



# Messaging Introduction

Systems Integration

PBA Softwareudvikling/BSc Software Development

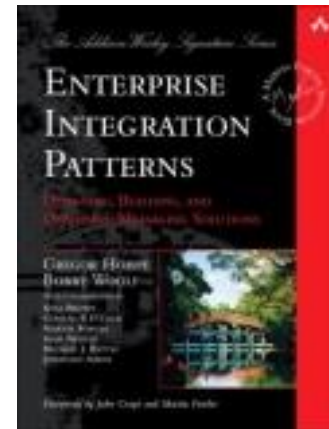
Tine Marbjerg

Fall 2018

# Today's Topics

---

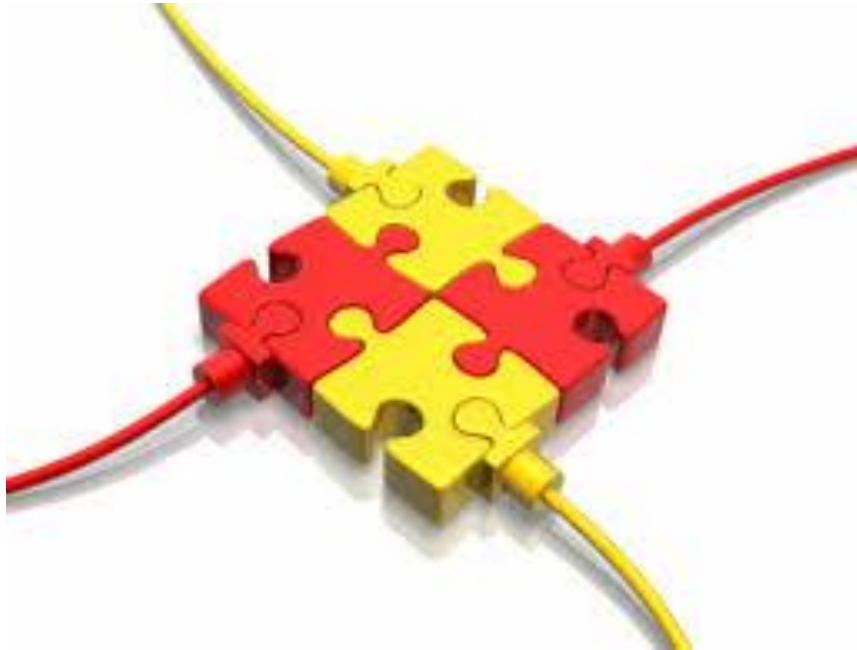
- Introduction to the SI course
- Solving Integration Problems Using Patterns (EIP chap. 1)
- Integration Styles (EIP chap. 2)
- Messaging Systems (EIP chap. 3)



# System Integration Course

---

*System integration is the task of making different applications work together to produce a unified set of functionality.*



# Semester Plan

---

- Course Part 1
  - System Integration Patterns (focus on Messaging technology)
- Course Part 2
  - Network Protocols and Integration Techniques and Technologies
- Project 1
  - Loan Broker (mandatory)
- Project 2
  - Blockchain (mandatory)

# Exam

---

- The exam starts with a ten minutes group presentation about the Loan Broker System.
- Subsequently, discussion about both projects, technologies, design decisions, their impact, etc.
- The exam will last in total 10 minutes per student, but at least 30 minutes pr. group.

# Course Learning Objectives

---

- To enable the student to work with integration of systems:
  - integrate existing systems
  - integrate existing systems and new systems
  - develop new systems that support future integration
- The course has both a theoretical and practical approach with 3 assessment levels (see [curriculum](#))
  - Knowledge
  - Skills
  - Competences

# Book Organization

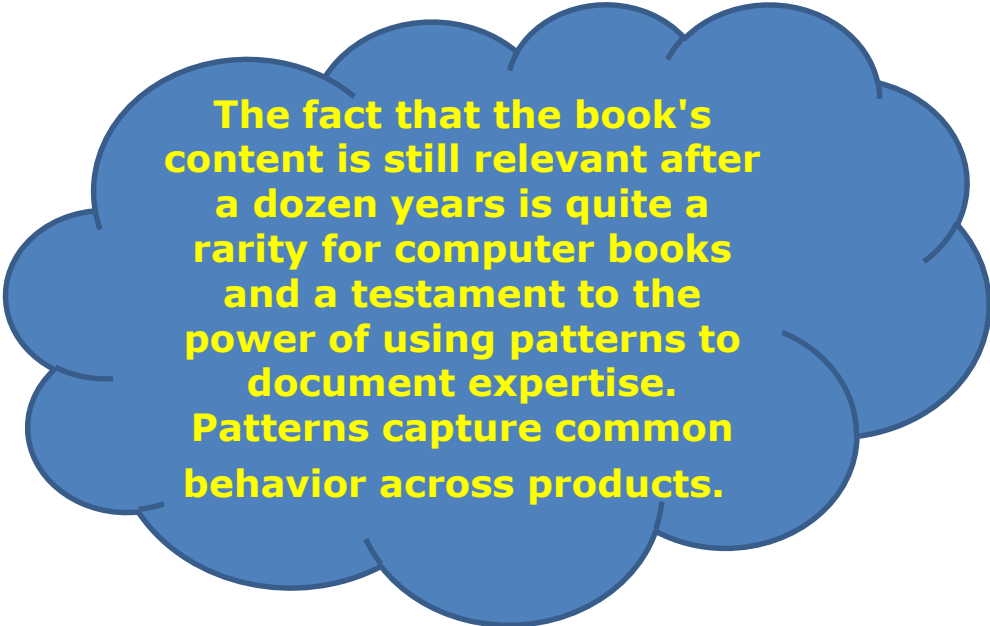
---

- **Visual and verbal language** to describe integration solutions
  - Not a precise specification language
  - No vendor jargon
- **Patterns** describe message components and concepts
  - Each pattern describes considerations and trade-offs
  - Combines patterns to describe larger solutions
- The patterns apply to a **variety of programming languages and platforms**, e.g.
  - JMS, MSMQ, BizTalk, TIBCO ([modern tech examples online](#))
  - A few larger examples in chap. 6+9 (=templates for your project)

# Old Book is Still Relevant 😊

---

Source: [IEEE 2016](#)



The fact that the book's content is still relevant after a dozen years is quite a rarity for computer books and a testament to the power of using patterns to document expertise. Patterns capture common behavior across products.



I found many of our patterns in the recently released [Google Cloud Pub/Sub service](#)

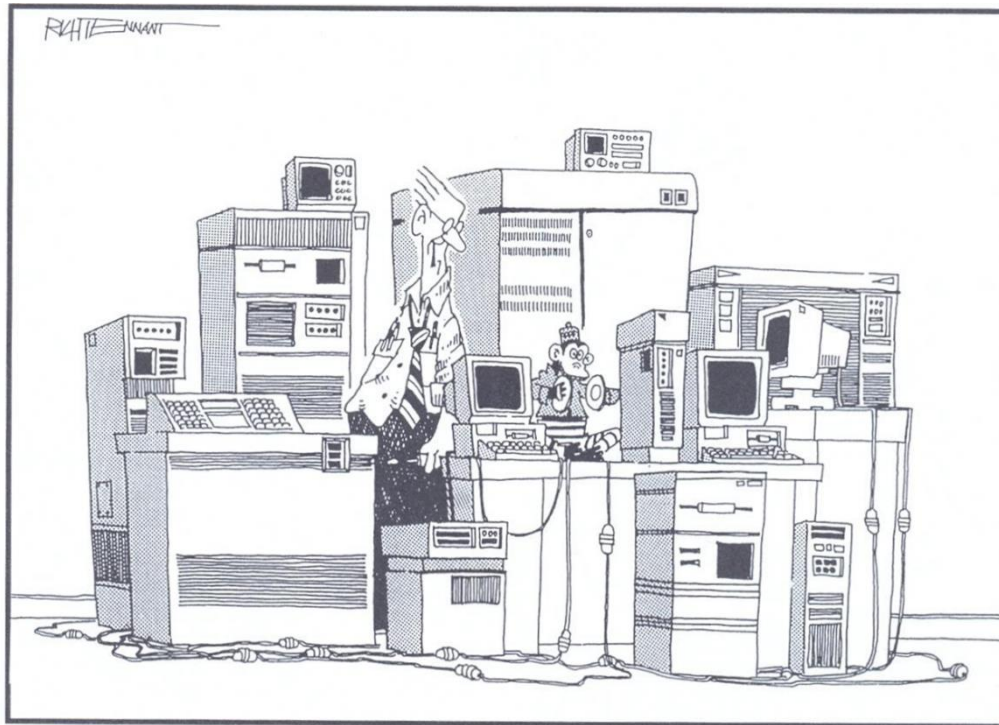


# System Integration Challenges

---

What are the challenges of integrating different apps?

- Discuss with student sitting next to you and make list of challenges



"Now, just when the heck did I integrate THAT into the system?"

# System Integration Challenges (EIP intro)

---

- Applications must be connected, but they might ...
  - Vary from custom developed in house to purchased from third-party vendors
  - Not be designed with integration in mind and therefore difficult to change
  - Run on multiple computers (multiple platforms, and maybe geographically different places)
  - Run outside of the enterprise by business partners or customers.

# How to Integrate Apps

---

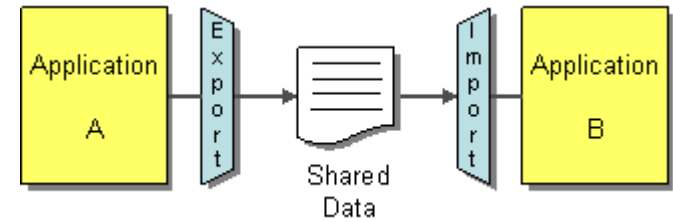
- How many integration options can you come up with?
  - Describe each approach shortly
  - Include pros/cons for each

# Application Integration Styles (EIP chapter 2)

---

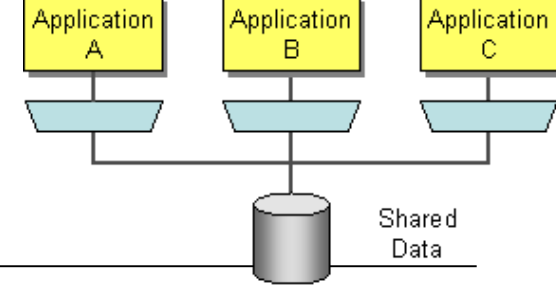
- 4 options
  - File Transfer (43)
  - Shared Database (47)
  - Remote Procedure Invocation (50)
  - Messaging (53)
- Pattern order above reflects an increasing order of sophistication, but also increasing complexity
- Brief descriptions of the patterns in the EIP book can be found at <http://www.eaipatterns.com/>

# File Transfer



- Each application
  - produces files of shared data for others to consume,
  - consumes files that others have produced
- Data oriented

Pro	Con
Simple technology	File processing is expensive
Universal storage mechanism	Stale data due to infrequent updates (out of sync)
Decoupled from applications	No data format enforcement

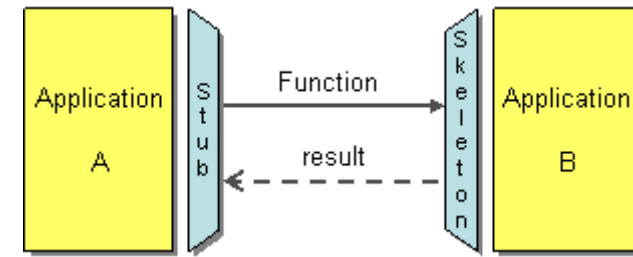


# Shared Database

- Applications store the data they wish to share in a common database
- Data oriented

Pro	Con
Data available more quickly	Performance bottleneck (many read/update on the same data)
Transaction management	Deadlock
Enforce data format	Tight coupling to database (unencapsulated data structure)
	Unified schema is difficult to design

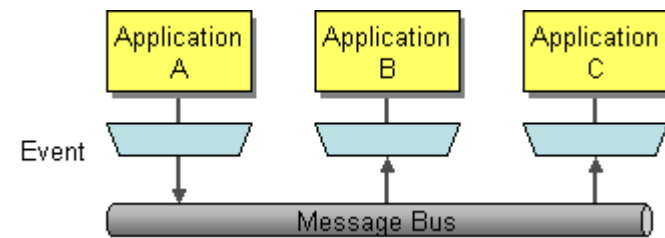
# Remote Procedure Invocation



- Applications
  - expose some of their procedures to be invoked remotely,
  - invoke those exposed procedures to run behavior and exchange data
- Functionality oriented

Pro	Con
Style familiar to programmers	Remote calls are slow
Encapsulate data	Remote calls are unreliable
Can deal with semantic dissonance (multiple interfaces to same data)	Fairly tight coupling

# Messaging



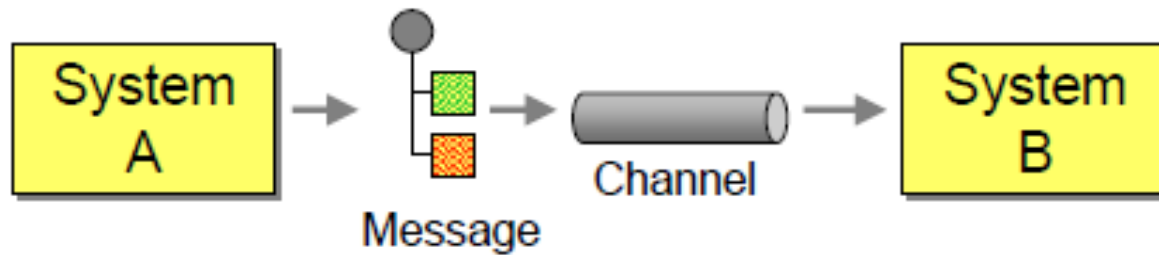
- Application
  - connect to a common messaging system,
  - exchange data and invoke behavior using messages
- Data and functionality oriented

Pro	Con
Decoupled	A bit slower
Asynchronous	Asynchrony has higher learning curve
Reliable	Testing and debugging is harder
No data format enforcement	



# Basic Messaging Concepts (EIP chapter 1)

---



- Channels are not part of the systems (applications)
- Channels are asynchronous & reliable
- Systems don't know each other (loose coupling)
- Data is exchanged in self-contained messages

# Channels & Queues are the same

---

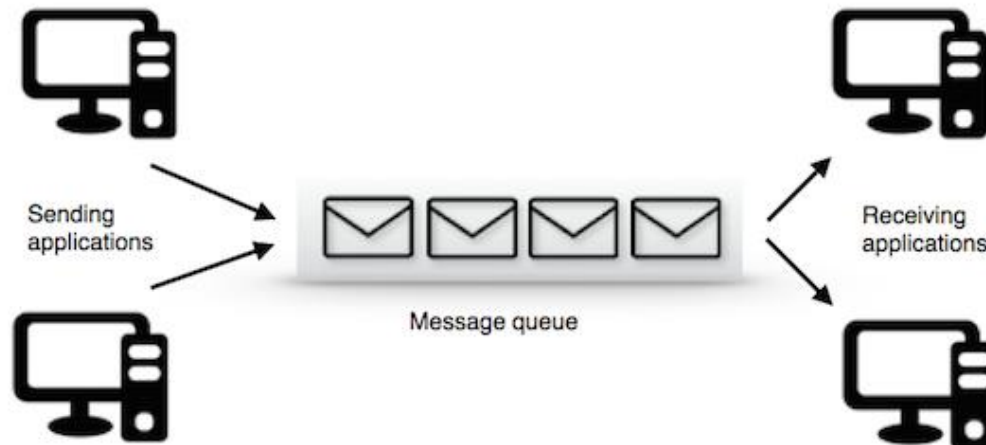
- The message queue provides a temporary message storage



# Decoupling and Scalability

---

- Messaging is not limited to physically distributed systems.
- It can be used as a programming model to define the interaction between different parts of an application.
- Decoupling is usually easier to maintain, extend and debug
- Each part of application can be scaled independently



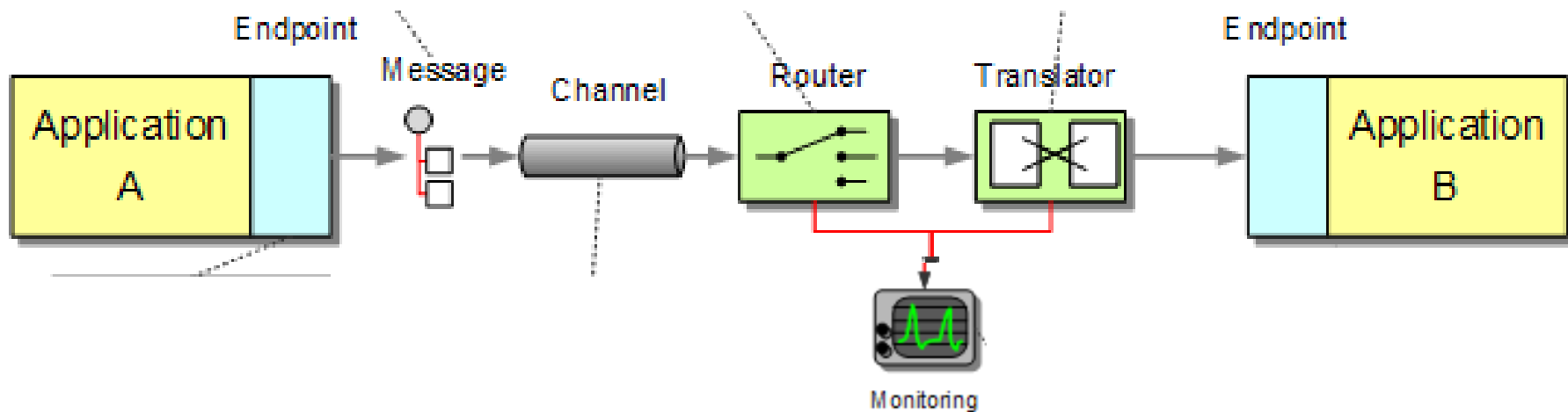
# Message Queue Use Cases

---

- Discuss with student sitting next to you and identify use case(s) where message queues could be useful

# Basic Messaging Architecture (EIP chap. 3)

- Basic messaging concepts



# Explanation of Elements in prev. Figure

---

- Message Bus
  - reliable and secure communication infrastructure across enterprise network
- Adapter
  - Makes app interface available to other applications
- Transformation function
  - Translates application data into common data format
- Business process
  - Model of a series of actions, each action carried out by different applications
  - Manages long-running business transactions
- Repository
  - Business rules and business object definitions (meta data)
- Portal
  - Aggregates information from multiple systems
- Monitoring
  - Centralized real-time monitoring. Can trigger restart or fail-over action