Business Information Systems
Unit 7
Enterprise Application Integration (EAI)

Dr. Martin Hepp

Goal of Enterprise Application Integration

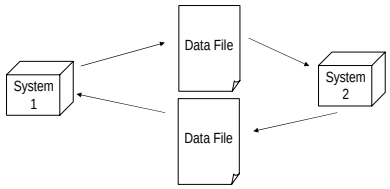
- Integration of business functionality, that is hosted on various systems, along the value chain
- Integration on the Data and Process Level
- The existing systems usually remain unchanged
- Effects
  - consistency (e.g. on the data level)
  - real-time operations
- Scope can include external systems (e.g. in Supply Chain integration challenges)

Four Basic Approaches

- File Transfer
  - Application 1 write a file for Application 2 to be read.
- Shared Database
  - Multiple applications share the same DB schema located in the same DB
- Remote Procedure Invocation
  - One application exposes functionality to be invoked remotely by other applications in real time and synchronously
- **Messaging**
  - One application publishes a message to a commin message channel. Other applications can read the messages from the channel at a later time.

Hohpe/Wulf, p. xxx

Systems Integration in the Pre-Web Era

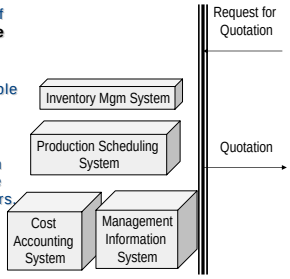


Usually nightly batch-runs each day, week, or month

Integration by Batch Data Exchange Doesn't Work in the Web Age

- On the outside borders of the enterprise, **real-time interfaces** are required.
- The results often require the combination of multiple systems.
- There are countless **interdependencies between operations** on various systems, e.g. the order of execution matters.
- This calls for a
  - fully automated and
  - process-oriented

approach



Messaging

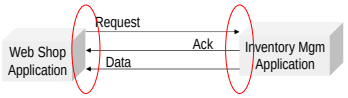
- Core technique in EAI
- Basic principle
  - Channels/Queues: logical array of messages for a given purpose
  - Sender or producers writes messages to a channel
  - Receiver or consumer reads and deletes a message from a channel
- Message Header
  - Data about the sending application etc.
- Message Body
  - Application Data to be conveyed

Advantages of EAI Messaging

- More immediate than file transfer, better encapsulated than shared databases, more reliable and better decoupling than remote invocation.
- One uniform system for platform/language integration
- Asynchronous communication, especially „send-and-forget“, if feasible
  - also, variable timing, better throughput
- Load balancing and throttling
  - receiver can process messages at its own pace
- Smooth integration of offline devices (e.g. PDAs)
- Sender does not need to know all recipients

cf. Hohpe/Wulf: Enterprise Integration Patterns, p. xxxvi

Achieved integration is like sand in your hand

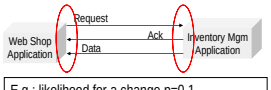


- Systems and their interfaces undergo change
- Examples
  - Data structure and representation
  - Choreography

Applications Change Over Time

- Data Level
  - due to inherent conceptual dynamics in any domain
  - due to new standards in the respective domain
  - due to mergers and acquisitions and resulting code alignment
- Choregraphy / Process Level

The Likelihood of Broken Integration



E.g.: likelihood for a change p=0.1

=> likelihood for a broken link = 0.1 + 0.1 = 0.2

E.g.: likelihood for a change p=0.1

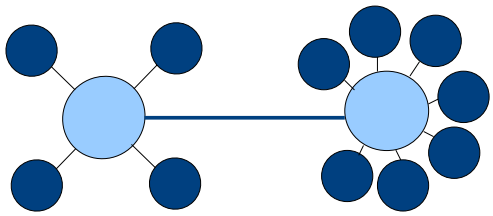
=> likelihood for a broken link = 0.1 + 0.1 +0.1 +0.1 = 0.4

Increases with number of systems, vendors, and depth of integration

Synchronous Processing

Asynchronous Processing

Hub and Spoke



Publish Subscribe

- Services can subscribe for notication in case of defined events
- Allows for a service to define in which cases it must know a change of state of another system.

Levels of Transformation

Layer	Deals with	Example
Data Structures	Entities, Associations, Cardinality	Condense N:M relationship into aggregation
Data Types	Field Names, data types, value domains, constraints, code values	Replace US State name with two-character code
Data Representation	Data Formats (XML, name-value pairs, fixed length data fields) Character/Code Sets (ASCII, EBCDIC, Unicode)	Convert ASCII to Unicode
Transport	Communications protocols: TCP/IP sockets, HTTP, SOAP, JMS	Move data across protocols without affecting message content

cf. Hohpe/Wulf: Enterprise Integration Patterns, p. 87

EAI vs. Middleware

- Traditional Middleware focuses on the low-level integration
- Examples
  - Bridging platforms
  - Bridging transport protocols
  - Converting code sets
  - Queuing
- EAI is centered on the process and logical view on interaction

Person-to-System Integration

- Integrate system just from the perspective of a human user
- E.g. one GUI for multiple systems
- Enterprise Portals

Screen Scraping

- For the integration of very old legacy systems, the keyboard and screen data can be captures/simulated in order to expose encapsulated functionality.

Metadata in EAI Scenarios

- A significant amount of metadata must be stored and managed to support an EAI service. Most EAI tools hold this data internally, often within proprietary data stores. However, businesses often require access to this metadata for purposes such as:
  - Replication to other instances of an EAI service
  - Programmable access by other services
  - Backing up and restoring for resilience
  - Sharing metadata with other businesses
  - Data mining and reuse
  - Change management

*