Integration of Applications

Contents:

- Integration Styles
- ETL
- EAI
  - Types of integration
- B2B
  - From EAI to B2B
  - Characteristics
- ESB

Mariano Cilia / mcilia AT gmail.com

The Need for Integration

- Enterprises typically comprised of hundreds of apps
  - Custom-built
  - Acquired from a third party
  - Part of a legacy system

  Operating in multiple tiers of different OS platforms
- Why this?
  - Creating a single, big app to run the complete business is impossible
  - ERPs (SAP, Oracle, …) only run a fraction of the business functions required in a typical enterprise
  - Vendors focus on a specific core function
  - For the company provides flexibility, since it can buy best-of-segment product
  - When companies start from scratch they don't imagine this complex situation

Integration Challenges

- Multiple apps running on multiple platform in different locations
- Business implications
- Integration constraint
  - Almost no control over "legacy" or packaged apps. Cannot be changed/adapted
- Incompatibility of standards
  - Remember interoperability problems of CORBA
- Semantic differences
- The mix of technologies

3

Coupling (Review)

- Measure of dependency between applications
  - Technology Dependency
  - Temporal Dependency
  - Location Dependency
  - Data Format Dependency

4

Tightly Coupled Systems

- Make many assumptions about each other
- Well suited for internal communication inside of an application
- Well suited for "near" communication where we have control over both sides of the interaction
- Generally more efficient, easier to develop and debug

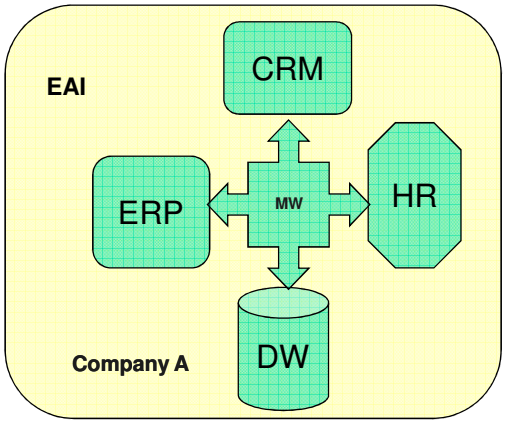
5

Loosely Coupled Systems

- Make fewer assumptions about each other
- Well suited for "far" communication where we do not have control over all of the systems
- Allows systems to vary independently from each other
- Generally less efficient, more difficult to develop and debug

6

Enterprise App Integration (EAI)



- EAI deals with the integration of applications within an enterprise
- systems are integrated through middleware

Application Integration Criteria

- Application coupling
  - Minimized dependencies
- Integration simplicity
  - Minimize integration code
- Integration technology
  - Reduce the use of specialized (vendor-specific) software
- Data format
  - Consider evolution and extensibility of data formats
- Data timeliness
  - Reduce latency of data sharing
- Data or functionality
  - Integration may include invoking the functionality of others
- Asynchronicity
  - Fire-and-go

Integration Styles

- File transfer (batch transfer)
- Shared database
- RPC (shared business function)
  - Method-oriented
  - Process-oriented
- Messaging
- Portal

9

Batch Transfer Integration

- Bulk data transfer and then batch updating using Extract, Transform and Load (ETL) techniques
- Due to the latency of batch updates and rekeying information (tel, fax, snail mail) → latency between business events and its digitalization
  - In supply chain this can result in not knowing what's in inventory at a given time

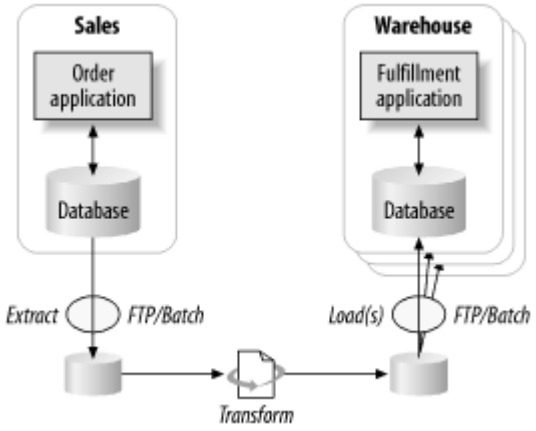
10

Extract, Transform and Load

- "overnight" batch process
  - An app exports its data to a common neutral format
    - Intermediary DB or flat file
  - The representation of data in a different format than the target → data needs transformation
  - The intermediary data is then transferred to another app (may be using FTP)
  - The destination app imports the data using a merge-and-purge process

11

Extract, Transform and Load



12

ETL Pro/Cons

- Most commonly used
- Low cost since applications don't need to be changed and redeployed
- Consists of a complex maze of applications, scripts, manual processes and FTP file transfers
  - complexity often appears lower than it is because of size (hundreds or thousands of tables) and semantic heterogeneity
- Additionally,
  - Unreliable transfer
  - Lack of data validity
  - Undesired downtime
  - Overall latency (data timeliness)
- There is no "overnight" anymore!
  - Today you cannot shutdown your system to synchronize your data
  - There is at least a 24 hours delay in getting access to real information

13

Shared database

- A central, agreed-upon datastore that all apps share
- Rely on SQL
- Need to resolve semantic heterogeneities
  - May impact on many apps
  - Find a unified schema is hard!
- Large **un**encapsulated data structure

14

RPC

- Applies the principle of encapsulation
- Each app can maintain the integrity of the data it owns
  - Each app can alter its internal data without affecting every other app
- Developers know traditional (local) invocation semantics but big differences with RPC
  - Specially performance and reliability
  - Rely on HTTP due to firewalls
- Difficult to add transformation components
  - Apps have to negotiate their interfaces with their neighbors
- Applications become tightly coupled

15

Method-oriented Integration

- Sharing of business logic that exists in the enterprise
  - e.g. updating customer record with one method from multiple applications
- share methods already existing inside applications using method-sharing technology (distributed objects)

16

Process-oriented Integration

- Places an abstract business layer on top of existing systems
- Abstraction of business processes to one common understanding
- Leverages other basic integration approaches (data, API, method)-oriented
- Trend if moving to service-based architectures (e.g. WS)
- Need business process manager

17

Messaging

- Like "file transfer" where
  - lots of little data packets (msgs) can be produced quickly,
  - transferred easily and
  - the app receiver is automatically notified when a new msg is available for consumption
- The message structure can be easily changed
  - can be transformed in transit
  - reflecting the changing needs of the enterprise
- Can mimic the RPC approach
  - but async and reliable

18

Portal-oriented Integration

- Use of one common interface (usually browser based) to access multiple applications
- Superficial integration at the user interface level
- Avoids more expensive back-end integration but has similar problems as data-oriented integration
- Burden of interpretation often placed on human being

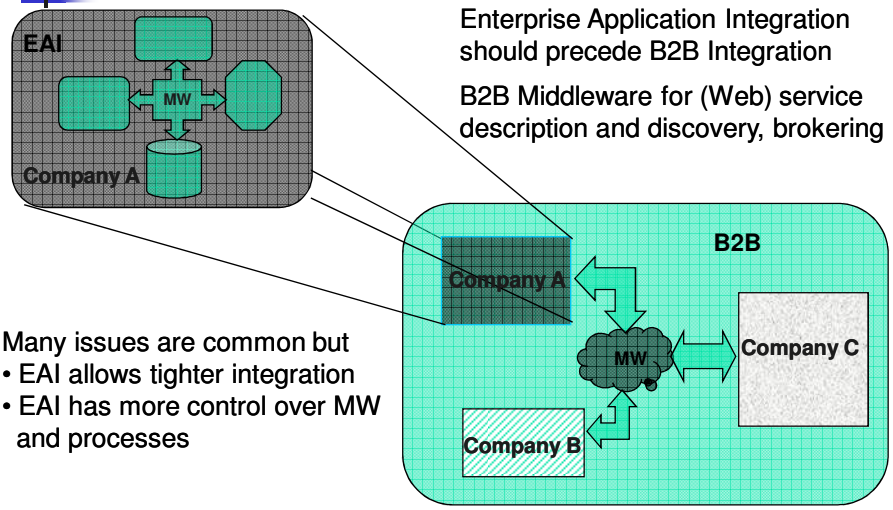
19

Integration Styles (Wrap up)

- File transfer and shared DB enable apps to share their data, BUT not their functionality
- To integrate apps' functionality rather than their data → RPC
  - BUT it tightly couples apps
- To enable frequent exchanges of small amounts of data → messaging
  - Relying on a format per data type instead of one universal schema

21

From EAI to B2B



Enterprise Application Integration should precede B2B Integration

B2B Middleware for (Web) service description and discovery, brokering

Many issues are common but
- EAI allows tighter integration
- EAI has more control over MW and processes

22

Business to Business Integration

From Bussler'03 and e-BizQ.com

Definition

- Trading:
  - B2B integration refers to all business activities of an enterprise that have to do with electronic message exchange between it and one or more trading partners
- Software Technology:
  - Infrastructure to connect any backend application system within enterprises to all its trading partners over formal exchange protocols (known as B2B protocols)

24

Motivation

- Enterprises use different backend systems that need to exchange data
  - The #of backend apps can vary from a few up to several hundred
  - Apps are designed and built to operate in isolation
- Enterprises normally exchange data
  - E.g. Purchase orders
  - Usually using phone, fax, or email
    - Reentering data manually in another system

25

B2B Integration Technology - Characteristics

- Backend apps and also enterprises are
  - heterogeneous
    - different conceptual models for expressing the business semantics
  - autonomous
    - changes on systems without consulting others
  - distributed systems
    - maintains its own state separate from each other (they do not share data or state)
- B2B needs to play the role of coordinator between them addressing these three properties

26

B2B Integration Technology - Requirements

- Heterogeneity
  - Data transformation is needed
    - Maps the representation and meaning of one system into the equivalent representation and meaning of the other system
- Autonomy
  - Must be able to observe state changes in order to allow coordination
- Distribution
  - Must transport data between the systems in order to enable the sharing of data

27

Evolution of B2B Integration

- Homegrown integration
- Point-to-Point integration
- Hub-and-Spoke integration
- Process-based integration
- ASP integration

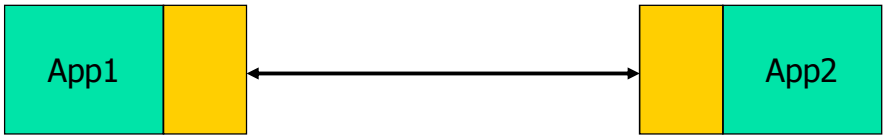
28

Homegrown Integration

- No integration products were available
  - Integration was not considered yet a profitable market
- Backend apps called each other synchronously to exchange data
- Asynchronous comms through intermediate storage (e.g. files or queues)
- Backend apps are now aware of other apps
  - Needs to know the correct recipient (if >2 apps)
  - Data transformation by applications

29

Homegrown Integration



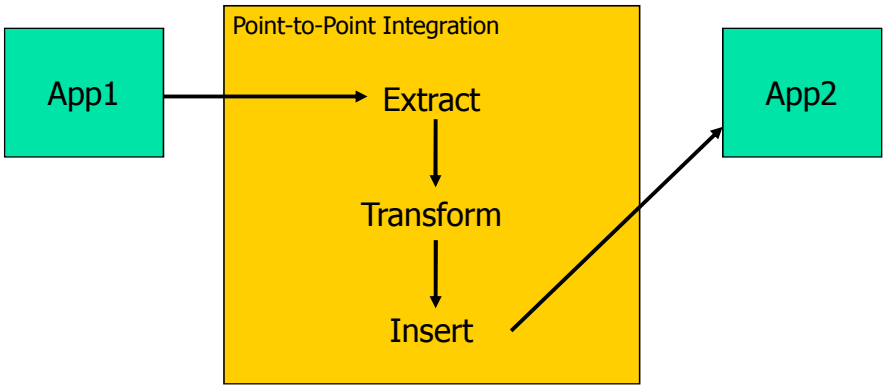
30

Point-to-Point Integration

- Pairwise integration of backend apps
  - For each pair a direct data transfer is established
  - Integration software
    - extracts business data from backend apps
    - transports it to the other app
    - inserts data
- Known as A2A if within an enterprise

31

Synchro Point-2-Point Integr.



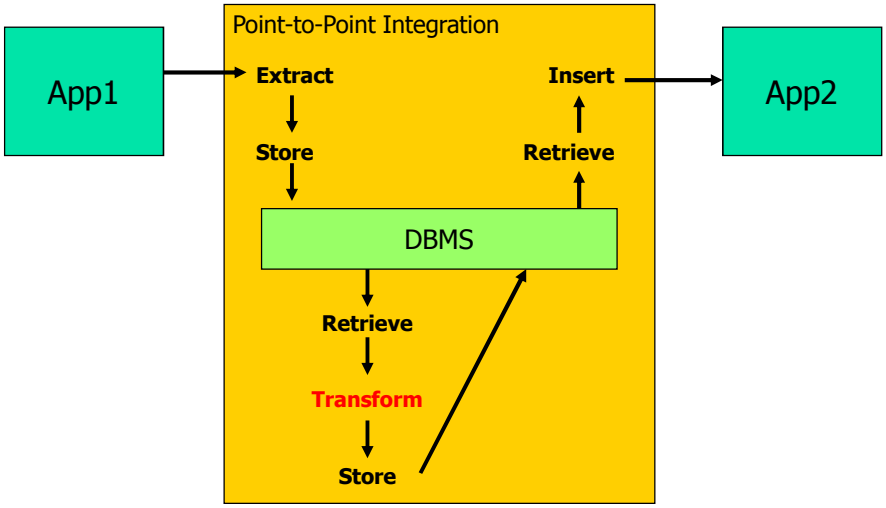
32

Asynchro Point-2-Point Integr.

- Intermediate storage
  - Integration buffer
  - Isolates backend apps from each other
- Transformation
  - As a separate process
  - After extraction (before intermediate storage)
  - After retrieved from intermediate storage and before inserted into backend

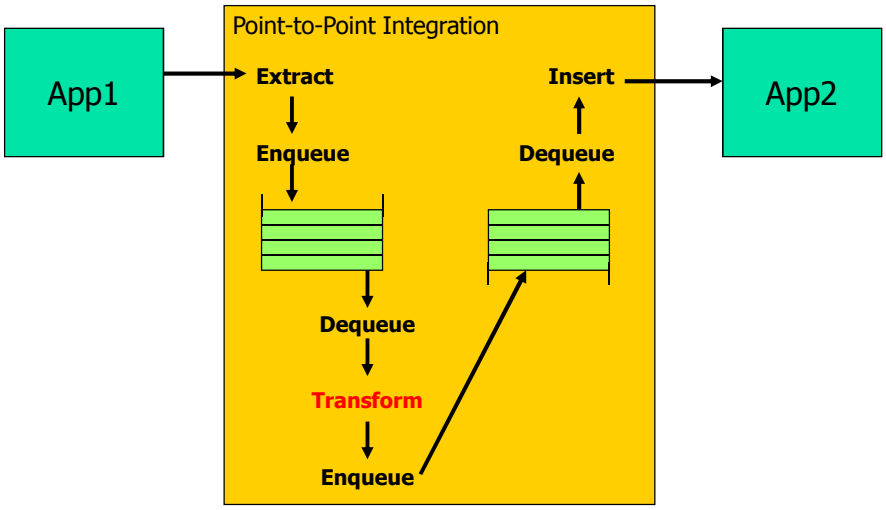
33

Asynchro Point-2-Point Integr.



34

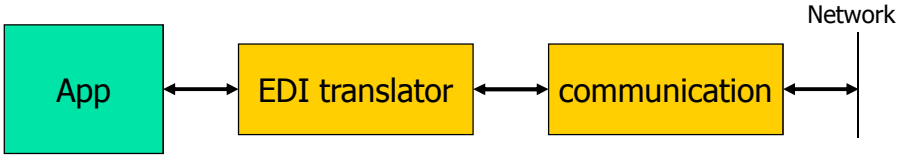
Asynchro Point-2-Point Integr.



35

EDI Approach

- Electronic Data Exchange (EDI)
  - Defines the syntax of the message
  - Communication software included



  - Protocols
    - SWIFT financial domains
    - RosettaNet supply-chain in the High-Tech Industry
    - ebXML emerging std suitable for across multiple industries

36

Point-2-Point Integration

- Provides basic integration functionality
- Limitation 1:n!
  - For each new app 2 data transfer links have to be established to each existing backend app
  - Transformation needs to be defined
  - Intermediate storage need to be understandable for monitoring purposes

37

Hub-and-Spoke Integration

- Change on topology
  - A central and common storage for all is available
  - Data transfer is no longer between each pair of backend apps
  - Each backend app (spoke) only communicates with the central hub
- The central hub transfers the data to the target spoke

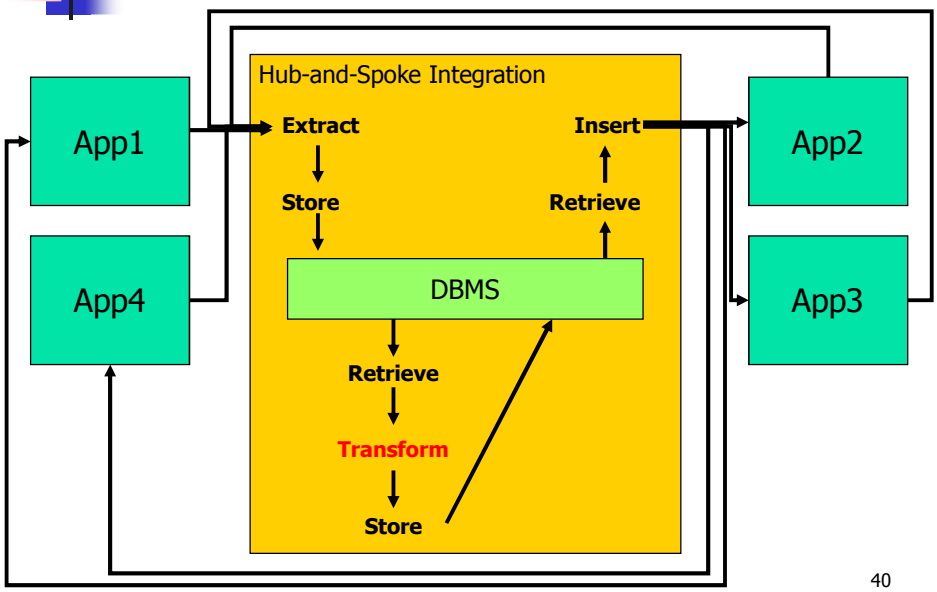
38

Hub-and-Spoke Architecture

- The sending spoke determines the target spoke (in the header of the msg)
- The msg is sent to the hub
- The hub …
  - Reads the header
  - Transforms msg, and
  - Forwards the msg to the target spoke

39

Hub-and-Spoke Integration



40

Hub-and-Spoke Architecture

- Advantage over Point-2-Point
  - Adding a new backend app only requires the addition of one data transfer (between app and hub)
  - One storage of msg for the whole hub
  - The number of data transformations is the same
- Evolution: The hub as a Pub/Sub NS
  - target spoke in the msg is not required
    - additionally, more than one target is possible
  - spokes issue subscriptions on business data

41

Drawbacks of .2. and H&S

- No multistep integration
  - One msg needs to be processed by more than one app
- No business logic
  - No additional business logic can be added between data extraction and data insertion
    - E.g. authorization activities, etc.
- Only one-way integration
  - No relationships is "possible" between msg
    - à la Request/Reply

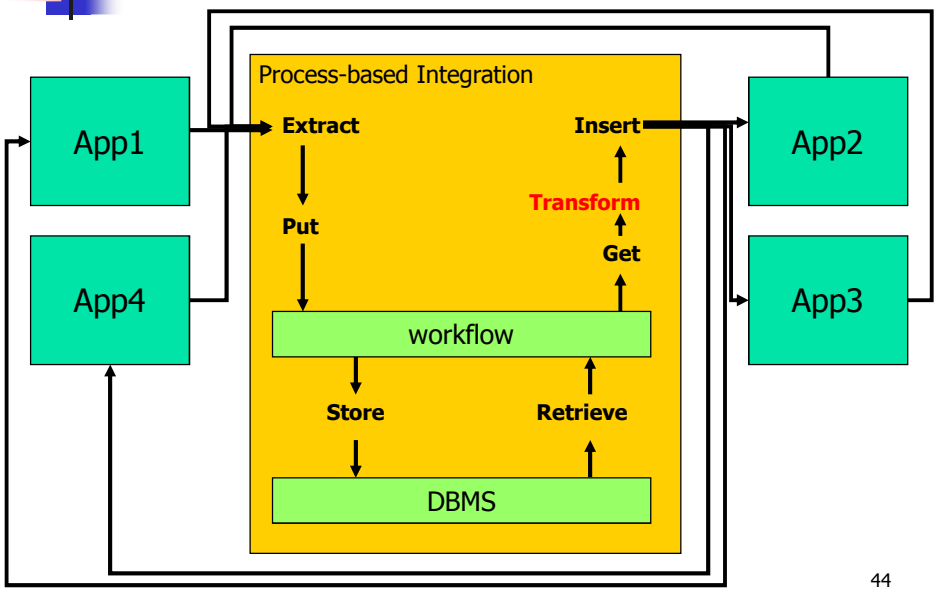
42

Process-based Integration

- Hub-and-Spoke extended with process mgmt functionality (in the form of workflow mgmt)
- Workflow governs the integration
  - Applying transformation and business logic
  - The Wf determines the recipient app (spoke) of the msg

43

Process-based Integration



44

Process-based Integration

- Multistep is possible due to the use of workflow
  - Workflow controls the integration by retrieving, storing and transforming msgs
- Business logic
  - Conditional branching, etc
- Workflow allows the definition of complex interaction patterns among spokes

45

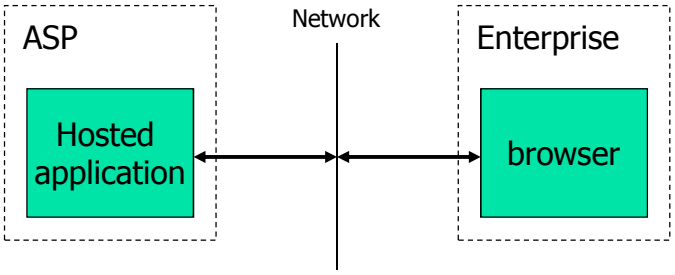
There are still Drawbacks

- Workflows need to deal with several formats
  - Making the logic quite complex since different workflow tasks deal with different formats of the business data
- Workflows need to deal with different protocols
  - Impeding the reuse of workflows
  - E.g. some backend apps need acks
- Maintenance becomes rather complex
  - If there is a high number of trading partners, or
  - If the integration rules change frequently

46

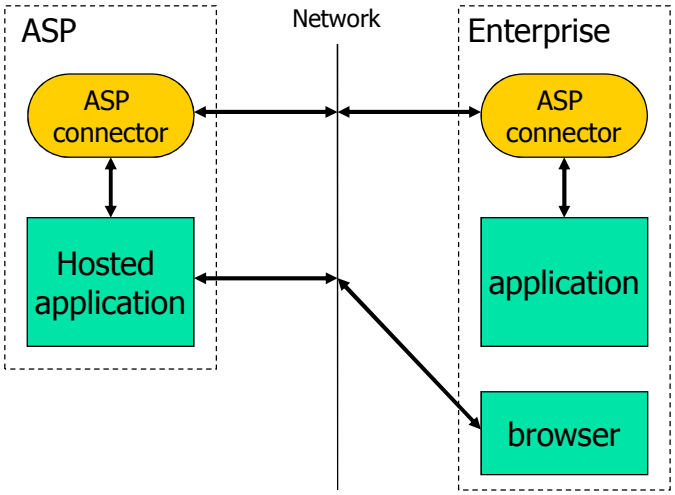
ASP Integration

- Application Service Provider (ASP) rents software



47

ASP Integration



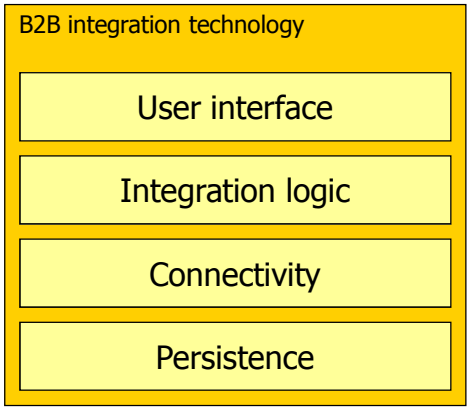
48

ASP Integration

- Integration of locally installed backend apps with hosted ones
- The integration crosses the boundaries of two companies
  - Therefore the ASP connector on both sides

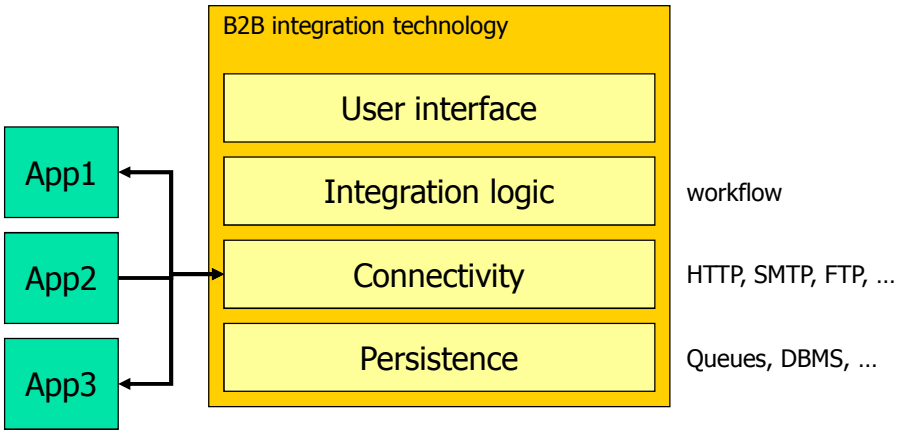
49

B2B Integration Technology



50

B2B Integration Technology



51

Enterprise Service Bus (ESB)

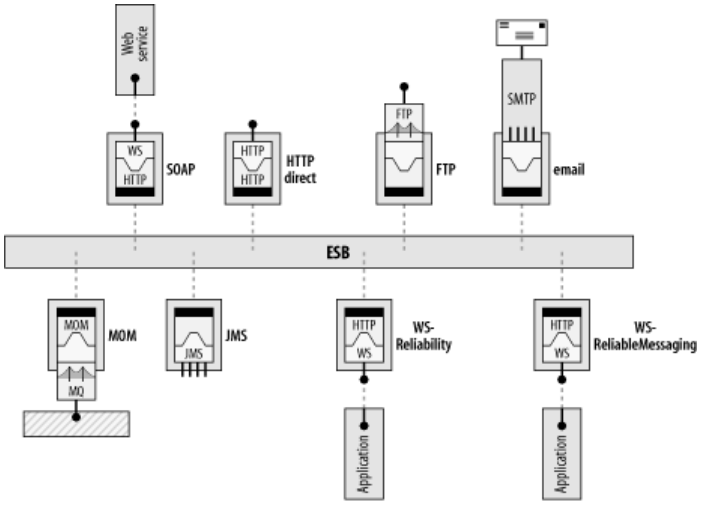
52

ESB Ingredients

- Provides a loosely coupled, highly distributed approach to integration
- An ESB consists (basically) of …
  - Message Oriented Middleware
    - Robust, reliable transport
    - Efficient movement of data across (abstract) channels
    - End-to-end reliability
  - Web Services
    - Service-oriented Architecture (SOA)
    - Abstract business services
  - Intelligent Routing
    - Based on content and context
    - Based on business process rules
  - XML data transformations
    - Based on XSLT and independent deployed services

53

The ESB MOM Core



54

ESB Capabilities

- Integrate across a common backbone to gain real-time access to the business data that is flowing between participants
- An app can rely on the ESB to provide a uniform, consistent approach to sending and receiving data
  - An app plugs into the bus
  - It posts data to the bus
    - It is responsibility of the ESB to get it where it needs to be
    - In the target data format that it needs to be in

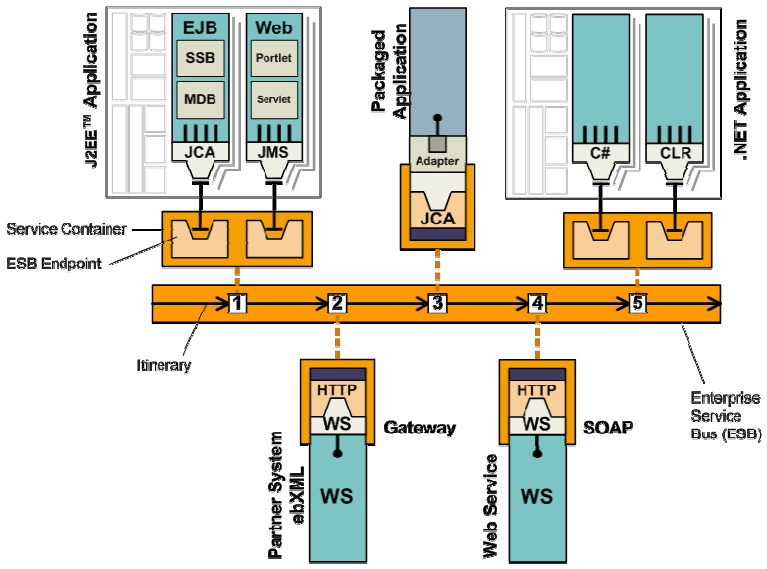
55

ESB Capabilities

- It separates the business process routing logic from the implementation of the apps that are being integrated
  - App owners do not need to worry about integration (e.g. routing, validation, …)
- Message itinerary very powerful and flexible
  - Through configuration and management tools, additional processing steps can be inserted as event-driven services
  - (processing pipeline)

56

ESB – Big Picture



57

ESB Characteristics

- ESB supports SOA across a series of abstract endpoints
  - Endpoints can represent apps and services of any size
- ESB is a managed environment
- ESB provides facilities that make the development, deployment and maintenance of integration services easy
- Unified view of security
- ESB MOM core as the basic infrastructure to build higher-level SOA
  - The details of creating and managing the messaging channels or setting QoS options are encapsulated in a contained-managed environment
    - Is configured instead of writing application code

58

Simplifying Integration

- Traditionally when dealing with MOM apps need to written code that
  - Establishes connection with the message server
  - Creates publishers, subscribers, queue senders and receivers, managing transactional demarcation and recovery from failures
- An ESB removes this complexity by delegating that responsibility to the ESB service container
  - Container relies on configurations

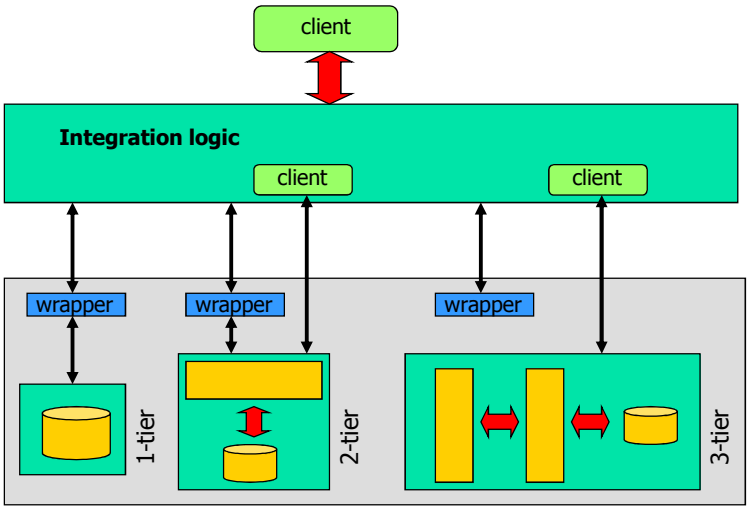
59

EAI and B2B - Wrap-up

- MW is about application integration
- EAI deals with (tightly) integration
  - EAI refers to software systems that facilitate the integration of heterogeneous, coarse-grained apps
  - More control over MW and processes
- B2B deals with loosely integration
  - Different approaches
  - Semantic heterogeneity
  - Web services
  - Protocols
- ESB makes integration pluggable

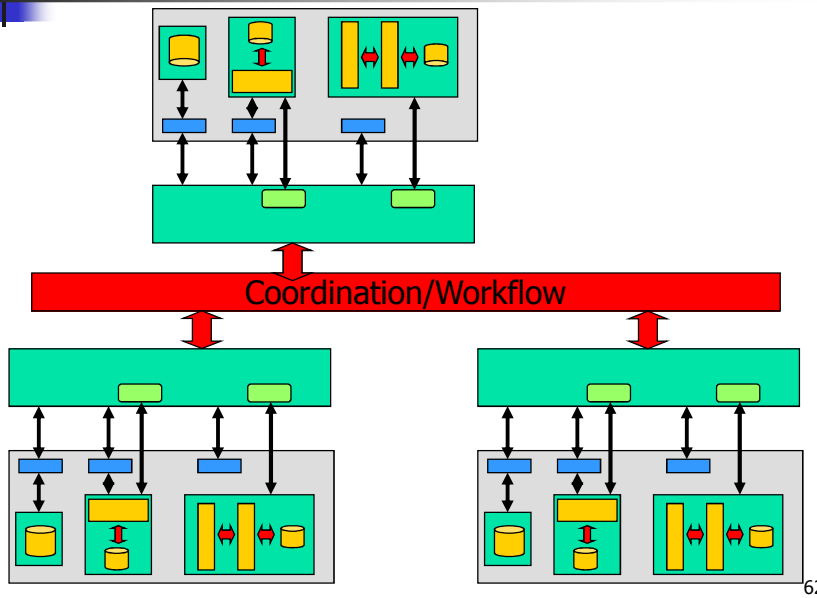
60

EAI

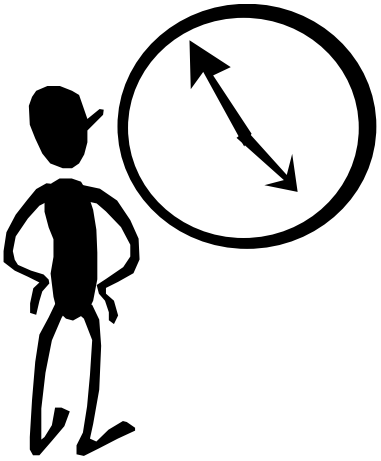


61

B2B



62



63