universität wien

Enterprise Application Integration

Faculty of Computer Science
Workflow Systems and Technologies

Interoperability

Introduction

- Enterprise application integration (EAI) aims to provide a *unified set of services* by *integrating data and functionality* of *multiple separate applications* with the support of integration approaches
- Enables unification and standardisation of processes in enterprises
- Examples:
  - Share data between organisations
  - Expose unified APIs that accomplish complex tasks involving the functionality of multiple dispersed systems
  - Orchestrate processes across different organisations

Application Integration Styles

- **File Transfer** Applications export and import files of shared data
- **Shared Database** Multiple applications store their data in a single shared database
- **Remote Procedure Invocation** Applications expose their services which enable to invoke behaviour remotely
- **Messaging** Applications interact with a common messaging system to exchange data and invoke behaviour

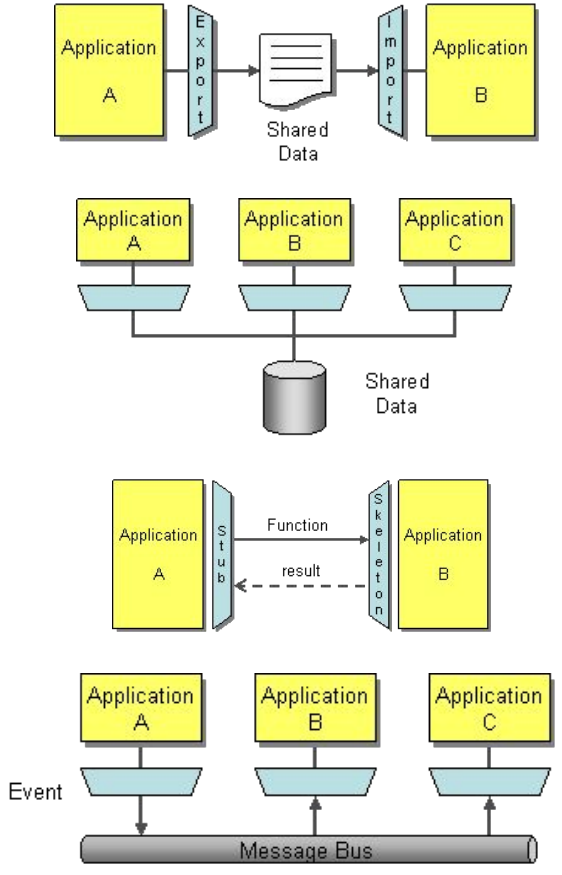


Illustration of the four Integration Styles [1]

Application Integration Criteria

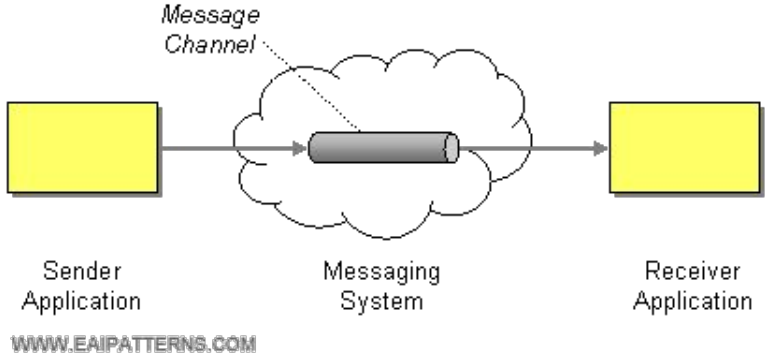
- **Integration** Does an application require collaboration with other systems to accomplish tasks?
- **Application Coupling** Integrated applications should avoid tight coupling and provide room for future changes.
- **Integration Simplicity** Advocate solutions which require minimal changes to the application and minimal amount of integration code.
- **Asynchronicity** Integration solutions must not assume constant availability of remote applications and block computational resources
- **Data or Functionality** Ability to handle exchange of data but also invoke (e.g. computationally heavy) behaviour

Application Integration Criteria cont'd

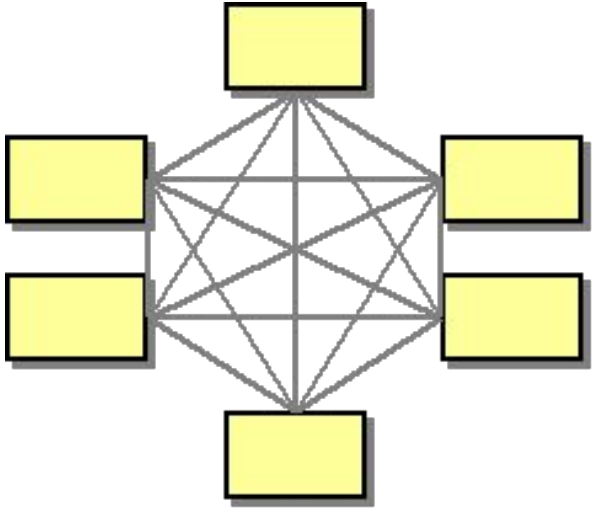
- **Data Format** Ability to unify different data formats and handle the evolution of those formats over time
- **Data Timeliness** Data to be shared produced by an application should be delivered to its designated consumers in a timely manner
- **Integration Technology** Certain integration approaches may require a highly specialised solution which may introduce further complexity and potentially result in vendor lock-in and high costs

Messaging

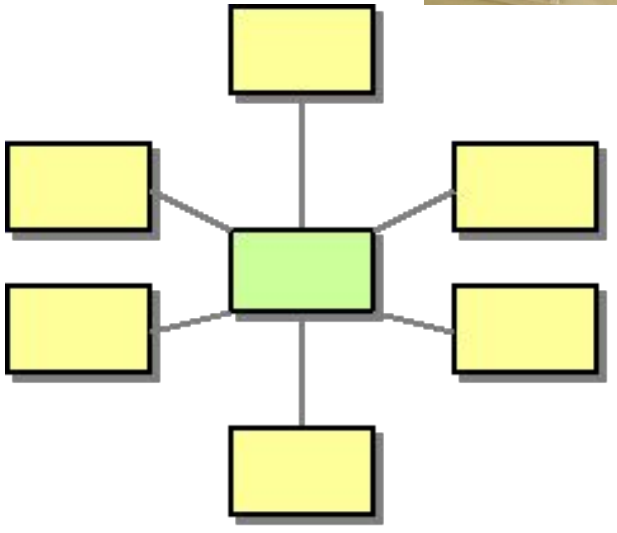
- Communication between applications via a *Message Channels*
- *Sender* writes information *to* the channel while the *Receiver* reads information *from* channel
- Sender *does not* necessarily know the particular recipients of the information provided
- *Choice* of Message Channel determines the recipients of a Sender's information



The n^2 Integration Problem

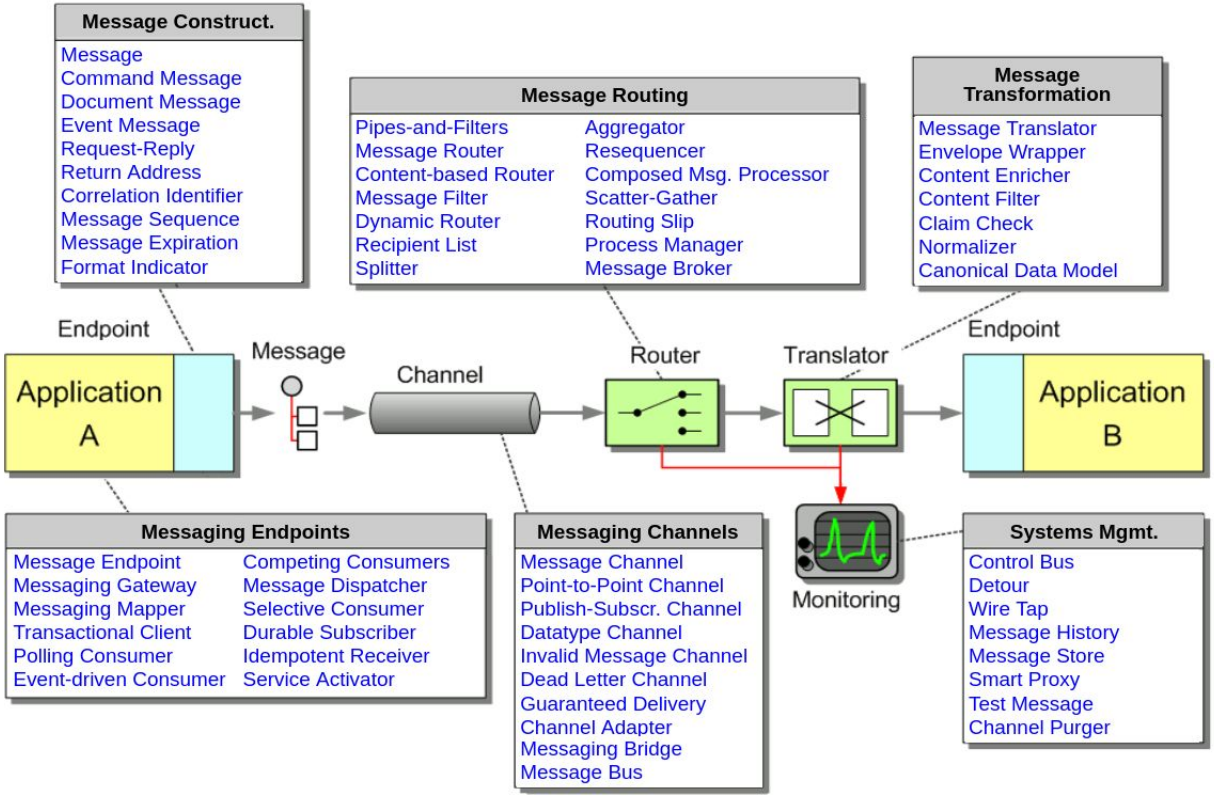


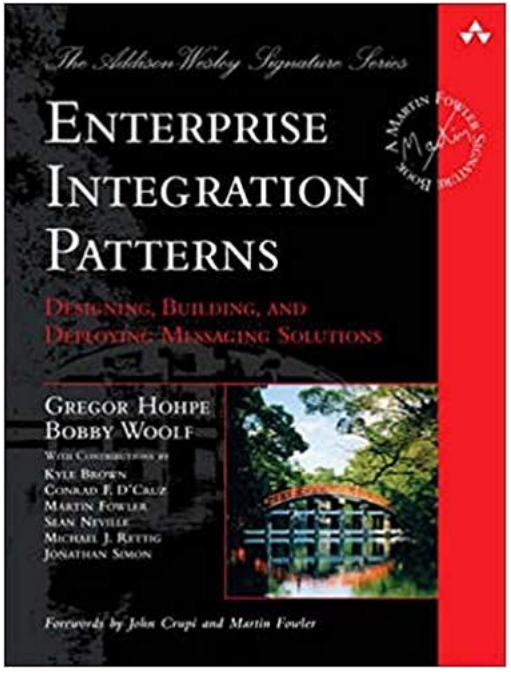
Spaghetti Integration [1]:
up to *n* * (*n* - 1) connections



Solution: Hub and Spoke [2]
a.k.a. "**Message Broker**"

Messaging Patterns





https://www.enterpriseintegrationpatterns.com/patterns/messaging/

Messaging Patterns

Messaging patterns provide *technology independent design suggestions* for integration problems.

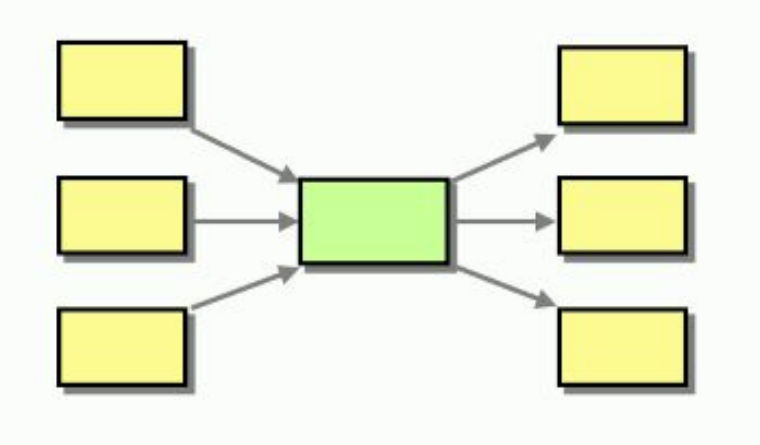
- **Channel Patterns** describe how messages are transported across a unidirectional message channel and how the sender and receiver can be decoupled
- **Message Construction Patterns** describe the intent, form and content of messages passed over a messaging system
- **Routing Patterns** describe how messages are routed from a sender to the desired receiver based on a set rules and conditions

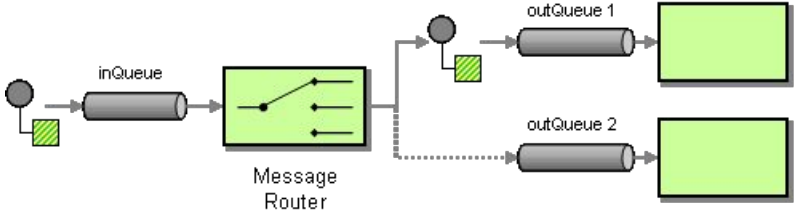
Messaging Patterns cont'd

- **Transformation Patterns** deal with the transformation of the content of messages into the appropriate format required by the receiver
- **Endpoint Patterns** deal with how messages are produced and consumed by the clients of messaging systems
- **System Management Patterns** describe how to maintain and monitor messaging systems

Message Broker

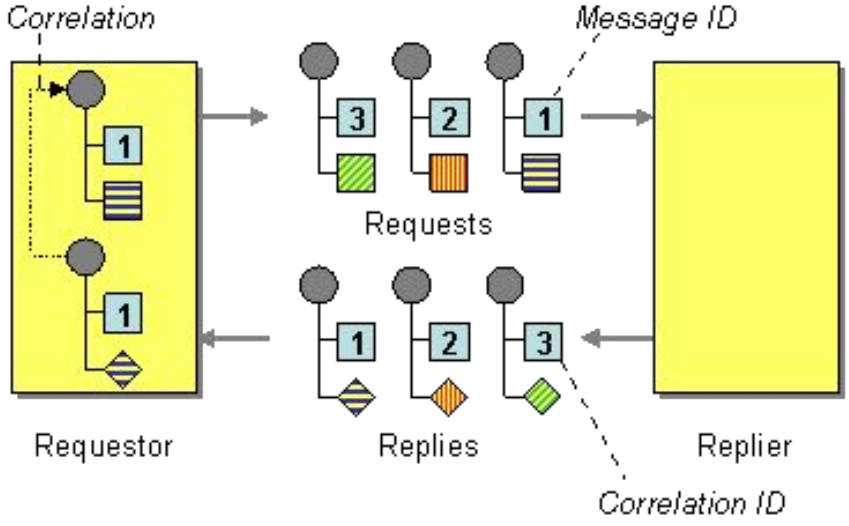
- Architectural pattern to facilitate the correct delivery of incoming messages to their intended target
- Can have multiple sources of incoming messages and receivers
- May implement several routing patterns to determine the appropriate channel for the incoming messages
- Potentially single point of failure





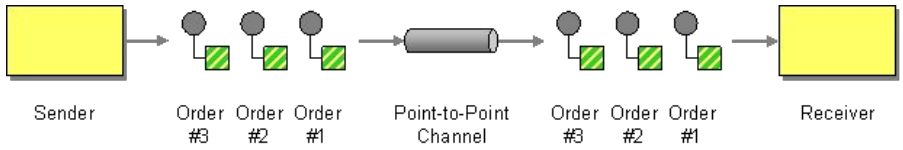
Correlation Identifier

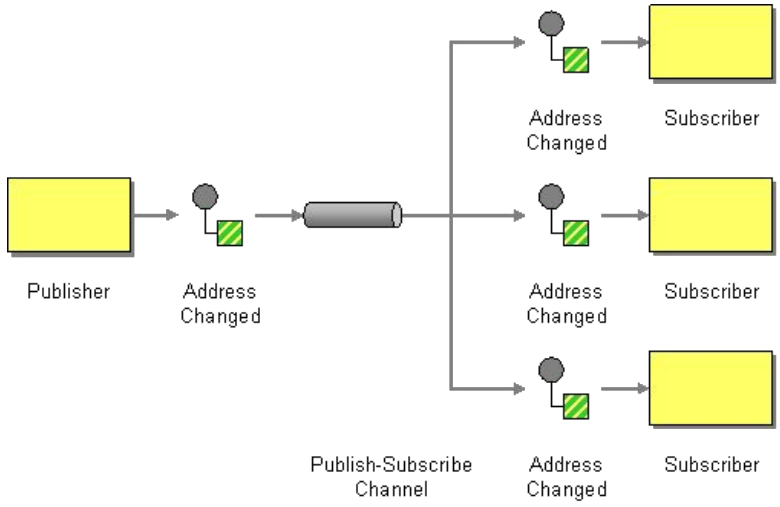
- Requestor includes a message identifier (ID) - a token that uniquely identifies the message
- Replier extracts the token - now referred as correlation identifier - from the received message and includes it in the response



Point-to-Point and Publish-Subscribe Channels

- Point-to-Point Channel only has one receiver
- In case multiple receiver exists only one of them gets to consume the message
- In contrast a Publish-Subscribe pattern delivers a message to all interested receivers (subscribers)
- Requirements may include reliability of message delivery





Service Composition

(Alonso et al., 2003)

- **Composite services** are implemented by combining the functionality provided by other web services.
- **Service composition** is the act of creating new services by composing existing services.
- Examples: WS-BPEL (OASIS) and WS-CDL (W3C) support defining service compositions for complex interactions.

Orchestration vs Choreography

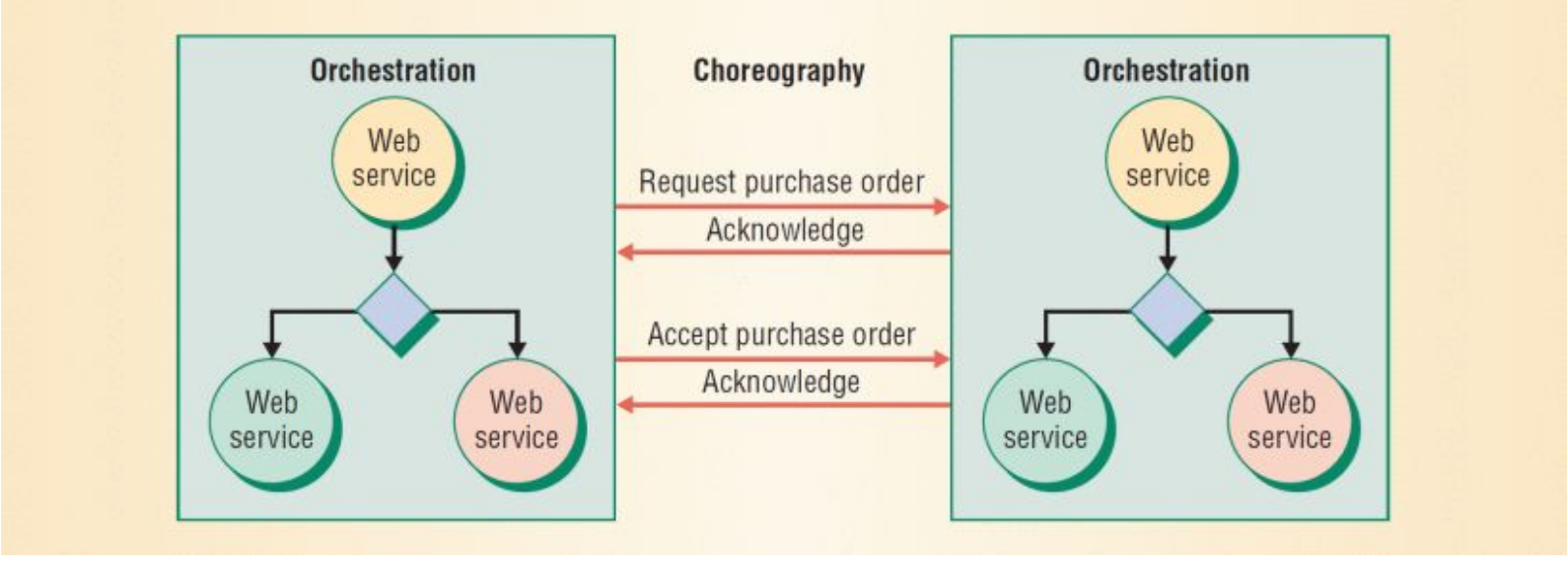


Illustration of the high level relationship between an orchestration and a choreography from (Peltz, C., 2003)

Orchestration vs Choreography

In the context of web service composition (Peltz, C., 2003):

- Orchestration focuses on executable business processes
  - Coordinated execution of internal and external web services
  - Execution is controlled by a single entity ("orchestrator")
- Choreographies describe the flow of messages between parties
  - Defines collaboration between two or more participants in terms of how the participants may interact (rules of engagement)
  - Focus typically on the public cross-organisational message exchange
- See BPMN 2.0 for a graphical modelling language for orchestrations and choreographies.

Resources

- Hohpeand, G. et Woolf, B. (2003). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Longman Publishing Co., Inc., 2003
- Hohpeand, G. (2003b). Hub and Spoke [or] Zen and the Art of Message Broker Maintenance. https://www.enterpriseintegrationpatterns.com/ramblings/03_hubandspoke.html
- Alonso, G., Casati, F., Kuno, H. and Machiraju, V. (2004) Web Services. Concepts, Architectures and Applications, Springer-Verlag Berlin Heidelberg.
- Peltz, C. (2003). Web services orchestration and choreography. Computer, 36(10), 46-52.