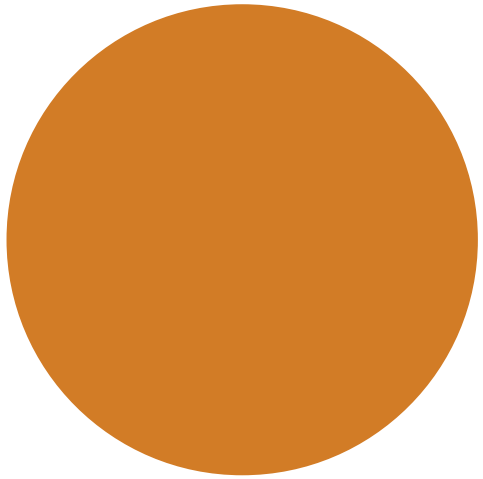
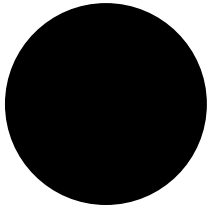




Enterprise Application Integration

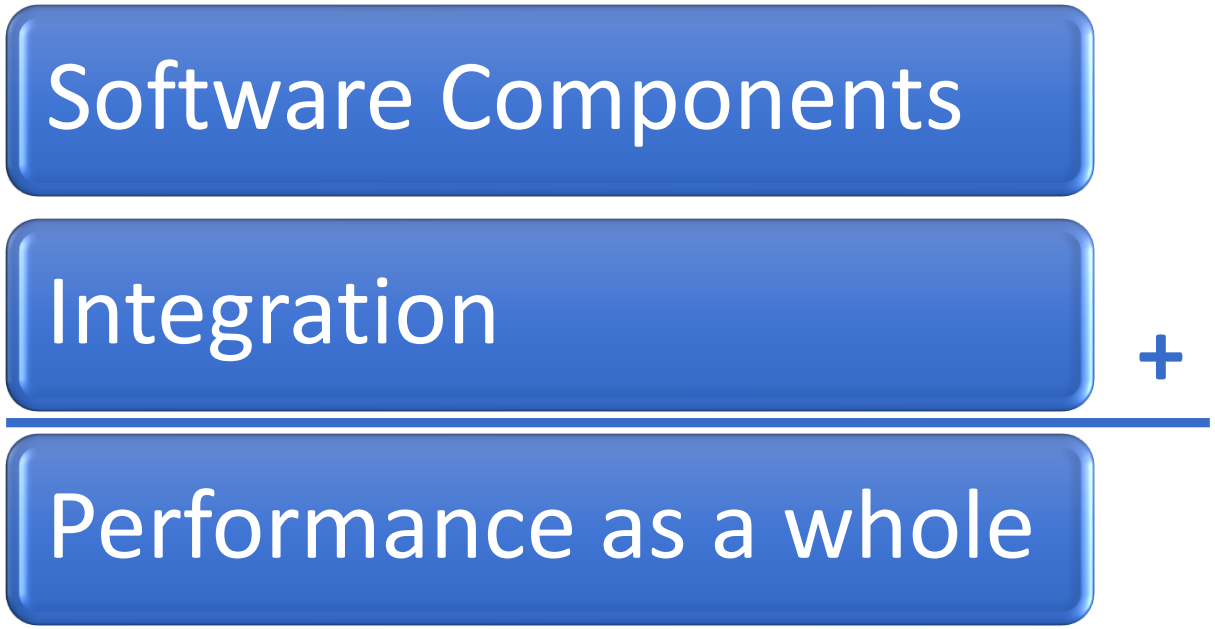
Lesson 1
EAI Introduction

Hanzehogeschool Groningen
University of Applied Sciences

Contents

- EAI Overview
- Key Business goals for EAI
- Why EAI?
- Micro and Macro Software Architecture and EAI
- EAI related quality requirements for software components and integration
  (design principles and integration styles)

EAI - overview

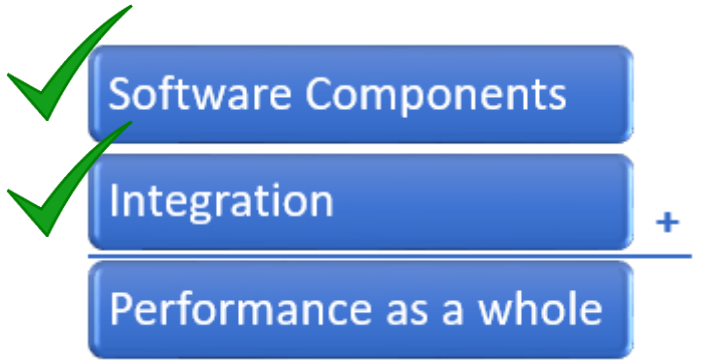


Definition of integration:

the act of bringing together the parts of a whole

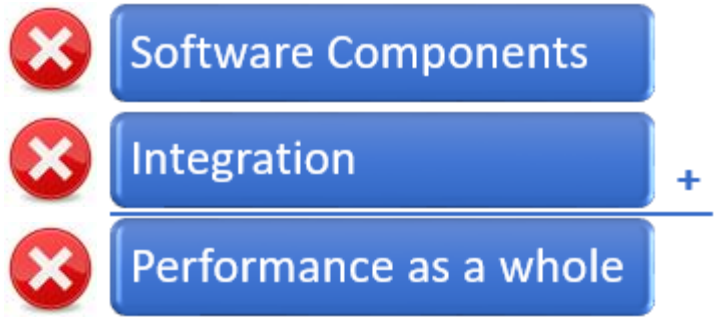
EAI - overview

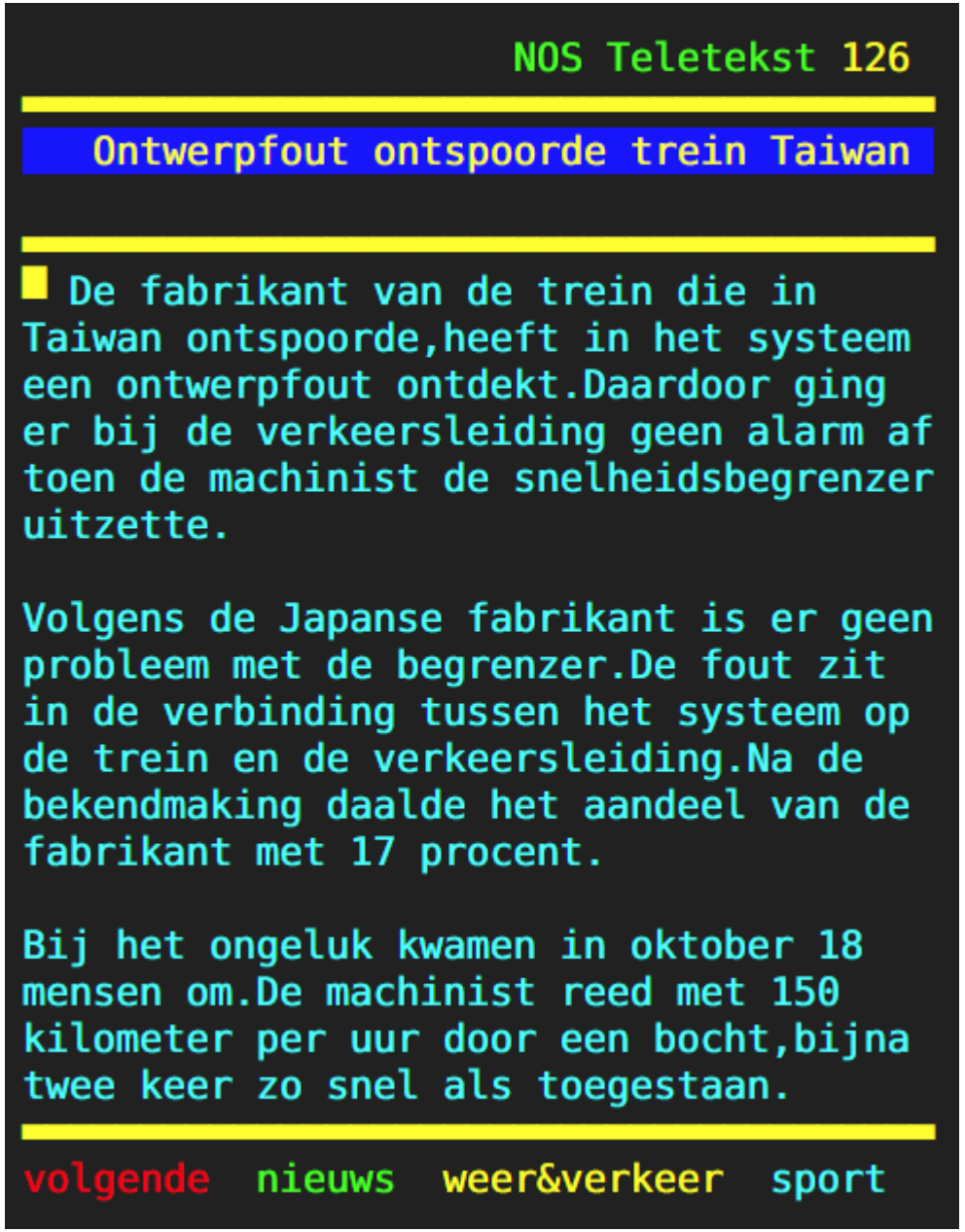
- Prerequisites for high performance of the Distributed Information Systems Landscape:
  - High Quality Component design
  - High Quality Integration



EAI - overview

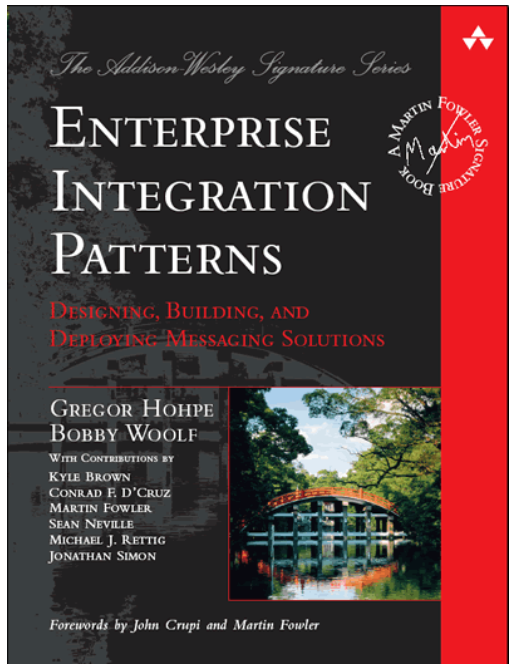
- Poor component design and poor integration pose serious problems for businesses.





https://nos.nl/teletekst 2 november 2018

Mind map for this EAI course



Theory on software design principles & patterns

Knowledge of Enterprise Applications

Theory on EAI messaging patterns

Software Components

Integration

Performance as a whole

+

Theory on EAI messaging technologies / standards

Theory on Integration Styles

Theory on IT architectures (SOA)

Key Business goal for EAI => Agility which enables short Time to Market

Knowledge on Middleware products from suppliers

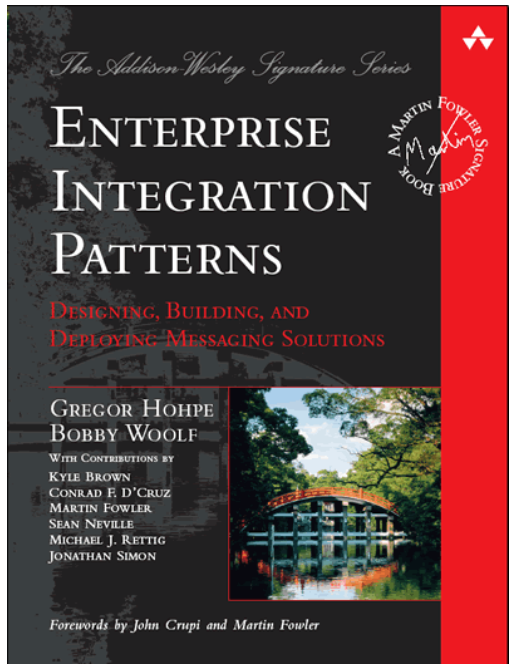
Key Business goal for EAI

Companies:

- face competition
- should be flexible and respond in time to changing circumstances / business needs

The IT systems landscape should be Agile to ensure a short Time to Market

Mind map for this EAI course



Theory on software design principles & patterns

Theory on Integration Styles

Theory on IT architectures (SOA)

Knowledge of Enterprise Applications

Software Components

Integration

Performance as a whole

+

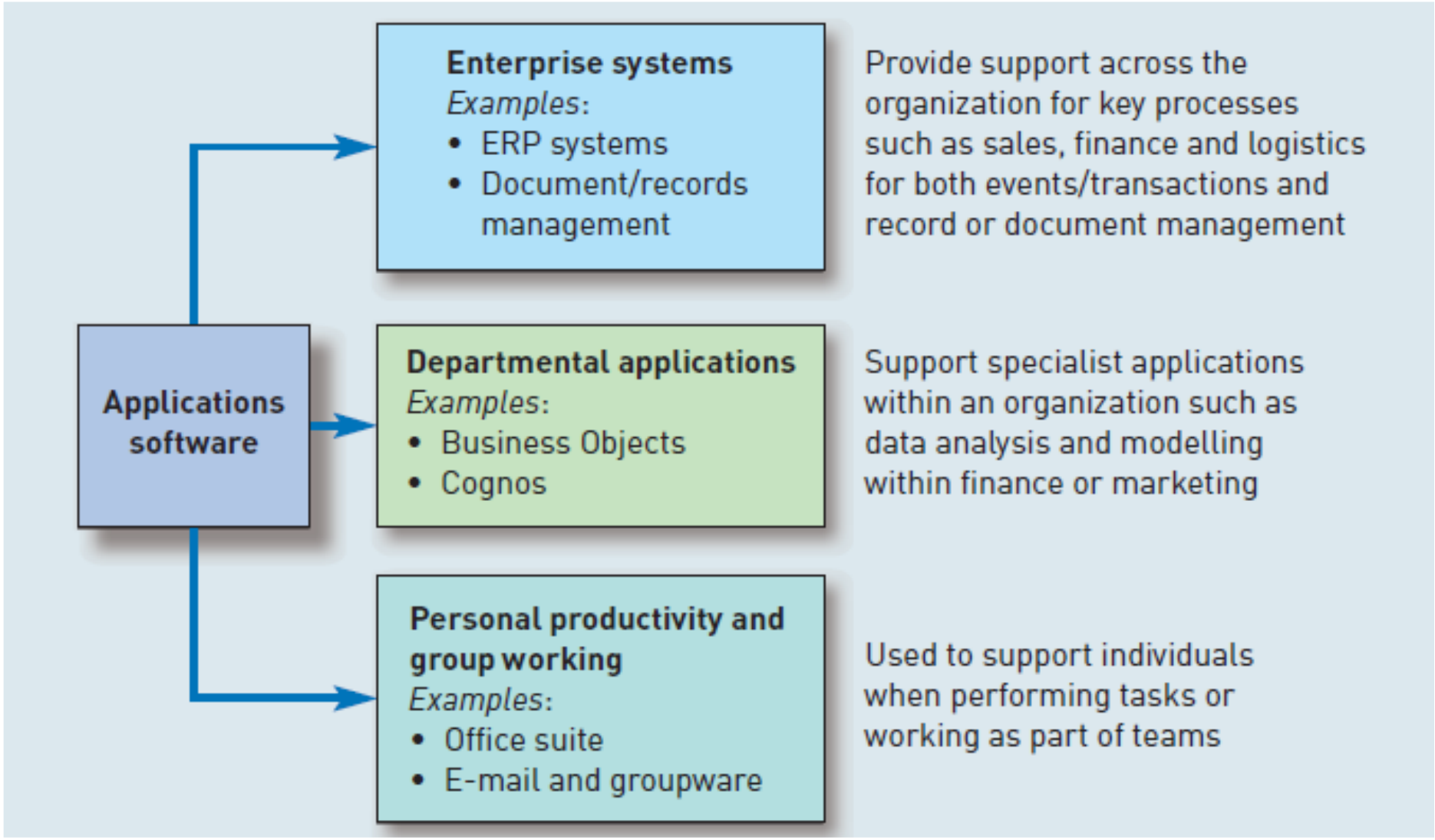
Key Business goal for EAI => Agility which enables short Time to Market

Theory on EAI messaging patterns

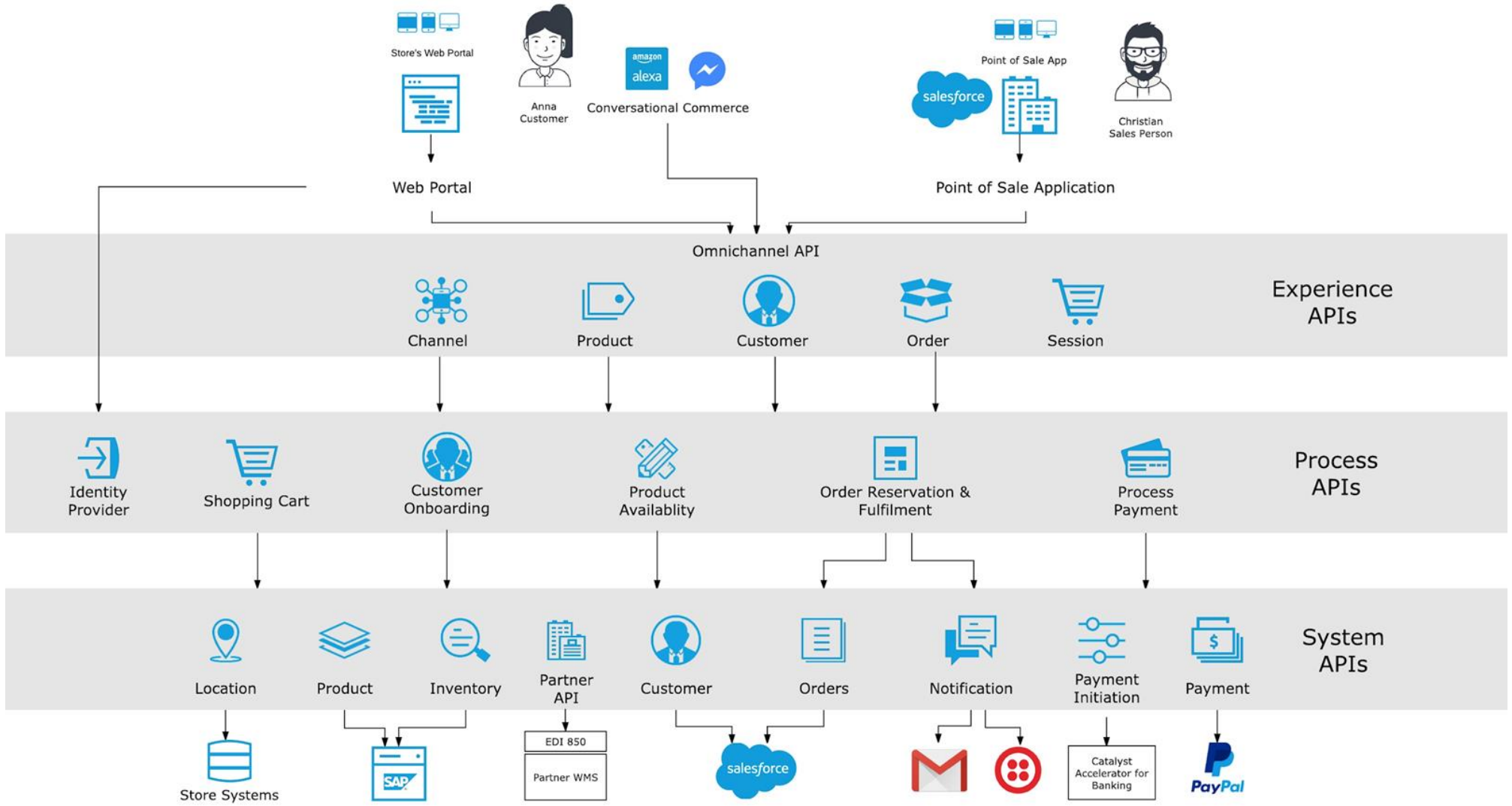
Theory on EAI messaging technologies / standards

Knowledge on Middleware products from suppliers

Main categories of business application software



Example of integrated business applications /distributed systems



EAI quality requirements for integration

Some key quality attributes of application integration

Consistency

Is replicated data (eventually) consistent?

Timeliness

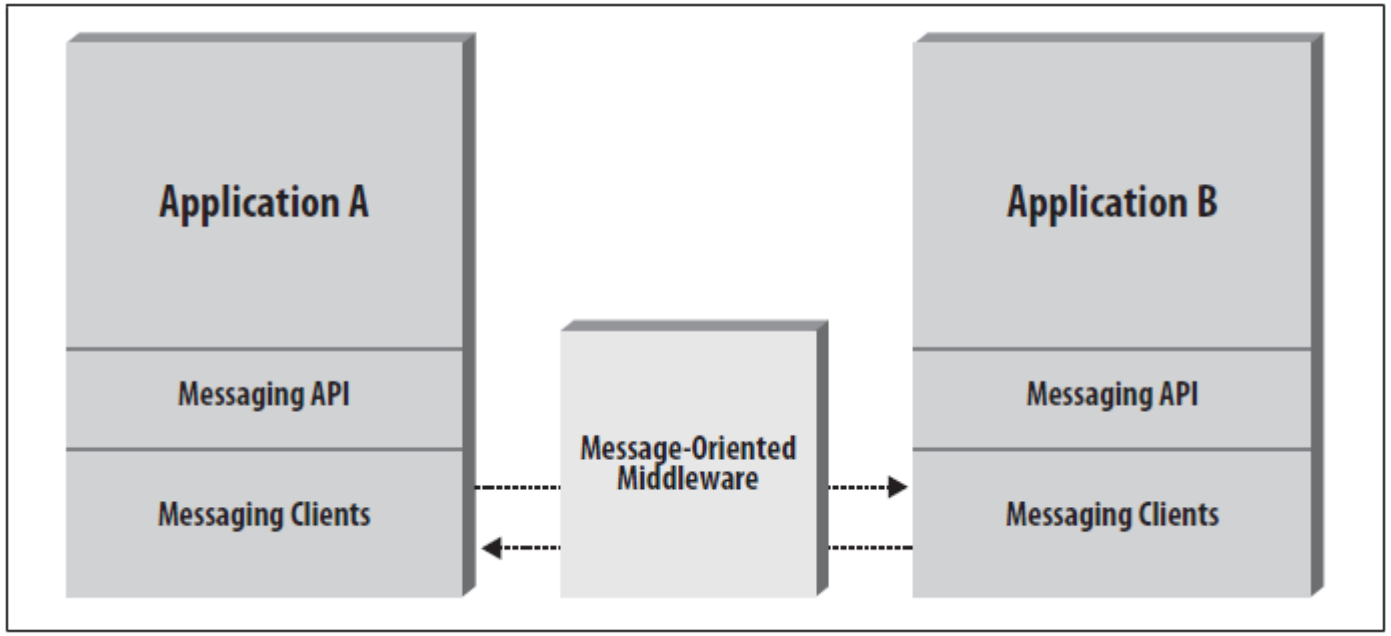
Is latency a hindrance for business?

Reliability

Is the network reliable and can we trusts that messages are delivered?

Middleware

- Software which is specifically designed to integrate Business Applications is called: Middleware. This is a synonym for Integration Platform.
- When the only integration style of the middleware is Messaging then it is: Message Oriented Middleware (MoM).
- Hybrid Integration Platform = Middleware which supports more integration styles than just messaging.
- Middleware integrates heterogeneous applications.



Why EAI?
Most companies have heterogeneous applications

- The IT systems landscape has multiple applications developed:
  - By Different teams independently
  - Using Different programming languages
  - On Different platforms

- Even companies which have a homogeneous IT systems landscape may suddenly face a heterogeneous situation. For example due to mergers and acquisitions.

Definition of Software Architecture

“A software architecture defines how a system is made up of its individual components. It describes the interfaces through which they are connected, and the processes of interaction.

It will impact all decisions related to it. Especially on technology selection and mapping to operational systems.

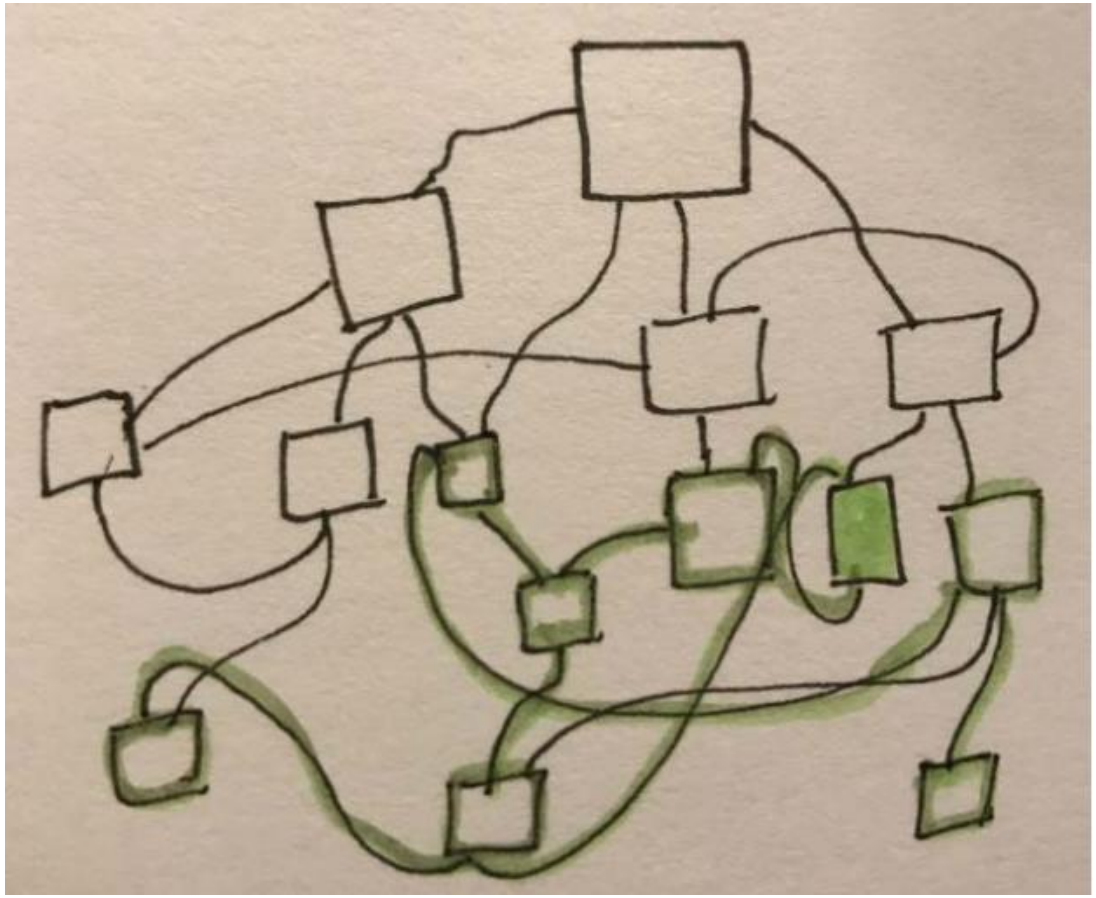
The goal is always to fulfill functional, as well as the non-functional requirements of the client.”

Business Software Architecture
Micro and Macro Software Architecture

Dowalil (2018) describes 2 levels of software architecture:

- Micro architecture or software (code) design.
- Macro architecture defines how the IT landscape is structured in components.

Business Software Architecture
Big Ball of Mud (Anti-pattern)



- "A **big ball of mud** is a software system that lacks a perceivable architecture. Although undesirable from a software engineering point of view, such systems are common in practice due to business pressures, developer turnover and code entropy. They are a type of design anti-pattern."
- Source: Big ball of mud - Wikipedia

Business Software Architecture

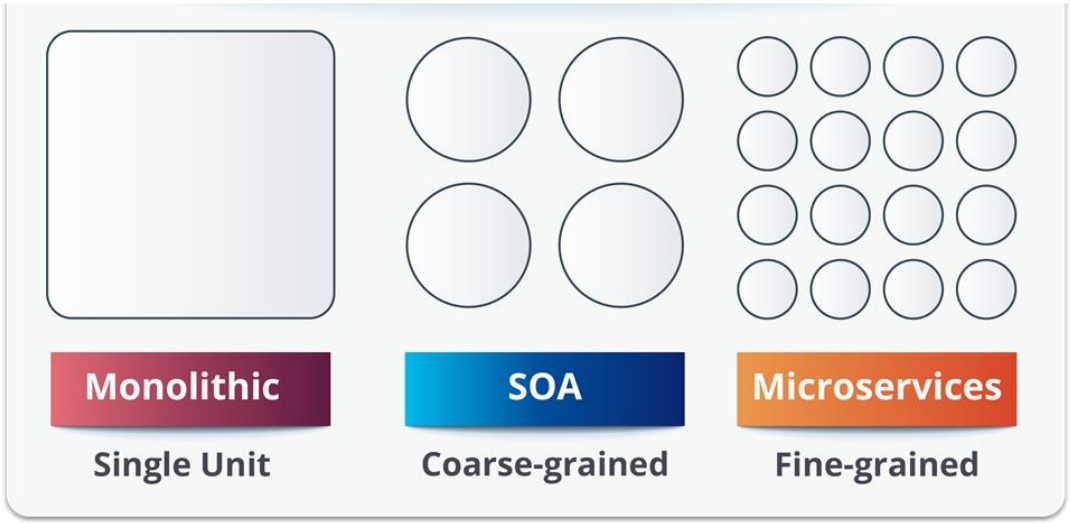
Monolith

- The software application is not a distributed application
- It can be deployed as 1 system
- Some consider this as an anti-patterns, and others don’t.
- When not tightly managed Monoliths can become a Ball of Mud over time.

Business Software Architecture

Hybrid

- In many companies different business software architectures coexist together.
- This could be the result of for example a Best-of-breed strategy or a merger.
- It is a mixed, hybrid situation.



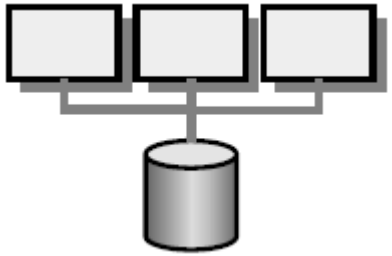
Key - EAI related – Business Software Architecture questions

- What are the EAI quality requirements for software components and integration?
- How to structure the IT landscape in components?
- Which integration styles and technologies should we use?
- Do we need middleware software from suppliers to serve as an integration platform?

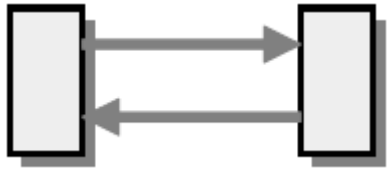
4 standard Application Integration Styles

1. File Transfer

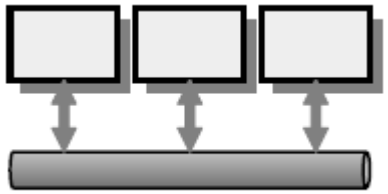
2. Shared Database



3. Remote Procedure Invocation



4. Messaging



Integration style: File transfer

Advantages

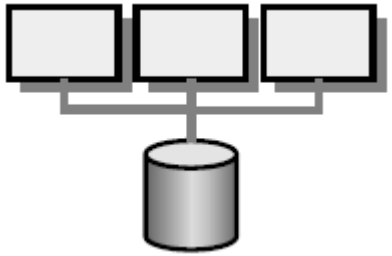
- The great advantage of files is that integrators need no knowledge of the internals of an application. The files effectively become the public interface of each application.
  => Loosely coupled systems.
- No tools or integration platform needed.

Disadvantages

- Managing files and agree on file formats, locations and unique filenames can be complex.
- Can lack timeliness => systems can get out of synchronization due to delays.
- Example problem *"If an address is updated inconsistently in rapid succession in two systems, how do you decide which one is the true address?"*
- semantic dissonance risks

(quotes from Hohpe & Woolf p. 44 etc.)

Integration style: Shared Database



Advantages

- Each application has access to any of the shared data whenever it needs it.
- Avoids adding yet another (DB access) technology for everyone to master.
- Developers will solve problems with sematic dissonance before software goes live.

Disadvantages

- provides a large unencapsulated data structure;

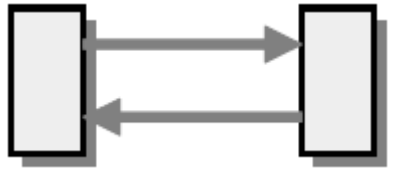
  => Database changes have a considerable ripple effect through every application

  => organizations often very reluctant to change

  => much less responsive to the changing needs of the business. Not agile.
- Coming up with a unified schema that can meet the needs of multiple applications is a very difficult exercise .. And developers find it hard to work with.
- If a critical application is likely to suffer delays in order to work with a unified schema, then often there is irresistible pressure to separate applications.
- A harder limit to Shared Databases is external packages. These database are changed by the supplier.
- Can become too slow to be practical (?)

(quotes from p. 47 etc.)

Integration style:
Remote Procedure Invocation



Advantages

- Applies the principle of encapsulation
- Shares data and invokes processing of data.
- Each application:
  => maintains the integrity of its own data,
  => can alter the format of its internal data without affecting every other application.
- Applications can provide multiple interfaces to the same data.

Disadvantages

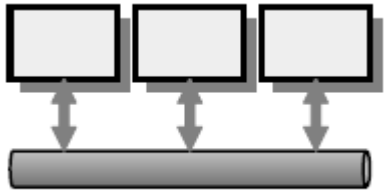
- Each application has to negotiate its interface with its neighbors.
- In some technical environments there are big differences in performance and reliability between remote and local procedure calls => risking slow and unreliable systems.
- Applications are still tightly coupled; doing certain things in a particular order can make it difficult to change systems independently.

Remote Procedure Invocation is also called Remote Procedure Call (RPC). In fact RPC is the commonly used term nowadays.

Using Web Services over HTTP is a RPC technology.

(quotes from p. 50 etc.)

Integration style: Messaging



Advantages

- Enables Loosely Coupled systems
  - Sender does not need to wait on the receiver
  - Does not require both systems to be up at the same time.
  - Messages can be transformed in transit
- Allows for immediate exchange of message content

Disadvantages

- Potential (temporary) data consistency problems
- Testing and debugging is harder
- Coding can be more complex
- Risk that "*that integrators are often left with writing a lot of messy glue code to fit everything together*"
- Additional vendor lock-in risks

(quotes from p. 53 etc.)

Classroom assignment

- Which other integration styles could you think of?
- Which integration style is mostly used in Business?
- Copy / paste ☺

Does Google Drive represent a new way of application integration?