Enterprise Application Integration (Middleware)

Gustavo Alonso
Systems Group
Computer Science Department - ETH Zurich
alonso@inf.ethz.ch
http://www.systems.inf.ethz.ch/
EAI Course Administration

- Lecture: Tuesdays 13.15 - 15:00 – CAB G 11
- Discussion and Exercises: Thursdays 08:00 – 10:00 – CAB G 11
- Web site
  - [http://www.systems.ethz.ch/courses/fall2012/EAI](http://www.systems.ethz.ch/courses/fall2012/EAI)
- Getting in touch with us:
  - Gustavo Alonso  
    alonso@inf
  - Pratanu Roy  
    pratanu@inf
  - Dimitrios Karampinas  
    dkarampi@inf
- Practical exercises:
  - Based on reading assignments
  - 3 written reports during the semester
  - Apply the concepts of the lecture, see their use in practice
  - Reports are mandatory (topics are included in the exam)
- Course material:
  - Foils of the lecture (download from the course website)
  - Additional documents (see web site)
- Exam:
  - Written exam, during exam session
Course outline

- Introduction – Enterprise Architecture
- Layers of an enterprise IT system
- Basic components and patterns of an EAI platform
- Synchronous interaction patterns – Traditional Middleware
- Transactional support – TP Monitors
- Asynchronous interaction patterns – Messaging
- Cloud computing in perspective
- Web Services
- SOAP
- WSDL / UDDI
- Messaging
- Business processes – Workflow and orchestration - BPEL
- Service Oriented Architectures
The course:
Enterprise Application Integration
Enterprise Architecture
What is enterprise architecture?

**Enterprise Architecture** is the practice of applying a comprehensive and rigorous method for describing a current and/or future structure and behavior for an organization's processes, information systems, personnel and organizational sub-units, so that they align with the organization's core goals and strategic direction. Although often associated strictly with information technology, it relates more broadly to the practice of business optimization in that it addresses business architecture, performance management, organizational structure and process architecture as well.


- Our focus will be on information technology
- The course is an introduction to the topic
  - Many sub-topics need a course on their own
  - Additional courses available (Web Services and SOA in Spring Semester)
- Goals for the course:
  - Understand what is enterprise architecture
  - Known and understand the basic technology
  - Become familiar with basic design patterns and systems
Enterprise architecture at Credit Suisse

Multiple backends, multiple frontends, flexible composition

Portal Platform

Multi Channel Platform

Service Oriented Architecture

Security Entry Infrastructure

Design and Development Platform

IT Operations and Platform Management

Hardware / System Software

Platform Security

Graphic courtesy of Claus Hagen, Stephen Murer and Hanspeter Uebelbacher of Credit Suisse
Enterprise Application Integration (EAI) is defined as the uses of software and computer systems architectural principles to integrate a set of enterprise computer applications.


- Can be seen as the techniques necessary to build the IT part of an enterprise architecture
- The key word is “integration”
- The course will cover EAI extensively:
  - Technologies
  - Design patterns and architectures
  - Tools and systems
- Goals of the course:
  - Know the technology, what it does and why
  - Know the design options and how to choose between them
  - Become familiar with the common problems of EAI
Integration Infrastructure - Credit Suisse

CS EventBus Infrastructure (Message based)

Bulk integration infrastructure (File Transfer)

Graphic courtesy of Claus Hagen, Stephen Murer and Hanspeter Uebelbacher of Credit Suisse
The course:
Basic concepts
 Architectural patterns

- We will use architectural patterns to capture existing solutions
  - As an abstract concept
  - As a general configuration
  and to study
  - General properties
  - Design options
  - Constraints and limitations

- Patterns have to be used with care in enterprise architecture (they are far more general and vague than in the case of development patterns)

- Yet, they are useful to capture existing knowledge. The course will introduce you to basic design and architectural patterns and their general properties.
Example – application and runtime

<table>
<thead>
<tr>
<th>APPLICATION PATTERN</th>
<th>DEMILITARIZED ZONE (DMZ)</th>
<th>INTERNAL NETWORK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside World</td>
<td>Protocol Firewall</td>
<td>Domain Firewall</td>
</tr>
<tr>
<td>User</td>
<td>Web Server Redirector</td>
<td>Application Server</td>
</tr>
<tr>
<td>Public Key Infrastructure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain Name Server</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RUNTIME PATTERN

Directly Integrated Single Channel application

Presentation - Application - Application

Existing Applications and Data

Directory and Security Services
Example – product mapping

We will not cover this in the course
Enterprise Service Bus
SOA Governance
SOA governance introduces different aspects to standard software engineering concepts:

- Service definition (the scope, interface, and boundaries of a service)
- Service deployment lifecycle (the lifecycle stages)
- Service versioning (including compatibility)
- Service migration (deprecation and sunsetting)
- Service registries (dependencies)
- Service message model (canonical data models)
- Service monitoring (problem determination)
- Service ownership (corporate organization)
Integration vs programming

<table>
<thead>
<tr>
<th>Level</th>
<th>Scope</th>
<th>Detail</th>
<th>Impact</th>
<th>Audience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agency/Organization</td>
<td>Low</td>
<td>Strategic Outcomes</td>
<td>All Stakeholders</td>
<td></td>
</tr>
<tr>
<td>Line of Business</td>
<td>Medium</td>
<td>Business Outcomes</td>
<td>Business Owners</td>
<td></td>
</tr>
<tr>
<td>Function/Process</td>
<td>High</td>
<td>Operational Outcomes</td>
<td>Users and Developers</td>
<td></td>
</tr>
</tbody>
</table>

From US OMB 2006 FEA Practice Guidance
Two examples

- Why using a database?
  - 70 - 90% of the software costs are maintenance costs.
  - Databases used as services remove about 40% of the code of commercial applications.

- Software as a Service (SaaS):
  - Completely outsource the running of some of the software needed in a company.
  - Rental and use cost lower than the costs of running the software locally.
Enterprise Architecture cycle

- Enterprise Architecture is a complex process
- Many frameworks exist to develop and maintain an enterprise architecture (e.g., TOGAF)
- In the course we will focus in the Information System Architectures and Technology Architecture phases
- We will do this by studying:
  - Architectures
  - Architectural patterns
  - Systems
  - Tools
  - Basic components

The Open Group Architecture Framework
Architecture Development Method (TOGAF ADM)
The course: Basic technology
Scale-up versus Scale-out

- Scale up is based on using a bigger computer as the load increases. This requires to use parallel computers (SMP) with more and more processors.

- Scale out is based on using more computers as the load increases instead of using a bigger computer.

- Both are usually combined! Scale out can be applied at any level of the scale up.

Diagrams courtesy of Jim Gray, Microsoft
Big computers

- Modern computers:
  - Multi-core: scale up
    - Multitenant -> virtualization
    - More resources available (memory)
    - More can be done inside a single machine
  - Clusters: scale out
    - Main memory systems
    - High degrees of parallelism
    - Commodity hardware
Clusters

- Computer clusters require:
  - Commodity hardware (to scale out to large numbers without incurring prohibitive costs)
  - Parallel computation models (to exploit the cycles available)
  - Mechanisms for fault tolerance and dynamic provisioning (to cope with the constant failures that occur at these scales and for this hardware)

- The biggest problem is the network!!
### Parallel Computation models

<table>
<thead>
<tr>
<th><strong>MapReduce</strong></th>
<th><strong>Parallel Databases</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>- No rigid schema</td>
<td>- Pre-defined schema</td>
</tr>
<tr>
<td>- No built-in indexes</td>
<td>- Hash and B-tree indexes</td>
</tr>
<tr>
<td>- Low-level programming (fully general)</td>
<td>- SQL-based programming (extensible with UDFs)</td>
</tr>
<tr>
<td>- No automatic support for data distribution and skew</td>
<td>- Optimizations sensitive to data distribution and skew</td>
</tr>
<tr>
<td>- Pull-based transfer of intermediate data (materialized)</td>
<td>- Push-based transfer of intermediate data (non-materialized)</td>
</tr>
<tr>
<td>- Finer-grained failure handling</td>
<td>- Transaction-level f. h.</td>
</tr>
<tr>
<td>- Novel methods to deal with slow nodes and load imbalance</td>
<td>- No special methods for slow nodes</td>
</tr>
</tbody>
</table>
The Modern Infrastructure: The Private Cloud

- **Sprawled**: Component-Orientation
- **Virtualized**: Layer-Orientation
- **Automated**: Service-Orientation

- **2002**
- **2002 to 2012**
- **2010 to 2020**

- Peak provisioning
- Costs of large installations
- Need for consolidation
- Computing becoming invisible

Real-Time Infrastructure

- Policies
- Optimization
  - Workloads
  - Data
  - Resources
  - Identities

- Provisioning
- Services
- Availability

© Gustavo Alonso, Systems Group, ETH Zurich