Architectures
- Midway evaluation
- Exercise

SDU  Torben Worm

February 2023

2

sdumk
#sdumk

2

The Maersk Mc-Kinney Møller Institute


#sdudk

Where are we?

- Use cases defined
- System structure determined
- Message bus(es) 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

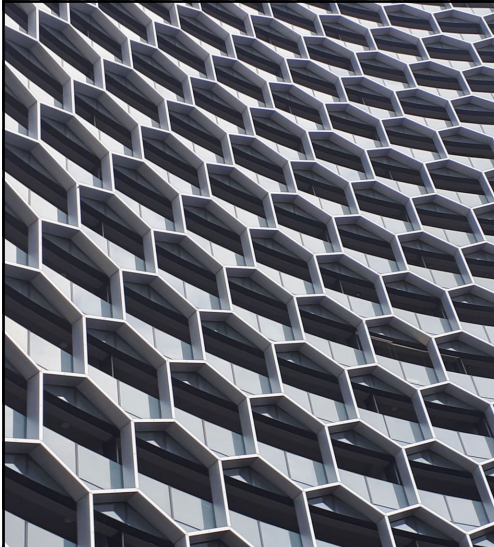
3


3

The Maersk Mc-Kinney Møller Institute

#sdudk

Architectural Patterns 2



SDU  Sune Chung Jepsen

February 2023

4

"Traditional" types of patterns

- Creational
- Structural
- Behavioral

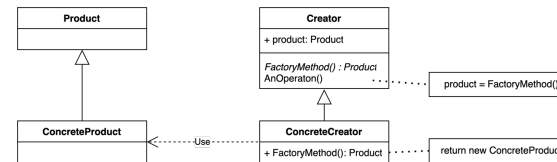
Purpose	Design Pattern	Aspects That Can Vary
Creational	Abstract Factory (87)	families of product objects
	Builder (97)	how a composite object gets created
	Factory Method (1607)	subclass of object that is instantiated
	Prototype (117)	class of object that is instantiated
	Singleton (127)	the sole instance of a class
Structural	Adapter (139)	interface to an object
	Bridge (151)	implementation of an object
	Composite (163)	structure and composition of an object
	Decorator (175)	responsibilities of an object
	Facade (185)	without encapsulating
	Flyweight (195)	interface to a subsystem
	Proxy (207)	storage costs of objects
Behavioral	Chain of Responsibility (225)	how an object is accessed, its location
	Command (233)	object that can fulfill a request
	Command (233)	when and how a request is fulfilled
	Interpreter (243)	grammar and interpretation of a language
	Iterator (257)	how an aggregate's elements are accessed, iterated
	Mediator (273)	how and which objects interact with each other
	Memento (283)	what private information is stored outside an object, and when
	Observer (293)	number of objects that depend on another object, how the dependent objects stay up to date
	State (305)	states of an object
	Strategy (315)	an algorithm
	Template Method (325)	steps of an algorithm
	Visitor (331)	operations that can be applied to objects without changing their classes

Creational

- Abstracts the instantiation process
- Makes the system independent of how objects are
 - Created
 - Comnposed
 - Represented
- Class creational
 - Inheritance
- Object creational
 - Delegation

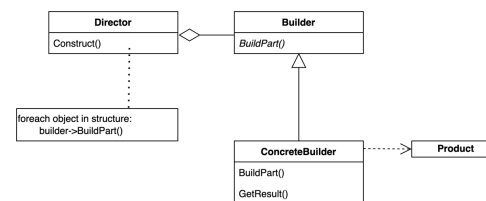
Factory Method – Class creational

- Product: Defines the interface to the object the factory method creates
- ConcreteProduct: Implements the Product interface
- Creator: Declares the factory method
- ConcreteCreator: Overrides the factory method to return an instance of the concrete product



Builder – Object creational

- Director: Constructs the object using the Builder interface
- Builder: Specifies an abstract interface for creating parts of a Product object
- ConcreteBuilder: Constructs and assembles parts of the product by implementing the Builder interface
- Product: Represents the complex object under construction

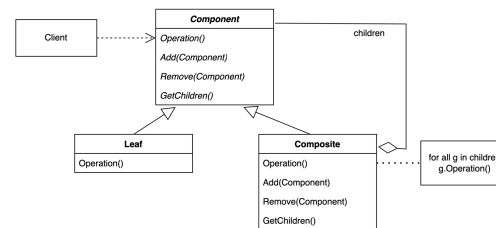


Structural

- Concerned with how classes and objects are composed to form larger structures
- Class structural uses inheritance to compose interfaces or implementations
- Object structural describes ways to compose objects to realize new functionality

Composite

- Component:
 - Declares the interface for the object in the composition and implements default behavior
 - Declares an interface for accessing and managing child components
- Composite:
 - Defines behavior for components having children
 - Stores children components
- Leaf
 - Represents leaf objects in the composition
- Client
 - Uses the component interface

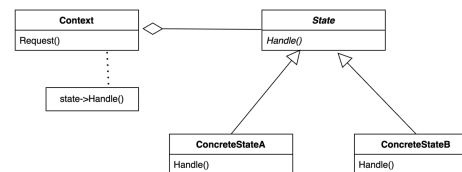


Behavioral

- Algorithms
- Assignment of responsibilities
- Patterns of communication

State


- Context
 - Defines the interface of interest to the clients
 - Maintains an instance of a ConcreteState that defines the current state
- State
 - Defines an interface for encapsulating the behavior associated with a particular state of the context
- ConcreteState
 - Implements the specific behavior for a state



The Maersk Mc-Kinney Møller Institute

Microservice Patterns

- Circuit Breaker
- Service Mesh Patterns

SDU  Torben Worm

February 2023

sdudk #sdudk 13


13

The Maersk Mc-Kinney Møller Institute

Monolith

- The software is a unified, self-contained application, i.e. it does not depend on other software
- Subsystems are tightly coupled with the rest of the application.
- Changes to a part of the application require a complete release of the entire software
- Issues
 - Difficult to roll-out features quickly because the whole system must be released
 - Difficult to change parts of the technologies because it's often used across many different parts of the system
 - Steep learning curve. The developer must know parts of the system irrelevant to the feature being developed
 - Scaling is typically only possible vertically

GUI/User interaction
Business logic
Persistence/Data base

SDU  Torben Worm

February 2023

sdudk #sdudk 14

14

The Maersk Mc-Kinney Møller Institute

Micro Services

→ A microservice architecture is an alternative to a monolith architecture

→ Various independent, loosely coupled units.

→ Independent business services with well-defined interfaces and operations.

→ Each service has its own independent context.

→ The services can interact with other services to perform the required task.

→ Advantages

- Agility
- Innovation
- Scalability
- Maintainability

→ Issues

- Requires stable network
- Latency
- Bandwidth
- Secure
- Topology
- ...

sdurdk #sdurdk

SDU Torben Worm

February 2023 15

15

The Maersk Mc-Kinney Møller Institute

Service Mesh

→ A service mesh is defined as a distributed system to address the microservice networking challenges in an integrated way.

→ Control plane

- The control plane is responsible for securing the mesh, facilitating service discovery, health checking, policy enforcement, and other similar operational concerns.

→ Data plane

- The data plane handles communication between services

sdurdk #sdurdk

[Sharma 2020]

SDU Torben Worm

February 2023 16

16

The Maersk Mc-Kinney Moller Institute

Service Mesh

→ **Problem**

- Cross-cutting concerns related to communication often lead to the duplication of code and a tangling division of responsibilities
- between infrastructure logic and business logic.

→ **Solution**

- Encapsulate cross-cutting concerns related to communication into independent modules that can be used by the different services.

→ Different ways of implementation

→ Service Mesh using a node agent

- Used in applications where the cost-effectiveness of resource use is one of the most important factors.

→ Service Mesh using a sidecar

- Used in applications where services are very different from each other and different cross-cutting configurations for almost every service would benefit the application.

sdudk
#sdudk

SDU Torben Worm

February 2023

17

17

The Maersk Mc-Kinney Moller Institute

Learning Objective

→ 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

→ Analyze existing software architectures and identify architectural problems

→ Ability to analyze and document software architectures and motivate the usage of adequate software architectures to obtain relevant quality attributes

sdudk
#sdudk

SDU Torben Worm

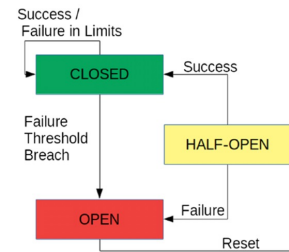
February 2023

18

18

Circuit Breaker

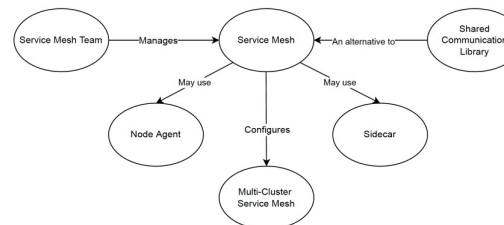
- Services responding slowly because of load.
- The Circuit Breaker pattern is aimed at handling this.
- Encapsulating the remote function call in an object that keeps track of failures.
- If there are more failures, then the service no longer invokes the external call; instead, it returns immediately with an error code.
- The circuit breaker keeps track of the connection in the following three states



- **Closed:** This is the state when external service calls are working fine without any reported failures. The circuit breaker keeps track of failures. This makes the circuit breaker more resilient against service performance blips. But when the number of failures exceeds a threshold, the breaker trips, and it goes into the Open state.
- **Open:** In this state, the circuit breaker returns an error without invoking the external call.
- **Half-Open:** After a configured time interval, the circuit breaker goes into a half-open state and validates if the external invocation still fails. If a call fails in this half-open state, the breaker is once again tripped. If it succeeds, the circuit breaker resets to the normal, closed state.

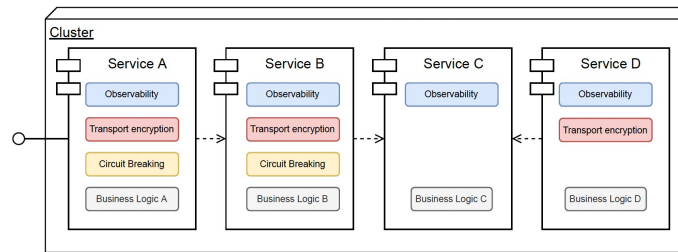
[Sharma 2020]

Service Mesh



[Maia 2022]

Without Service Mesh



21

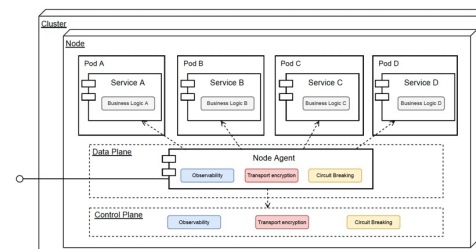
Node Agent

→ Problem

- Cross-cutting concerns often lead to the duplication of code and a bad division of responsibilities between infrastructure logic and business logic.

→ Solution

- Encapsulate cross-cutting concerns related to communication into a NodeAgent that acts as a proxy for all the services that are running on the same working node.



22

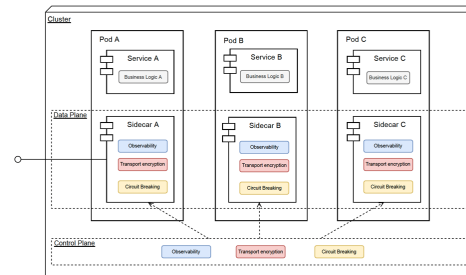
Sidecar Pattern

→ Problem

- Cross-cutting concerns often lead to the duplication of code and a bad division of responsibilities between infrastructure logic and business logic.

→ Solution

- Encapsulate cross-cutting concerns related to communication into a new module that runs alongside each of the services of the application.
- Instantiate a new container (the sidecar) for each of your services. These new containers will live in the same pods as the services.



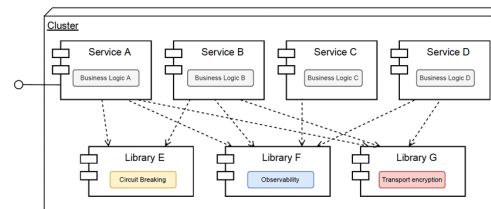
Shared Communication Library

→ Problem

- Cross-cutting concerns often lead to the duplication of code and a bad division of responsibilities between infrastructure logic and business logic.

→ Solution

- Encapsulate cross-cutting concerns related to communication into libraries that can be used by the different services.



Service Mesh Team

→ Problem

- The teams are not enough qualified to solve all the problems and bugs that appear related to the Service Mesh, also teams are dependent of others teams to take actions regarding the service mesh.

→ Solution

- Create a new team inside the company whose main goal is to guide actions related to the service mesh and remove the dependencies between teams.

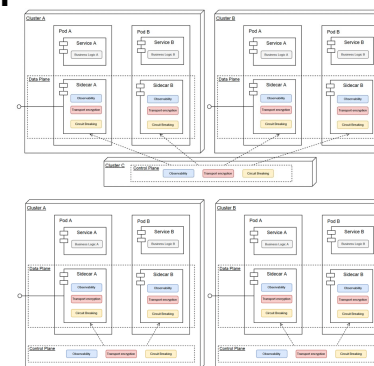
Multicluster Service Mesh


→ Problem

- In the case of a cluster or control plane degradation, user workloads may suffer an outage thus making the application not available

→ Solution


- Make the service mesh deploy one control plane in each cluster where the application is running.





The Maersk Mc-Kinney Møller Institute

Evaluating software architectures - 1

SDU  Sune Chung Jepsen

February 2023


#sdumk

27

The Maersk Mc-Kinney Møller Institute

Motivation

- Architectures allow or preclude nearly all of the system's quality attributes
- Therefore
- It is possible to evaluate architectural decisions with respect to their impact on those quality attributes
- Architecture evaluation is a cheap way to avoid disaster

SDU  Torben Worm

February 2023

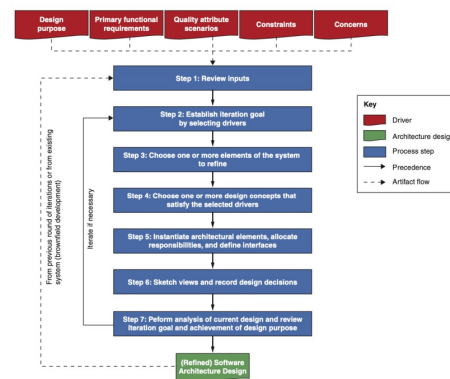
#sdumk

28

Architectural

→ To be architectural is to be the most abstract depiction of the system that enables reasoning about critical requirements and constrains all subsequent refinements

Steps and Artifacts of ADD




- Design purpose:
 - To create architectural prototype(S)
- Primary FR
 - For you to determine
 - Primary use cases
- QAS
 - See exercise – and add new if needed
- Constraints
- Concerns

The Maersk Mc-Kinney Møller Institute

When should the architecture be evaluated

- Early
 - Discovery review
- Late
 - Hold an evaluation when development teams start to make decisions that depend on the architecture and the cost of undoing those decisions outweigh the cost of holding the evaluation

SDU  Torben Worm

February 2023 31


sdudk #sdudk

31

The Maersk Mc-Kinney Møller Institute

Who's involved?

- Architecture team
 - Conducts evaluation
 - Perform analysis
- Stakeholders

SDU  Torben Worm

February 2023 32

sdudk #sdudk

32

The Maersk Mc-Kinney Møller Institute


What's the outcome?

→ Answers:

- Is this architecture suitable for the system for which it was designed?
- Which of two or more competing architectures is the most suitable one for the system at hand

→ Suitability

- The system that results from the architecture will meet its quality goals
- The system can be built using the resources at hand

SDU  Torben Worm

February 2023

33

#sdumk

33

The Maersk Mc-Kinney Møller Institute

Benefits and costs

→ Put stakeholders in the same room

→ Force an articulation of specific quality attributes


→ Prioritization of conflicting goals

→ Clear explication of the architecture

→ Improved quality of documentation

→ Opportunities for reuse

→ Improved architecture practices

SDU  Torben Worm

February 2023

34


#sdumk

34

The Maersk Mc-Kinney Møller Institute

ATAM

1. Present ATAM
2. Present Business Goals
3. Present Architecture
4. Identify Architectural Approaches
5. Generate QA Utility Tree
6. Analyse Architectural Approaches
7. Brainstorm and Prioritize Scenarios
8. Analyse Architectural Approaches
9. Present Results

SDU  Torben Worm

February 2023

35


sdudk
#sdudk

35

The Maersk Mc-Kinney Møller Institute

Mini – Architecture Analysis

1. Review essential use cases (from your assignment)
2. Review your quality attributes
3. Review your architecture
4. Construct (if you didn't in the previous exercise) QA Utility Tree
5. Analyze architectural approaches
6. Capture results
7. Iterate if necessary

SDU  Torben Worm

February 2023

36

sdudk
#sdudk

36

The Maersk Mc-Kinney Møller Institute


Midway Evaluation

→ Process:

- 1. During the next 10 minutes discuss e.g. the questions below
- 2. Write any items you would like to discuss in the Padlet (link in today's plan)
- 3. We discuss in class afterwards

→ Questions (examples):

- Good things?
- Could be better?
- Volume of reading material?
- Number of exercises?
- Number of assignments?
- Difficulty?
- ...

SDU  Torben Worm

February 2023

sdumk
#sdumk

37