LECTURE 6:
ENTERPRISE APPLICATION INTEGRATION (EAI),
SOA, MICROSERVICES & MIDDLEWARE IN
ENTERPRISE ARCHITECTURE
Lecture Contents

• The world of Enterprise Information Systems (EIS)
• Intro to SOA & Middleware in EIS
  – EAI & Middleware and where this fits in to Business Processes
  – Evolution from EAI to SOA
  – Some theory of SOA
  – A SOA SCM example
  – Microservices
  – Middleware in Business Applications
Enterprise Information Systems in Brief

• **Enterprise information systems:**
  – Integrated ISs that support core BPs and functions.
  – e.g Marketing, Accounting, Finance, Info security, HR, Compliance, Production, Purchasing, and Logistics.

• Know them from terms e.g.
  – ERP: Enterprise resource planning
  – SCM: Supply chain management
  – CPFR: Collaborative planning, forecasting, and replenishment
  – CRM: Customer relationship management
  – KM: Knowledge management
  – BI: Business Intelligence, Data Analytics etc

• Main EIS need is for data integration (data sharing/ exchange):
  – e.g, ERP & SCM improve SC; KM & CRM for (un)profitable customers
  – All these are facilitated by communication!
The Lie of the Land...

- A layer-wise outline of some technologies & how they interrelate.
- Up to now, only three layers have been considered (mostly BP layer)
The Changing Context: Terminology

- Integrating enterprises’ existing IS applications to run BPs with many s/w systems has used **Enterprise Application Integration (EAI)**
  - User Interface Integration,
  - Data Integration
  - Method or Function Integration
  - Business Process Integration

- **Middleware** is comm facilitator in EAI often implemented by the **Enterprise Service Bus (ESB)** (like EAI message router).

*s/w layer allowing many systems to seem to users as a single coherent system with a variety of functionality.*
The Changing Context(/2): EAI & Middleware

- Where in multi-layer architecture is business-logic?
  - Can’t put in client (UI) tier
    - Leads to Fat client, reimplemented for each different client type
    - Redistributing clients after each software update
  - Not Data tier as different applications have different uses for same data
  - Has to reside on Middle Layer

- Enterprise Application Integration (EAI)
  - Integrates applications & enterprise data sources to share BP & data
  - Done without much changes of applications/ data sources
  - All data conversion, security, comms between computers is seamless
The Changing Context (/3): Challenges to BPM

• Methods of BPM are useful when optimising BPs within an enterprise.

• Some business environments require many different process designs
  – BP Mass-customization =>Automatic BP creation (eg patient health records)
  – BPs evolve dynamically as they execute, through the exchange of information among participants whose relationships evolve as a result

• But BPM is neither scalable nor adaptive by nature
  – BPM-based F/w to build business applications is too tightly coupled to adapt.
  – For each change, business dept must interact with IT dept to change software.

• Still need BPM as processes will need to be optimised
  – Need increased agility in BPs for loosely-coupled business networks

• SOA is a solution for this
Service-Oriented Architecture (SOA)

Ecosystem Evolution

**Component-Based Development**

- Messaging Backbone
  - Point-to-Point connection between applications
  - Simple, basic connectivity

**Enterprise Application Integration (EAI)**

- EAI connects applications via a centralized hub
- Easier to manage larger number of connections

**Service-Oriented Architecture**

- Integration of services thro an Enterprise Service Bus (ESB)
- Flexible connections with well defined, standards-based interfaces

**Flexibility**

- Different parties (even in the same company) may have different:
  - OS, interface, data format, infrastructure, interaction protocols, language, etc
- Automating Supply Chain Mgmt => bring all these together
- As seen, EAI currently solves this but evolution has been to SOA
  - Supports flexible s/w dvpt thro ‘loose service coupling’ => no need to talk to IT.

*Source: [IBM SOA]*

© IBM Corporation
What is Service Oriented Architecture (SOA)?

• Data & BP sharing between applications are EAI’s primary purposes.
  – Links enterprise applications to talk to one another & do “batch” data transfers
  – But EAI also defines principles for linking multiple systems, such as message-oriented middleware (MOM)
  – EAI is maybe old with SOA, but still EAI tools useful for large scale integrations

• SOA provides ‘transactional’ data transfer, needs no third-party s/w:
  – It differs from EAI in that it does not depend on a third-party solution.
  – Links interacting & contracted services via comms protocol (i.e. Web Services)

• Services are useful because they:
  – Are reusable in heterogeneous environments at multiple levels, including code, platform, so more flexible in the design of enterprise applications
  – Are implemented by 1/more code components in homogeneous environments
  – Aggregating 1/more components into a service, accessible through asynchronous messaging using open standards.
SOA in Practice: Example of SCM

- **Supply Chain Management**
  - No parts at Plant? ERP system messages HQ -> queries ERP system at other plant for item. None at HQ? HQ sends e-order to supplier’s ERP system
  - EA for inventory query/supply order:
    - EA needs 4 systems connected by 3 proprietary interfaces.
    - Mainframe at 1st plant connects to HQ’s Windows servers -> connects to 2nd plant’s IS & supplier Sun box.
    - As seen, this tightly coupled integration is inflexible & costly to modify/maintain.
    - E.g., in the EA, to add new suppliers, competitive bidding on supplier contracts are complex/expensive.
SOA in Practice: Example of SCM (/2)

- **B2B Commerce Facilitation with SOA**
- Converting to an SOA allows for B2B commerce without system reworking systems.
- As well as eliminating proprietary interfaces, SOA enables 1st plant to check directly with 2nd plant & place orders without need for HQ’s computer.
- HQ sees transactions with own WS & to-and-fro messages btw 2nd plant & supplier.
SOA in Practice: Example of SCM (/3)

- SOA increases B2B commerce by manufacturer holding competitive bidding system.
- Suppliers bidding to win business required to use WS to connect to bidding system.
- Again, can do with traditional technology but costs are so high that it’s rarely done.
- SOA allows manufacturer to manage suppliers' costs & suppliers can get business.
- Further, new/changed suppliers, IT responds quickly/cheaply to the business decisions.
Solving a SOA Problem: Microservices

• Unfortunately with Services, Goldilocks comes in again – what size?
• SOA have an *enterprise* scope, apps talking to one another.
• Services are exposed through standardized interfaces between apps.

• *Microservices* is another approach, breaking apps into smaller, completely independent parts
• This enables them to have greater agility, scalability, and availability.
• The microservices architecture has an *application* scope, with a focus on the structure and components *within* an application.
Monolith Vs Microservices

monolith

UI - team

SERVER - team

Dba - team

A product not a project

UI

dba

SERVER

UI

dba

SERVER

UI

dba

SERVER

microservices

Lecture 6: Technology Architecture: Middleware

CA4101 Lecture Notes (2019)
Characteristics of Microservices

- Main ones: loose coupling and high cohesion of services:
- Applications are made up of small independent services, functionality distributed among different services.
- Services independent, separate pieces of code each small complete functionality units.
- Services are independently modifiable and (re)deployable.
- Services communicate thro well-defined technology agnostic protocols REST (HTTP)
- Logic related to a particular service should be kept in a single place.
- Microservices based system architecture & continuous deployment is a match made in heaven.
- Decentralized data management: each service can have its own DB.
Advantages of Microservices

- No big team req’d to develop, each service can be modified independently.
- Developing, modifying, and deploying one service has no effect on other’s operation.
- No long-term commitment to a certain type of technology.
- Ease of integration with third party services and tools.
- Flexible use of languages, frameworks, DB within the same application.
- Easy to implement continuous deployment.
- Latest technologies can be adopted quickly.
- Fault isolations: 1 microservice crash does not make the entire application crash.
- Efficiently scalable (Only services in demand need scaling) and Load balancing.
- Easy to use with Docker (and other container technologies).
- No big design efforts upfront as facilitates early release, continuous iteration, flexible product evolution.

- Services based on Microservices
  - Netflix
  - Amazon
  - Twitter
  - Paypal
Disadvantages of Microservices

- Increased network latency.
- Increased processing time.
- Bug tracking can be tedious.
- Information barriers and communication difficulties.
- Developing a use case could need the cooperation of multiple teams.
- Developers can struggle to see the big picture.
- Communication between multiple teams can cause decreased productivity.
- Duplication of efforts and replication of functionalities.
- Moving an engineer to a new team could be problematic if they use a different stack.
SECTION 6.2: MIDDLEWARE IN DISTRIBUTED SYSTEMS
General Role of Middleware

• **Observation**
  – Role: to provide common services/protocols in Distributed Applications
  – Can be used by many different distributed applications

• **Middleware Functionality**
  – (Un)marshalling of data for transport to remote systems/apps
  – Naming protocols: to allow easy sharing, discovery of resources
  – Enforces business rules
  – Security protocols: for secure communication
  – Scaling mechanisms, such as for replication & caching (e.g. decisions on where to cache etc.)
  – Rich set of communications protocols: to allow some applications to transparently interact with others regardless of location.
Classification of Middleware

- Classify middleware technologies into the following groups:
  
1. **Bog-standard Sockets**
   - The basis of all other middleware technologies.

2. **RPC – Remote Procedure Call**
   - RPCs provide a simple way to distribute application logic on separate hosts
   - Allow one host to request a service from a host on another computer in a network without having to understand network details.

Stubs are pieces of code that can connect to other network procedures but pretend to be local procedure calls. Have to wrap/unwrap data/results.
Classification of Middleware (1/2)

3. **TPM - Transaction Processing Monitors:**
   - TPMs are a special form of MW targeted at distributed transactions.

4. **DAM - Database Access Middleware:**
   - DBs can be used to share & communicate data between distributed applications.
Classification of Middleware (/3)

5. Distributed Tuple:
   • Distributed tuple spaces implement a distributed shared memory space.
   • In practice this works like a DB, separating ‘sender’/’receiver’ in time

   ![Diagram of Distributed Tuple]

6. DOT (Distributed Object Technology)
   • Here both sender/receiver share an object which they both operate on.
   • Example of this is Enterprise Service Bus

   ![Diagram of DOT]
Classification of Middleware (/4)

7. **MOM (Message Oriented Middleware):**
   - In MOM, messages are exchanged asynchronously between distributed applications (senders and receivers).

8. **Web services:**
   - Web services provide access to services via a defined interface, typically accessible through the web protocol HTTP.
Summary of Communications Middleware

• Essentially a range of types of communications middleware
• All can be used to implement others, all are suited to different cases
  – All carry some payload from one side to another <with details>
  – Some of these payloads are ‘active’ and some are ‘passive’
  – Also differ in granularities and whether synchronous or not.

Message-Oriented Middleware

Sockets <Pkt1, Pk2>

RPC <Args>

DOT <Obj>

TPM

DTM <{}, {,}>
References


