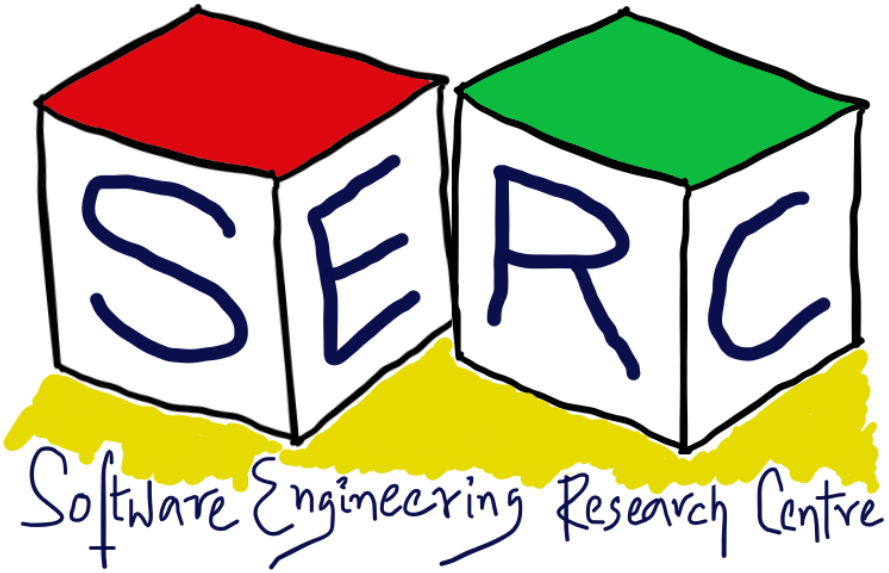
EDA and Concluding Thoughts

CS6.401 Software Engineering



Dr. Karthik Vaidhyanthan

karthik.vaidhyanathan@iiit.ac.in

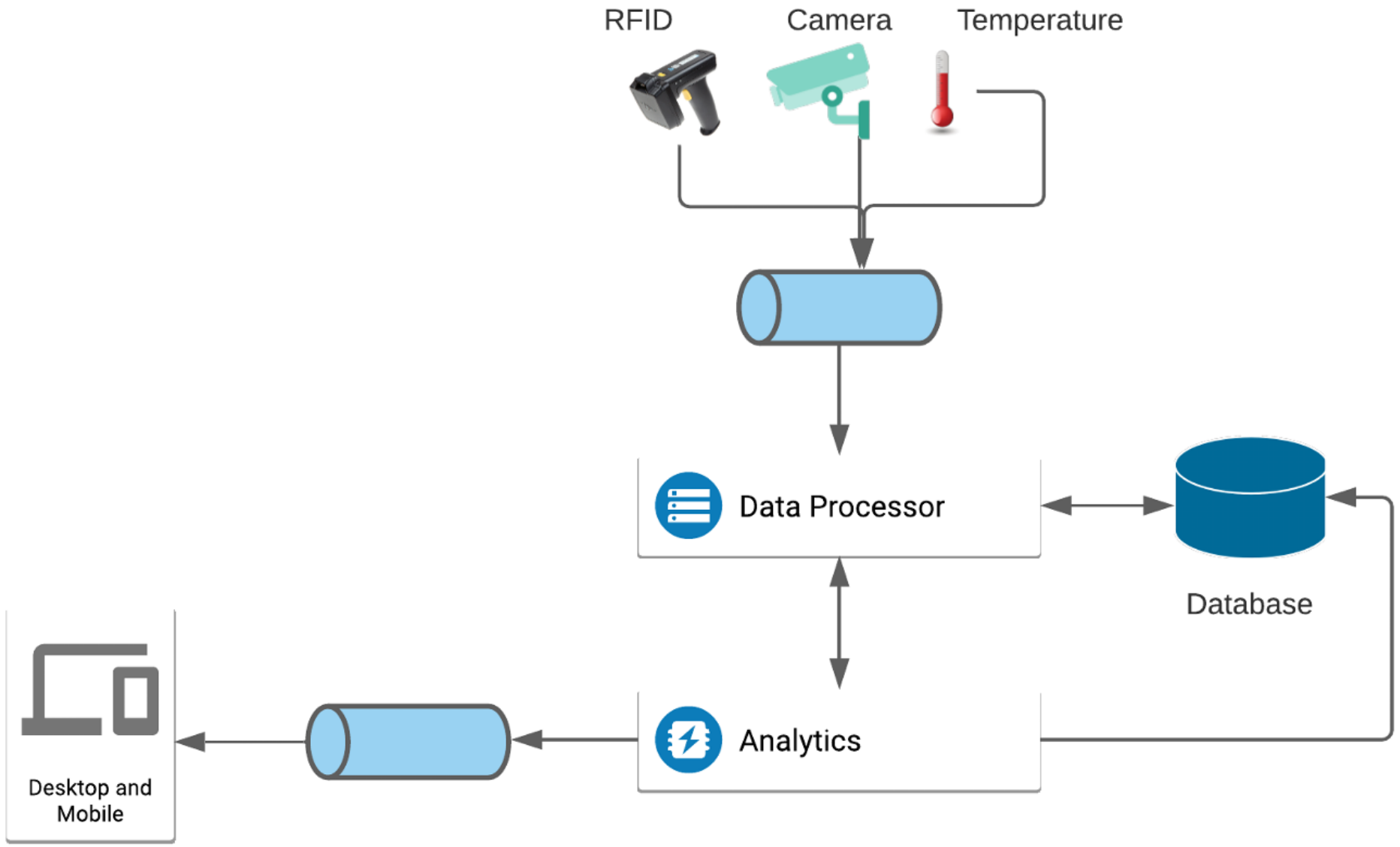
https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Acknowledgements

The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge -- Karthik Vaidhyanathan

Enters Event Driven Architecture



Event Driven Architecture: An Overview

- Independent components asynchronously emit and receive events communicated over event buses
- Produce, detect and consume events
- Highly decoupled components – Minimal amount of coupling (topics, queue names, etc.)

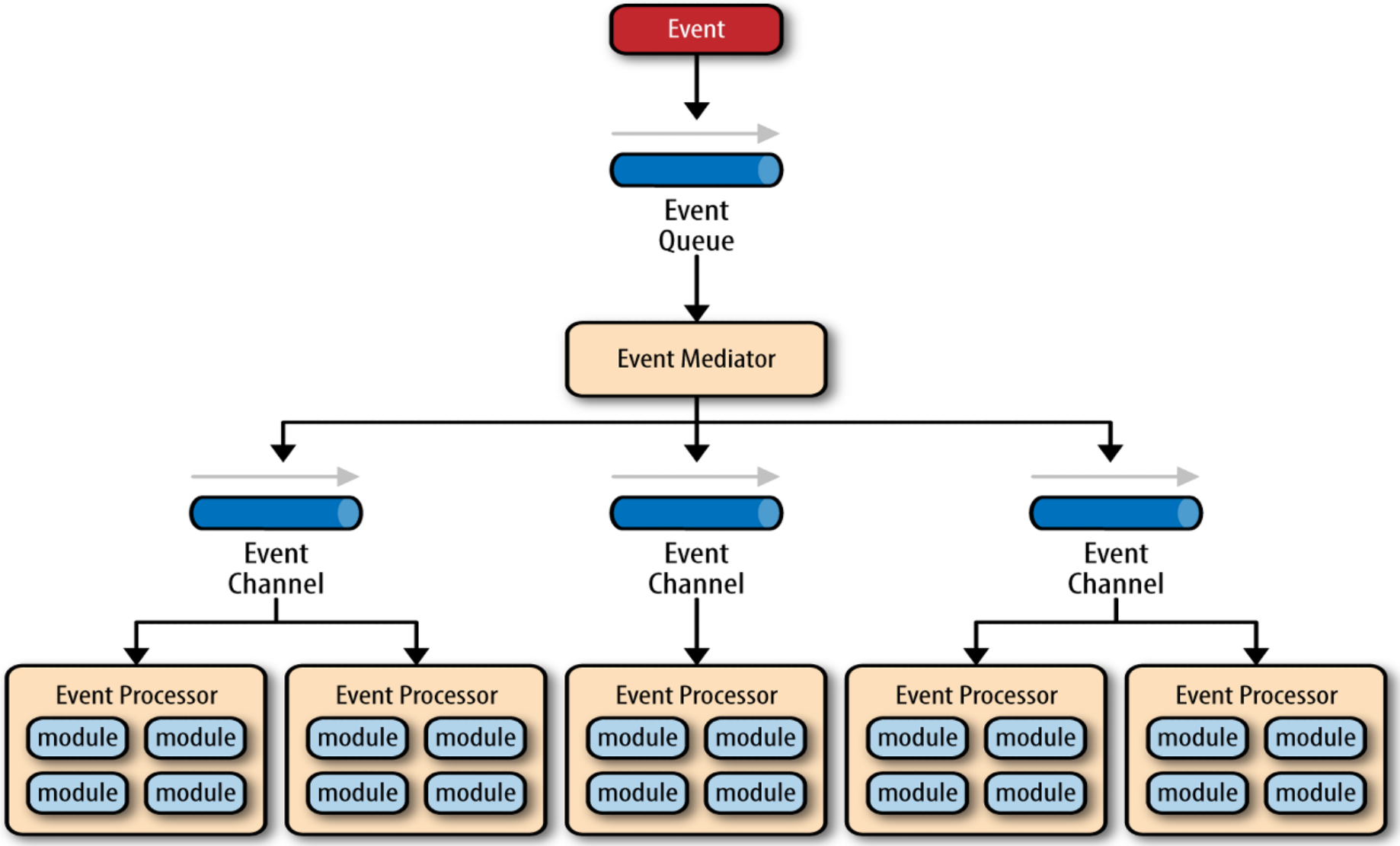
Design elements

- Components: concurrent event generators and event consumers
- Connectors: event bus (may be more than one)
- Data: events

Topology

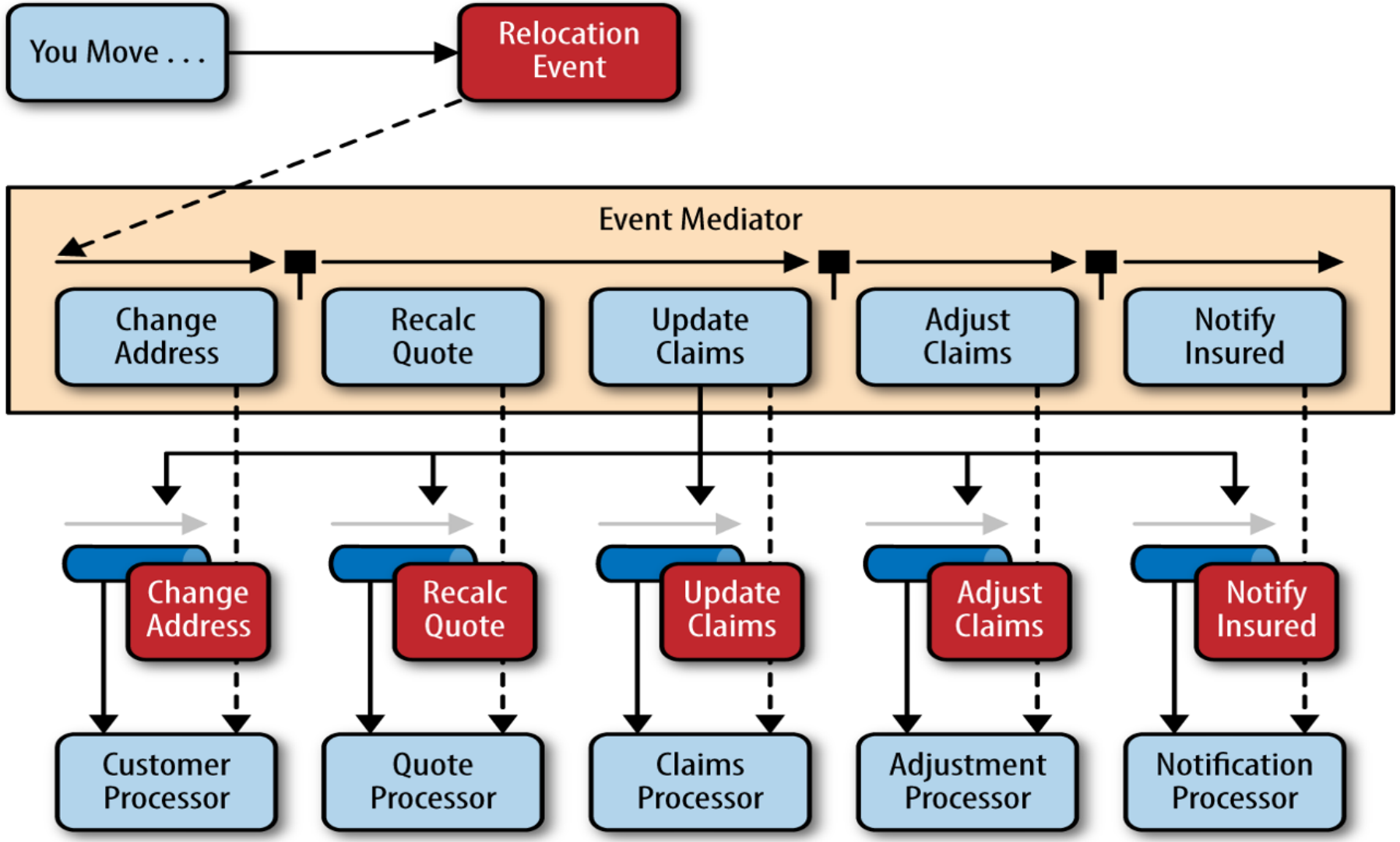
Communication via the event bus or link only (Mediator or Broker)

Event Driven Architecture: Mediator



https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html

Event Driven Architecture: Mediator



https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html

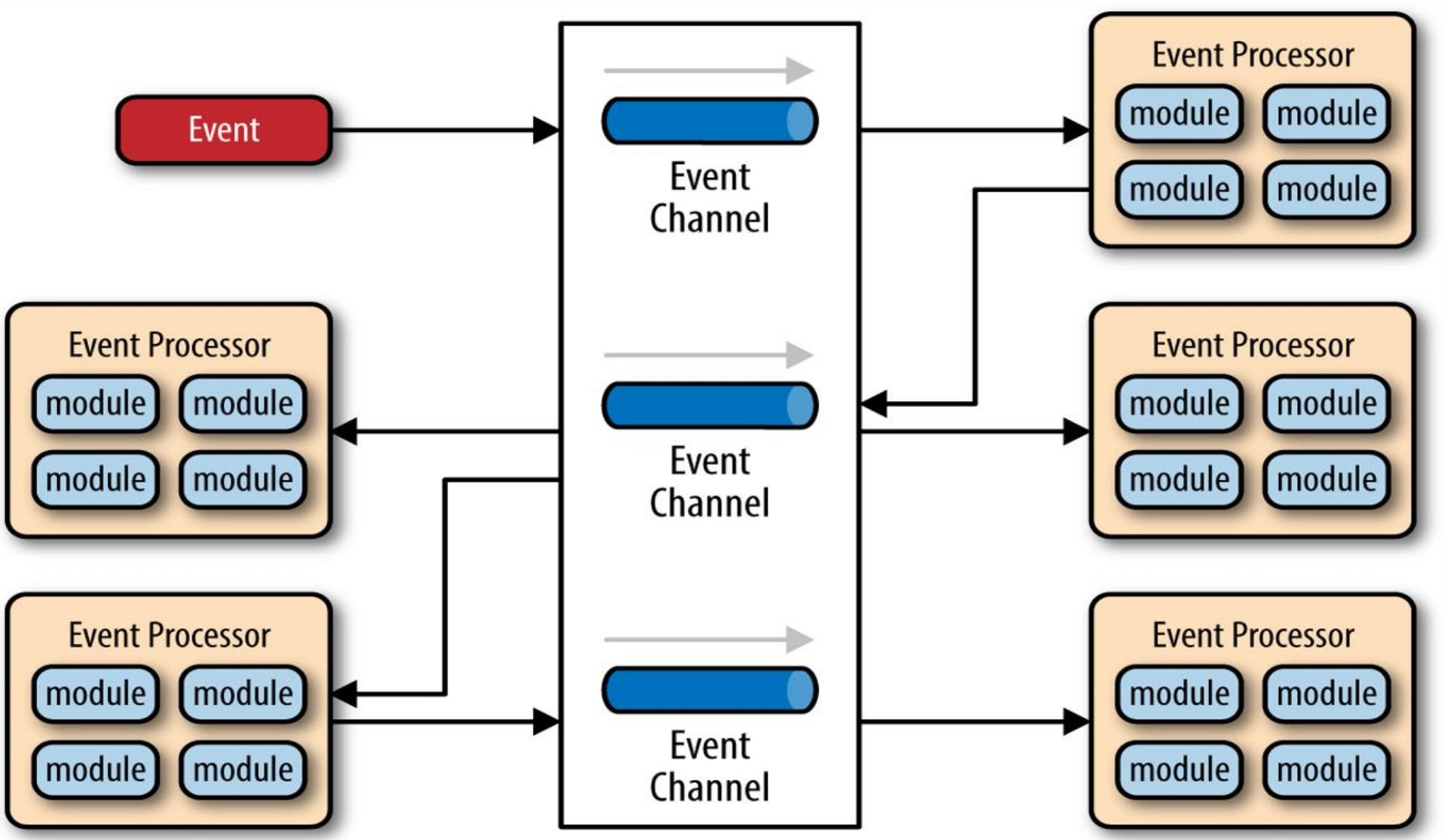
Mediator Topology: An Overview

Similar to the Orchestration in traditional SOA
Two key events – **Initial and Processing event**

Four main types of components:
1. **Event queue –** Responsible to transfer events to event mediator
2. **Event Mediator –** Orchestrates the processing of events to accomplish the overall functionality
3. **Event Channel** - Topics or queues to which events are ingested by mediator (eg: Kafka topic)
4. **Event Processor** - Implements the business logic
    1. Can be fine grained or Coarse grained)
    2. Advice: keep it to one functionality

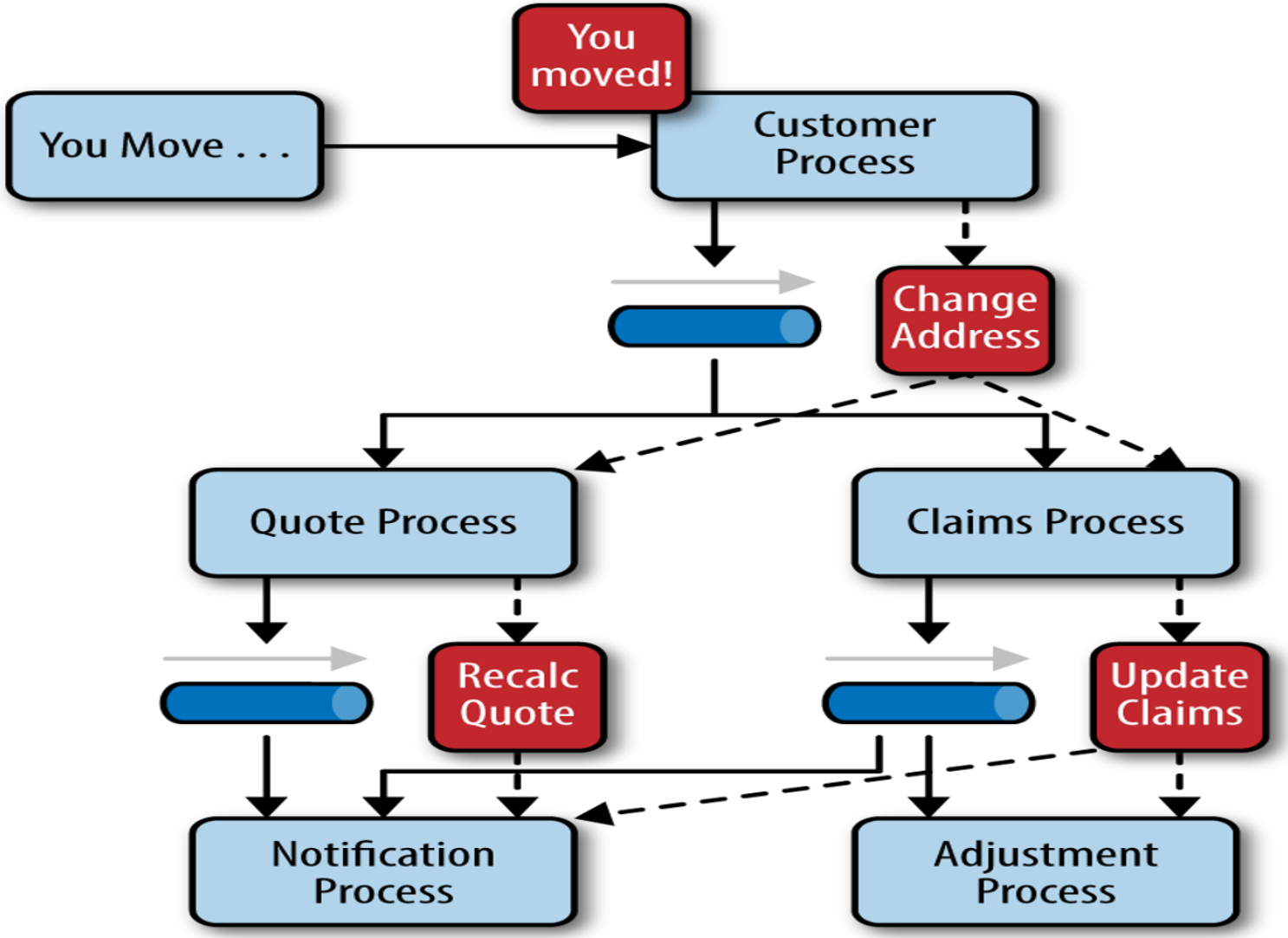
https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html

Event Driven Architecture: Broker



https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html

Event Driven Architecture: Broker



https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html

Broker Topology: An Overview

- Similar to the Choreography in traditional SOA
- Two main types of components:

1. Broker – Consists of all the event channels for event processing. Can be topics or queues
2. Event Processor – Responsible for processing the event and sending a notification to the event channels

https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html

How to Decide?

Advantages

1. High performance
2. High Scalability
3. Ease of Deployment
4. Ease of modifications/Evolved easily

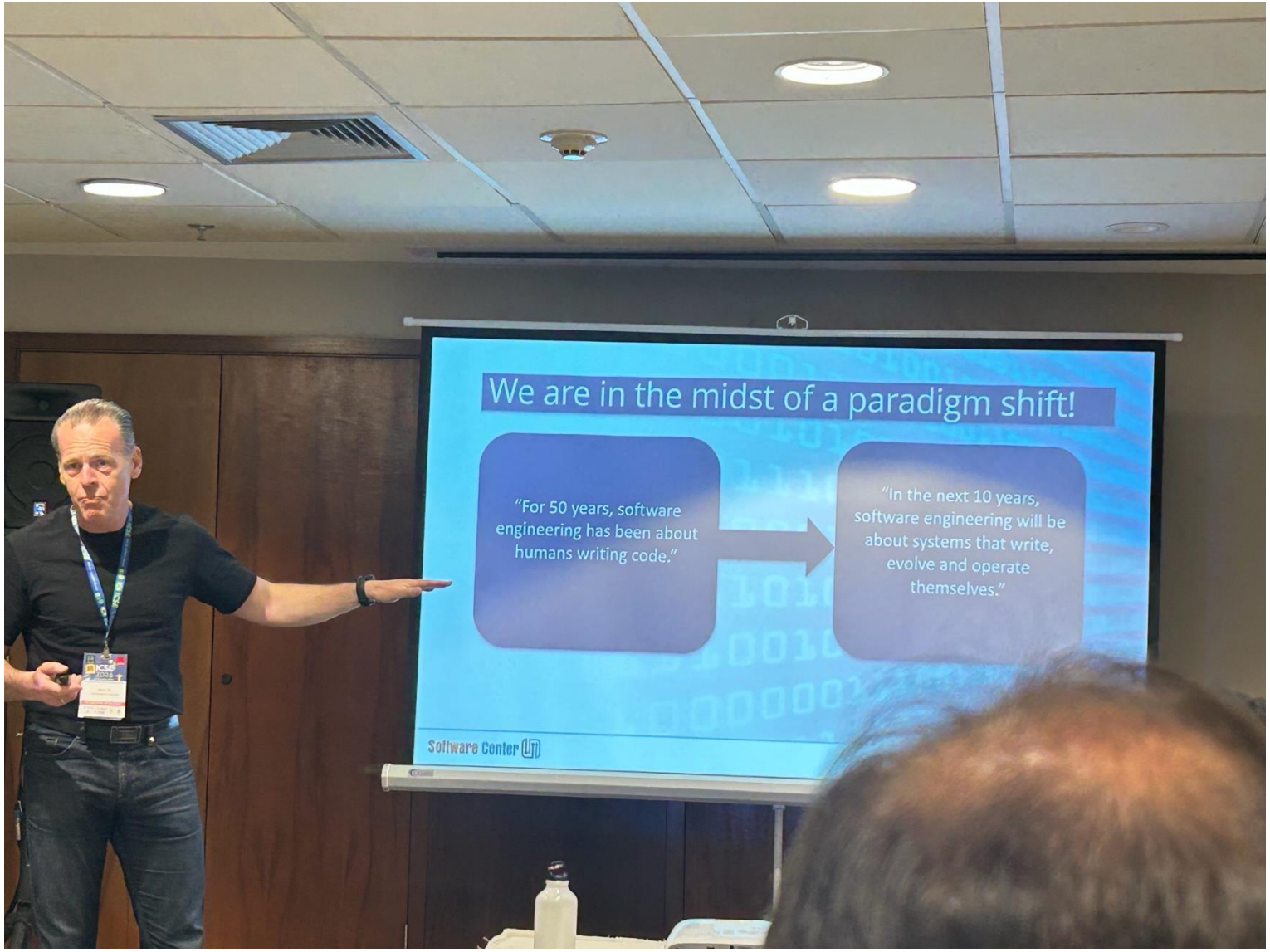
Disadvantages

- Remote process availability – Liveliness of a consumer
- Lack of responsiveness
- Broker or mediator failures
- Testing can be tedious
- Development can be complex

https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html

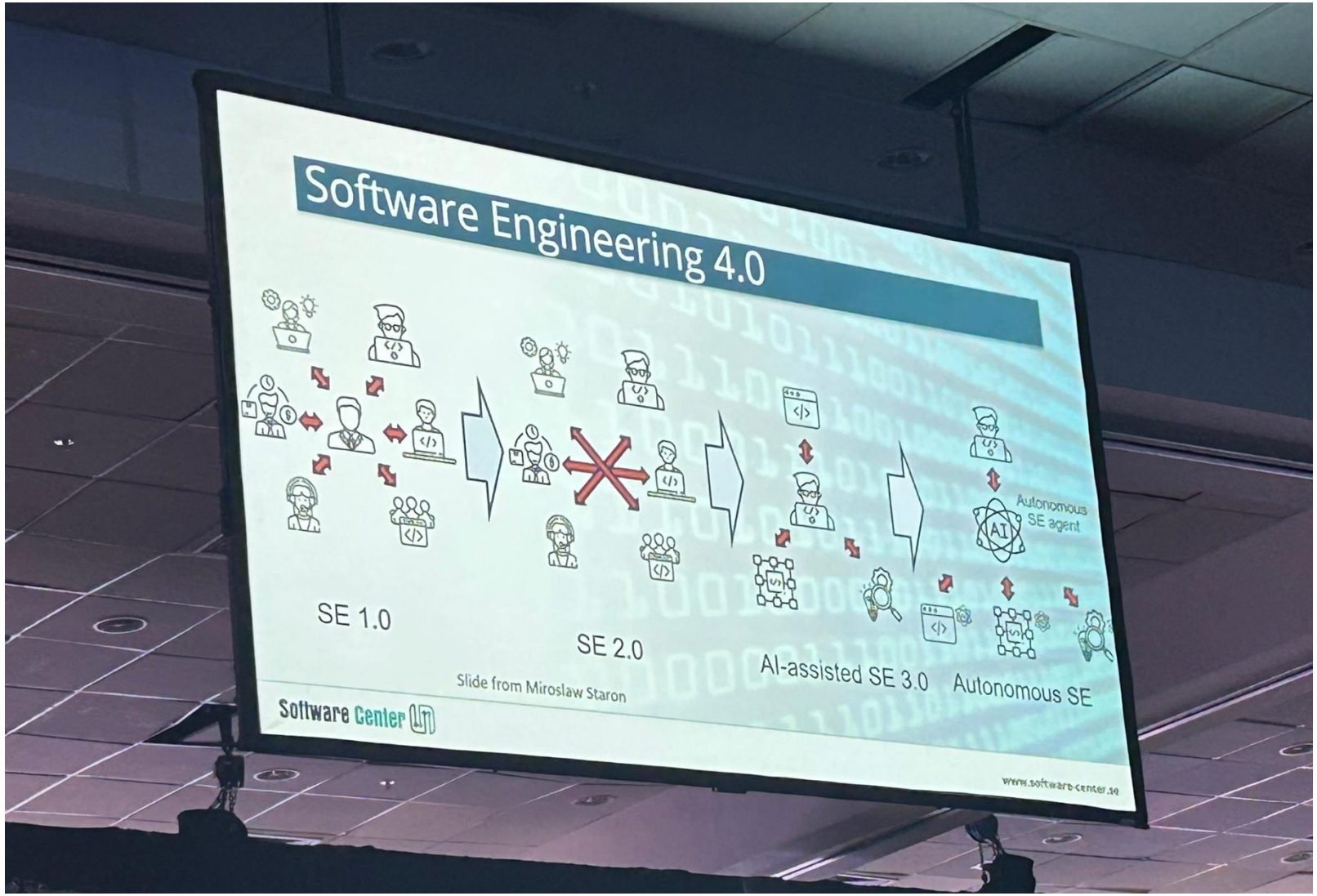
Software Engineering!

Some Aspects of Software Engineering is Changing!!



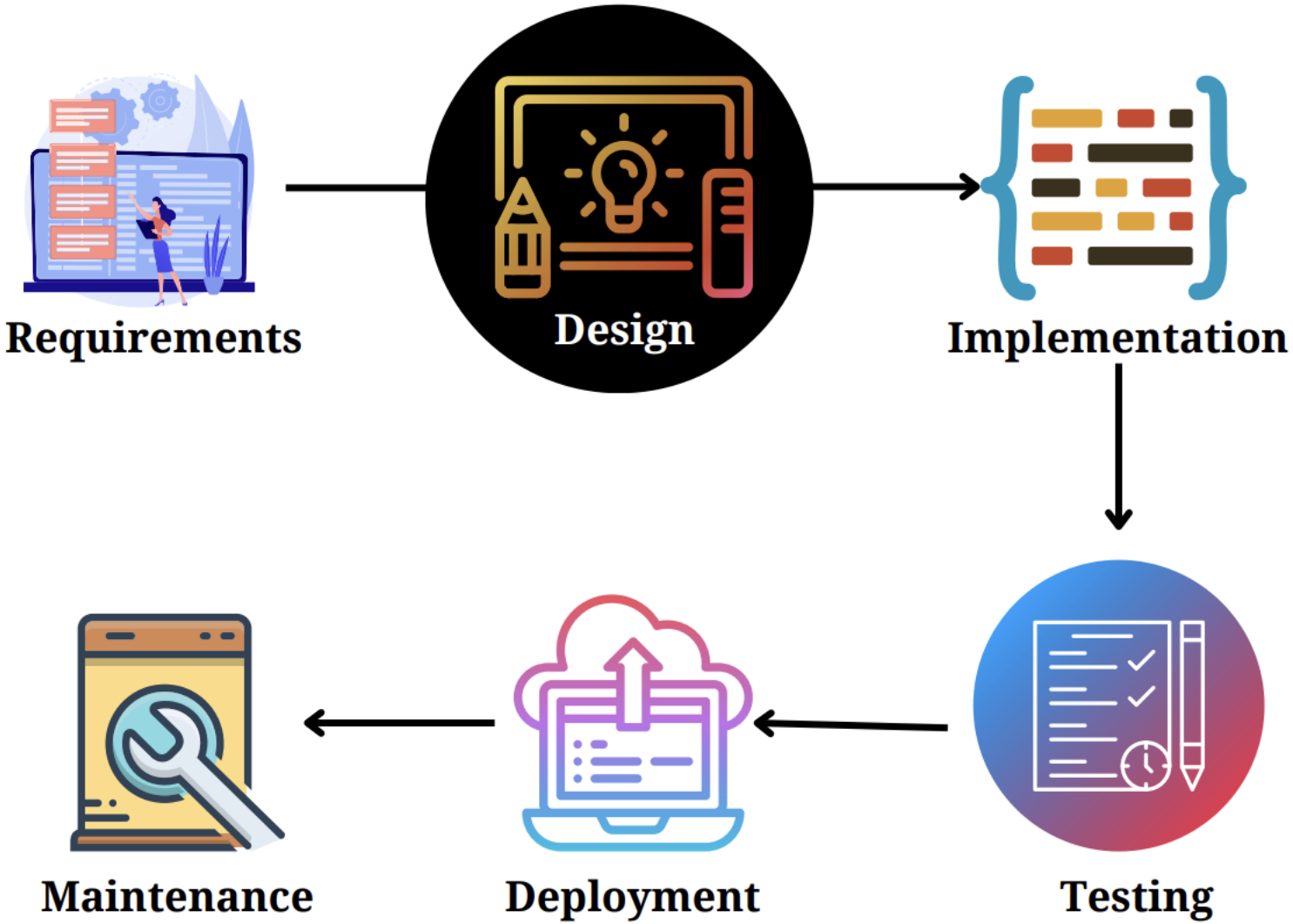
Jan Bosch, Keynote, CAIN@ICSE 2026

Some Aspects of Software Engineering is Changing!!

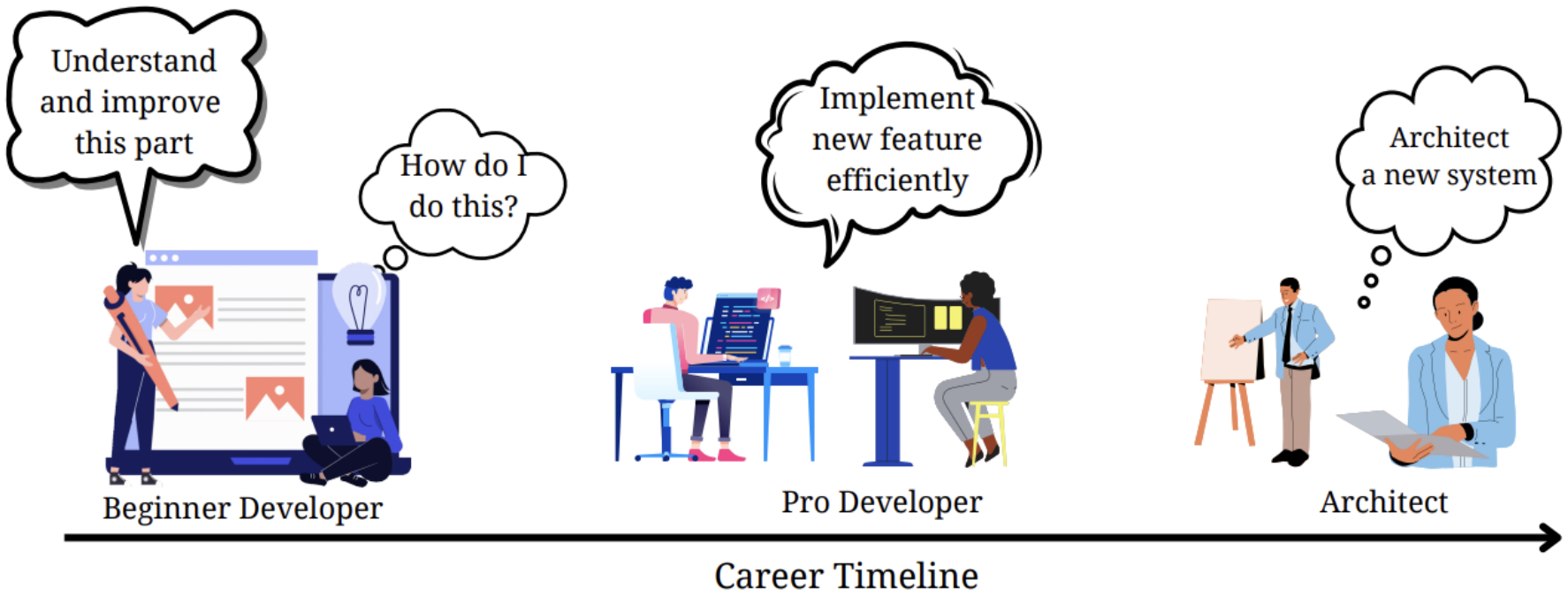


Jan Bosch, Keynote, ICSE 2026

Software Development Lifecycle



What we learned so far?



This Course

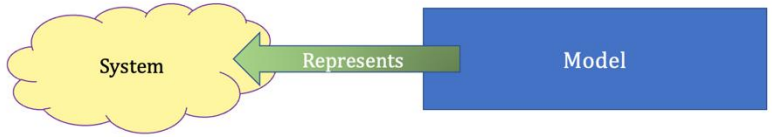
Modeling	Design Principles	Architectural Framework
Refactoring	Design Patterns	Architectural Patterns

Quick Recap

Modeling and Refactoring

So what is a software model?

A simplified or partial representation of a real system, defined in order to accomplish a task or to reach an agreement.



Mapping: A model is always a mapping of some real system

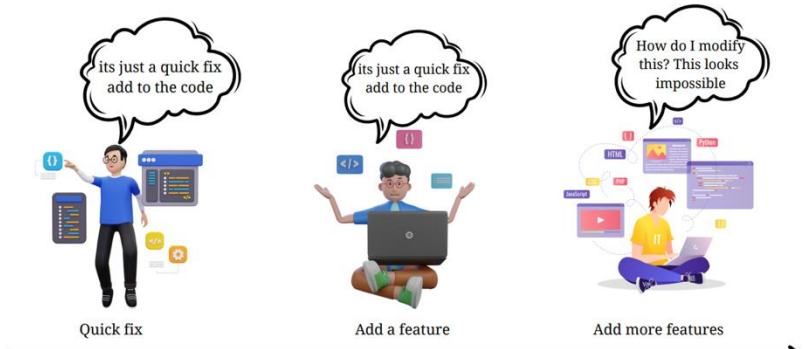
Reduction: A model reflects only relevant set of properties of original system

Pragmatism: A model needs to be usable in place of the actual system with respect to some purpose

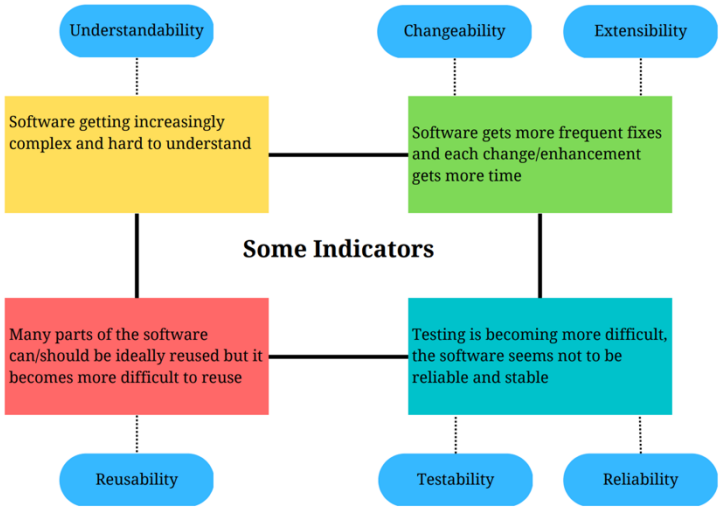
Technical Debt - Definition

Technical debt is the **debt that accrues** when you knowingly or unknowingly make **wrong or non-optimal design decisions**

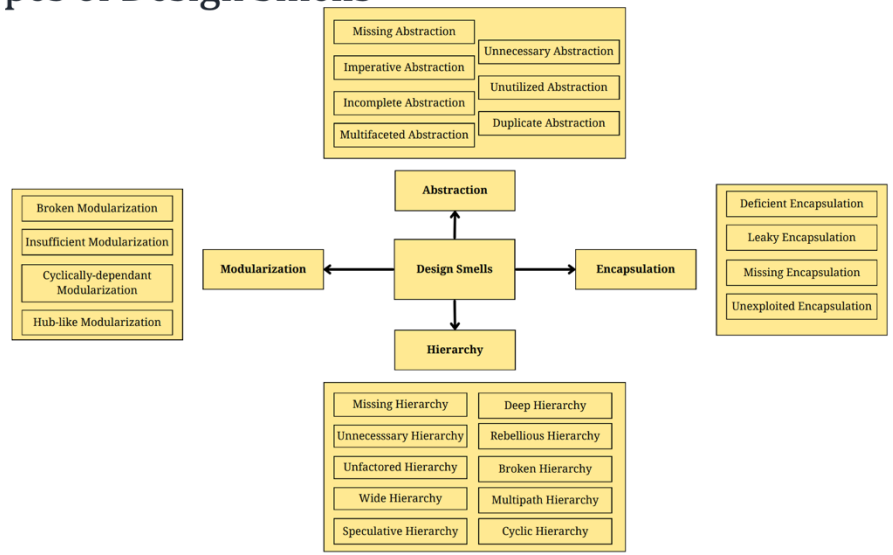
Metaphor coined by *Ward Cunningham*, 1992



Software Quality as an Indicator



Types of Design Smells



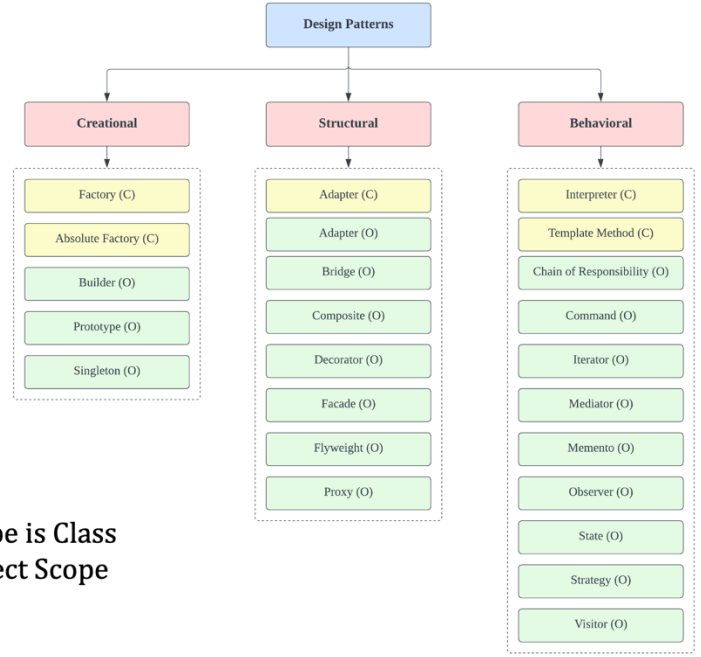
Design Principles and Design Patterns

Design Patterns

*Each Pattern describes a problem which **occurs over and over again** in our **environment** and then **describes the core of the solution** to that problem, in such a way that you can **use this solution a million times over**, without ever doing it the same way twice* -- Christopher Alexander

Patterns captures {Context, Problem, Solution}

What are some of the patterns you can think of? 🤔



C – Scope is Class
O – Object Scope

Program to Interface Not Implementation

- One of the most important OO Design Principles
- "Program to interface" refers to the idea of ensuring loose coupling
  - Does not only mean the "Interface"?
- Very useful when lot of changes are expected
- Create an interface, define methods -> create classes that implements them
- Allows external objects to easily communicate
- Maintainability and flexibility increases

Favor Object Composition over Class Inheritance

- Two most common techniques: Inheritance and Composition
- Class inheritance: White-box reuse
  - Internals of parent class are visible to child class
  - Defined statically at compile time
  - Sub class can override methods of parent class
- Inheritance is not always the go to solution - "breaks encapsulation"
- Composition: Black-box reuse
  - Objects acquiring references to other objects
  - Defined dynamically at run time
  - Encapsulation is not broken – Objects are accessed through interfaces
  - Get what is needed by assembling and not by creating

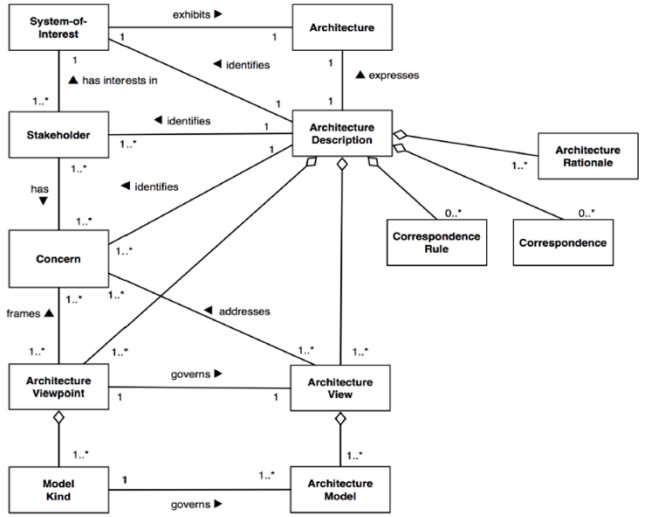
Software Architecture Framework and Patterns

Software Architecture Definitions

- Garlan and Shaw, '93:
  Architecture for a specific system may be captured as "**a collection of** computational **components** - or simply components - together with a description of the interactions between these components - the **connectors** "
- Bass et al.:

"The software architecture of a program or computing system is the **structure or structures** of the system, which comprise **software elements**, the **externally visible properties** of those elements, and the **relationships** between them."

Architecture Description

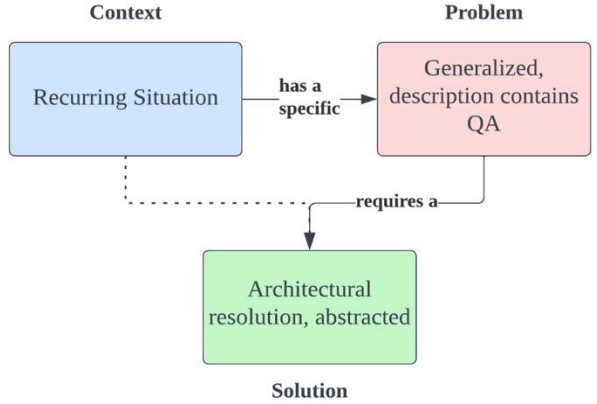


ISO/IEC/IEEE 42010, Systems and Software Engineering – Architecture Description

What about Architectural Tactics?

"Characterization of architectural decisions that are used to achieve a desired quality attribute response"

Architectural Patterns



Pattern documentation template: {*context, problem, solution*}

Research Avenues

Active Research Areas

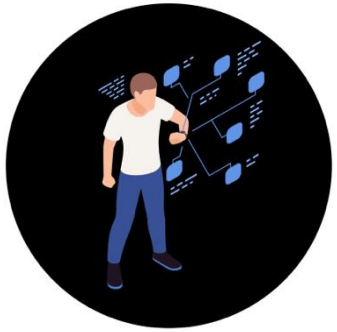
- Software Testing
- Software Architecture
- Software Maintenance and Evolution
- Model Driven Engineering
- Software Verification
- Software Sustainability
- Software Processes
- Human Aspects
- Many More … (RE, PL,…)

Research Areas in SERC

Virtual Labs

Gamification

VR and AR



IoT

HCI

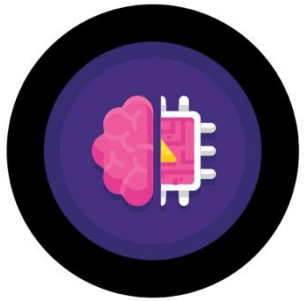
Software Quality

Programming Languages

Formal Methods

Software Analytics

Self-adaptive Systems



SE and ML

Software Sustainability

Our Explorations in GenAI&SE

Requirements

Design

Development

Testing

Deployment

Maintenance

ICSA 2026

LLMs for migrating from Monoliths to Microservices [Collaboration]

AGENT 2026

Enroute to Agentic Architecture Knowledge Management

ICSE 2026

Architecting self-coding Information Systems [Collaboration]

ICSA 2025

Automated generation of serverless functions

ICSA 2024, EASE 2026

Evaluation of Context required for Generating ADRs [Collaboration]

ICSA 2025

Architecture Views from Source code

CHANGE Framework for AgentOPS

CAIN 2026

Self-adaptation approach for QoS routing in Small Language Model based Systems

SEAMS 2026

Multi-agentic reasoning for self-adaptation

SEAMS 2026

Autonomous CloudOps (Multi-agent)

CAIN 2025

Self-adaptation using Large Language Models

ICSA 2024

Generating IaaC [Collaboration]

ISEC 2026

Automated development and deployment of services

ICSA-C 2025

Agent Assesment Framework

AGENT 2026

Study across Agentic AI Frameworks

IEEE Software 2026

Agentic Approach to Architecture Knowledge Management

AGENT 2026

Agents for generating Microservices [Collaboration]

ICSA 2026

Energy Efficiency of code generated by LLMs [Collaboration]

ICSE 2026

Function calling Using Small Language Models [Collaboration]

EASE 2025

SWEnergy: Energy efficiency of Agentic Frameworks for code resolution

AGENT 2026

Some Conferences

ASE 2026

What Next!

1. Topics in Software Engineering Course
2. PhD/Software researcher/MS in Software Engineering (Erasmus, SE4GD, EDISS, …)
3. SDE/Data scientist/ML Engineer/Data engineer…..
4. Junior architect/software architect/consultant…
5. Research Software Engineer

Course Logistics Management

Thanks to the wonderful team!

Chandrasekar S

Adarsh Sharma

Akhila Matathammal

Anshi Gandhi

Chirag Damija

Divyansh Pandey

Gargie Tambe

Kritin Madireddy

Shaunak Biswas

Tejas Cavale

Vyakhya Gupta

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

SERC
Software Engineering Research Center

Best Wishes!!

Thank You

Course website: karthikv1392.github.io/cs6401_se

Email: karthik.vaidhyanathan@iiit.ac.in
Web: https://karthikvaidhyanathan.com
LinkedIn: /In/karthikv1992