# SWENG568: Enterprise Integration

## *Lesson 1: Informatioon Silos, Integrated Systems, and Business Intelligence*

Introduction (1 of 10)

***The only constant is change.***

***— Heraclitus***

## Learning Objectives

- Understand the evolution of computing and networks technologies
- Know the importance and challenges of enterprise integration
- Master basic concepts: service, information silos, packaged applications, integration, middleware, integrated systems, and business intelligence

By the end of this week, make sure you have completed the readings and activities found in the Lesson 1 Course Schedule.

The Evolution of Computing and Inter-application Communication (2 of 10)

## The Evolution of Computing and Inter-application Communications

The term *computing* might have different meanings to different persons who have different backgrounds and work in different fields. According to Wikipedia [1]:

> [C]omputing includes designing and building hardware and software systems for a wide range of purposes; processing, structuring, and managing various kinds of information; doing scientific studies using computers; making computer systems behave intelligently; creating and using communications and entertainment media; finding and gathering information relevant to any particular purpose, and so on. The list is virtually endless, and the possibilities are vast.

It has sometimes been narrowly defined while fitting well in this course, as in a 1989 ACM report on *Computing as a Discipline* [1]:

> The discipline of computing is the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application. The fundamental question underlying all computing is 'What can be (efficiently) automated?

Essentially, computing is what and how software or computer programs can create values that meet the desires of end users. The desires include improved productivity, reduced labors, better quality, enhanced safety features, and a variety of supporting services, etc. A module is typically considered as an individual computing component, and a computer program is an aggregated and integrated whole from computing components. A piece of software consisting of numerous computations can be simply illustrated as follows:

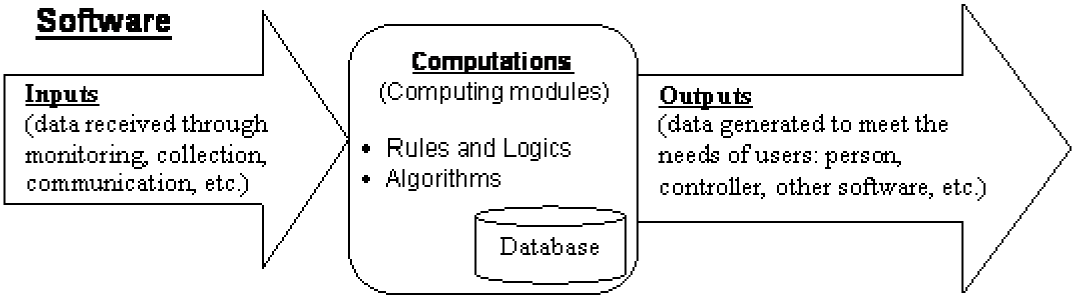

Software
Inputs
(data received through monitoring, collection, communication, etc.)
Computations
(Computing modules)
• Rules and Logics
• Algorithms
Database
Outputs
(data generated to meet the needs of users: person, controller, other software, etc.)

Figure 1.1: A Schematic View of Software

The development of software requires a rigorous software engineering process. *Programming* or *coding*, an essential part of software development, is the process of writing, testing, debugging, and maintaining source codes to enable a list of desired computations. The source code is written in a programming language [1], for instance C++ or Java. The following figure shows the evolution of program languages.

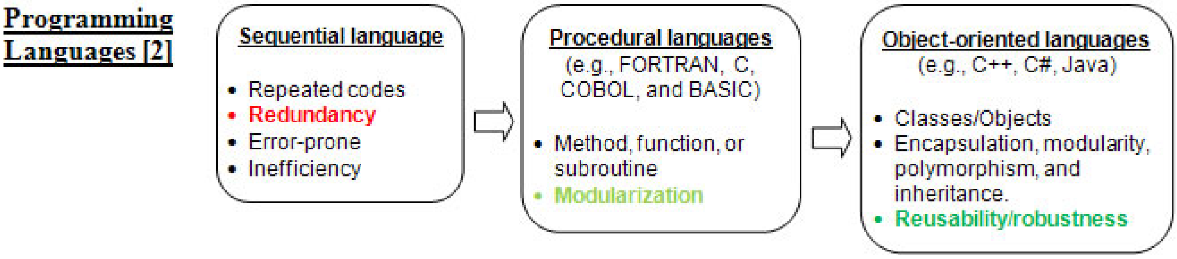

Programming Languages [2]
Sequential language
• Repeated codes
• Redundancy
• Error-prone
• Inefficiency
Procedural languages
(e.g., FORTRAN, C, COBOL, and BASIC)
• Method, function, or subroutine
• Modularization
Object-oriented languages
(e.g., C++, C#, Java)
• Classes/Objects
• Encapsulation, modularity, polymorphism, and inheritance.
• Reusability/robustness

Figure 1.2; The Evolution of Programming Languages

What is a Service? (3 of 10)

# What is a Service?

Service is broadly considered as an application of specialized knowledge, skill, and experience, performed for co-creation of respective values of both the consumer and the provider [3].

As discussed in the textbook [2], when the code written in FORTRAN, C, or the like that needed to be repeated was separated out as a simple procedure, such as a method, function, or subroutine, the most rudimentary form of software service was created. As the concept evolves, a (software) service is essentially a computation that is successfully completed by other module, program or packaged application, internally or externally, locally or remotely.

Software Quality Requirements (4 of 10)

# Software Quality Requirements

According to Wikipedia [1], software might be developed using a variety of approaches, however, the final delivered program must satisfy some fundamental quality requirements to satisfy the user's needs. The most relevant ones include:

- ***Efficiency/Performance***
- ***Reliability***
- ***Robustness***
- ***Usability***
- ***Portability***
- ***Maintainability***.

Distributed Computing (5 of 10)

# Distributed Computing

Why do we need distributed computing?

The term *distributed computing* is more or less referring to autonomous computing processes that might run on the same physical computer or separated while networked computers and interact with each other by message passing. Different from parallel computing, each process in distributed computing has its own private memory (or distributed memory), where data/information for computations is exchanged by passing data between the processes (as shown in the following diagram).

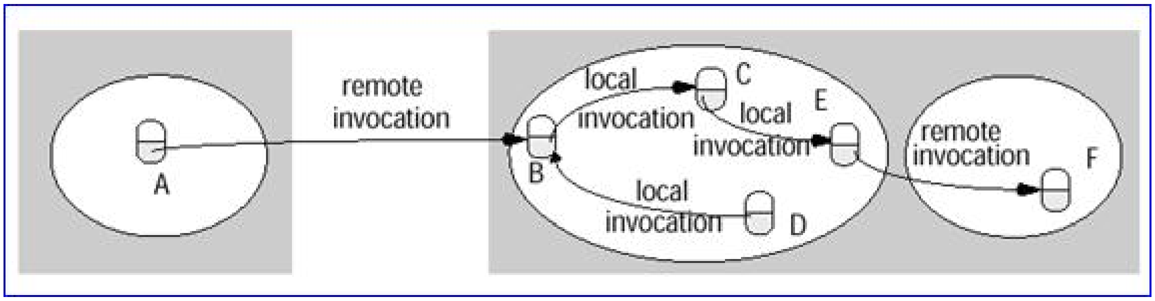


Figure 1.3 A Schematic View of Local and Remote Services

In Figure 1.3, each bubble represents a computing process. Module A is running on one computer, while Module B-F are running on a different computer. On the second computer, Module F is in one process, and Module B-E are in another process. Apparently, Module A is requesting a remote service provided by Module B.

Network communications enable data/information exchanged between remote processes. It is the socket as the rudimentary connectivity that forms the fundamental concept while laying the foundation of all other types of inter-application patterns (Figure 1.4).

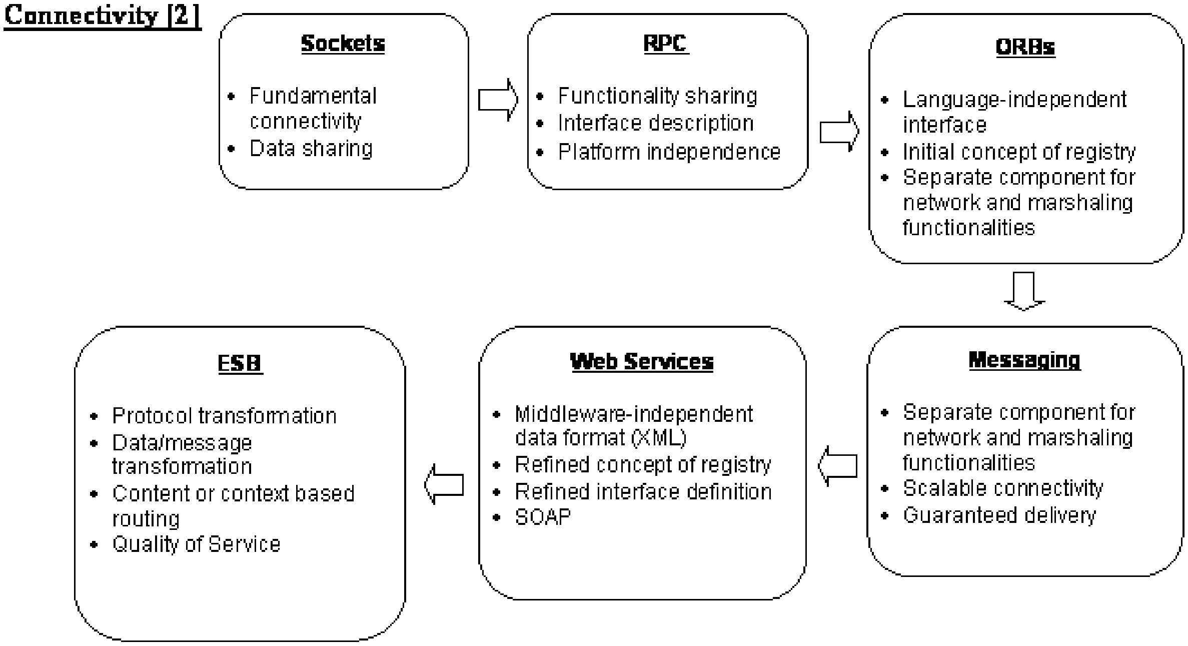
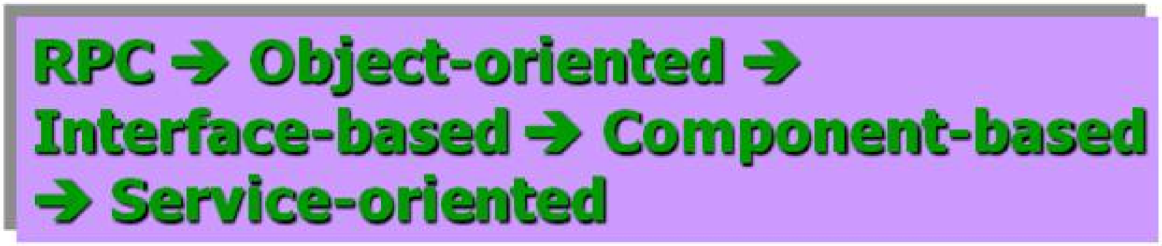

Connectivity [2]
Sockets
Fundamental connectivity
Data sharing
RPC
Functionality sharing
Interface description
Platform independence
ORBs
Language-independent interface
Initial concept of registry
Separate component for network and marshaling functionalities
Messaging
Separate component for network and marshaling functionalities
Scalable connectivity
Guaranteed delivery
Web Services
Middleware-independent data format (XML)
Refined concept of registry
Refined interface definition
SOAP
ESB
Protocol transformation
Data/message transformation
Content or context based routing
Quality of Service

Figure 1.4: The Evolution of Distributed Computing: The Connectivity Perspective

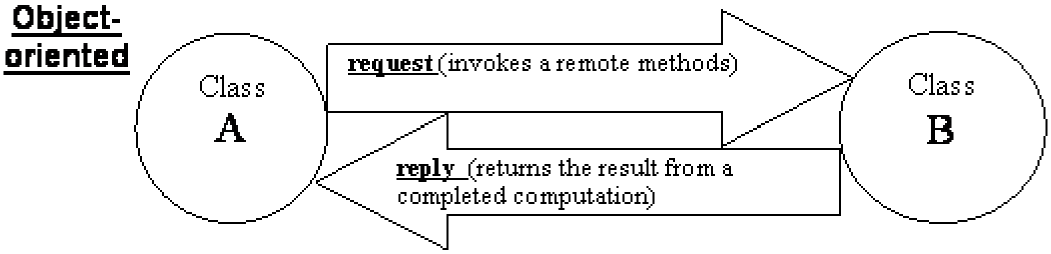
The evolution of distributed computing (Figure 1.5) gradually improves reusability of software services by technically enabling loose coupling and transparency.

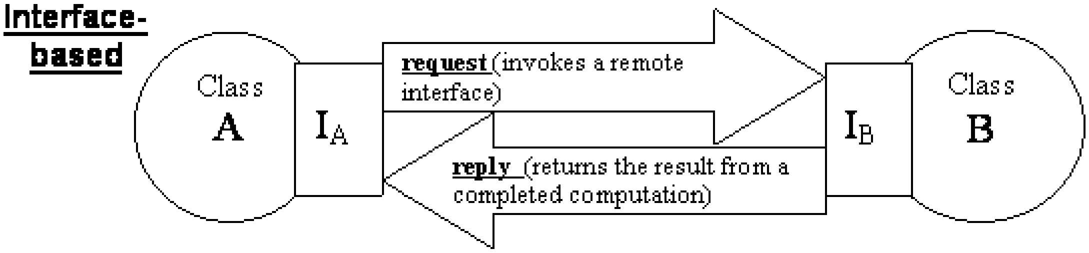


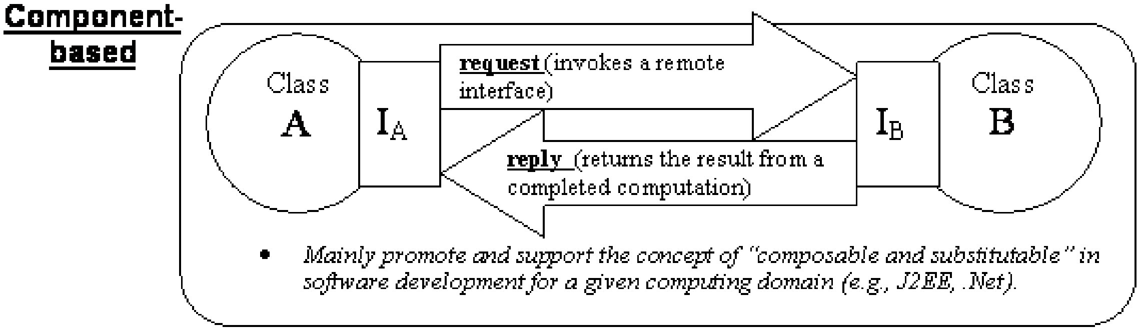
RPC → Object-oriented → Interface-based → Component-based → Service-oriented

Figure 1.5: The Evolution of Distributed Computing: The Programming Perspective

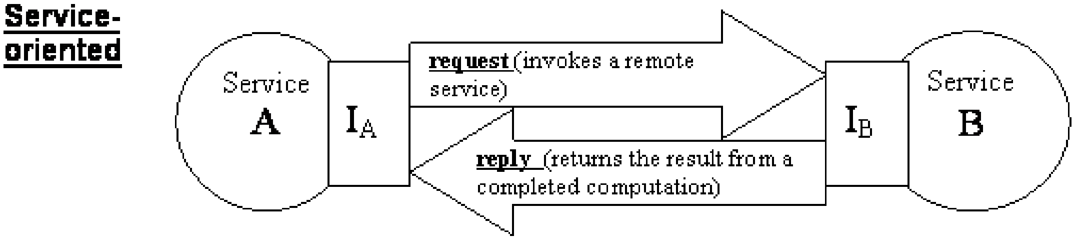
What are the differences between these programming models?


Object-oriented
Class A
request (invokes a remote methods)
reply (returns the result from a completed computation)
Class B


Interface-based
Class A
I$_A$
request (invokes a remote interface)
reply (returns the result from a completed computation)
I$_B$
Class B


Component-based
Class A
I$_A$
request (invokes a remote interface)
reply (returns the result from a completed computation)
I$_B$
Class B

- *Mainly promote and support the concept of "composable and substitutable" in software development for a given computing domain (e.g., J2EE, .Net).*


Service-oriented
Service A
I$_A$
request (invokes a remote service)
reply (returns the result from a completed computation)
I$_B$
Service B

- Focusing on the delivery of language-, platform-, and middleware-independent (information) services

The Importance and Challenges of Enterprise Integration (6 of 10)

# The Importance and Challenges of Enterprise Integration

With the significant advances in technologies, people, organizations, and heterogeneous (information) systems now can be linked together more efficiently and cost-effectively than ever before. The quick advances of information technology (IT) in general significantly transforms not only science and engineering research but also expectations on how people live, learn, and work as we witnessed during last decade or so. Life at home, work, and leisure gets easier, better, and enjoyable.

In the business world, because of the rich information linkages the right data and information in the right context could be delivered to the right user (e.g., people, machine, device, software component, etc.) in the right place, at the right time, resulting in the substantial increase of the degree of business process automation, continual increment of production productivity and services quality, reduction of services lead time, and improvement of end users satisfaction. As a variety of devices, hardware, and software become network aware, almost everything is capable of being handled over the network. Obviously, It is Enterprise Integration that would enable the delivery of the right information to the right user in a timely manner, giving businesses a competitive edge in today's global market.

The Business Need of Enterprise Integration (7 of 10)

# The Business Need of Enterprise Integration

Business situations nowadays, different from one not long ago, change very quickly. The following five reasons mainly contribute to the changes:

- Business mergence
- Business acquisition
- Dynamically changing market conditions (due to the globalization and changing customer needs)
- Quick technology advancement
- The nature of business relationship (for competitiveness)

The Challenges of Enterprise Integration (8 of 10)

# The Challenges of Enterprise Integration

The vast technology landscape, numerous legacy applications, heterogeneous data sources, and changing business environments are continuously adding complexities and challenges to enterprise-wide integration projects. In general, the following issues are commonly and frequently confronted by the integration practitioners:

- **Complexity:** Corporate best practices capable of accommodating the needs of different cultures have been embodied in information systems through complex and dynamic business processes.
- **Diversity:** Operational and information needs have been diversified from department to department, facility to facility, and corporate to corporate.
- **Heterogeneity:** Information silos have been developed using different tools and methodologies, written in different programming languages, and run on different operating systems and communications networks.
- **Scalability:** An integrated system has to be scalable in terms of continuously delivering the expected information services when the system and supported business grow with time.
- **Agility:** An integrated system has to be flexible, responsive, and adaptable as market demands fluctuate and technologies advance.

Given the ever-changing business environments and technical challenges of integrating information silos, how to maximally leverage your IT investment becomes the ultimate goal of integration practitioners. Correspondingly, integration practitioners are increasingly paying attention to the following industrial best practices when working on integration projects:

- The efficiency and cost effectiveness are the result of the reusability.
    - Loose coupling
    - Implementation and delivery transparency
    - Standardization
    - Off-the-shelf solution

- Services most likely come from heterogeneous information systems.
- Enterprise integration becomes essential for business successes in the global marketplace.
    - Focusing the potentials of (nearly) real-time services
    - Providing services in a consistent manner
    - Meeting different needs of different users, regionally and globally

Basic Concepts - Loose vs. Tight Coupling (9 of 10)

# Basic Concepts - Loose vs. Tight Coupling

The textbook has a very comprehensive discussion on these concepts. You should read the textbook very carefully and then be actively involved in next online discussion forum.

| Level/Category | Loose Coupling | Tight Coupling |
|---|---|---|
| **Physical Connection** | Indirect connection through an intermediary | Direct connection |
| **Communication Style** | Asynchronous | Synchronous |
| **System Type** | Weakly typed system | Strongly typed system |
| **Interaction Pattern** | Distributed logic | Centralized logic |
| **Service Binding** | Dynamics binding | Static binding |

Table 1.1: Loose vs. Tight Coupling

References (10 of 10)

# References

[1] Wikipedia:

- Computing: http://en.wikipedia.org/wiki/Computing,
- Computer Programming: http://en.wikipedia.org/wiki/Computer_programming,
- Distributed Computing: http://en.wikipedia.org/wiki/Distributed_computing,
- Object-oriented Programming: http://en.wikipedia.org/wiki/Object-oriented_programming,
- Interface-based Programming: http://en.wikipedia.org/wiki/Interface_based_programming,
- Component-based Programming: http://en.wikipedia.org/wiki/Component-based_software_engineering,

- Service-oriented Programming: http://en.wikipedia.org/wiki/Service-oriented_programming,
- Information Technology: http://en.wikipedia.org/wiki/Information_technology,
- Information Systems: http://en.wikipedia.org/wiki/Information_Systems,
- Enterprise Integration: http://en.wikipedia.org/wiki/Enterprise_integration,
- Software as a service (SaaS): http://en.wikipedia.org/wiki/Software_as_a_service.

[2] Roshen, W. 2009. SOA-based Enterprise Integration: A Step-by-Step Guide to Services-based Application Integration. McGraw-Hill, New York, USA.

[3] Qiu, R. 2009. Computational Thinking of Service Systems: Dynamics and Adaptiveness Modeling. Service Science, Vol. 1, No. 1, 2009, 42-55.

[4] Dalbey, J. Pseudocode Standard. http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html.

Please direct questions to the World Campus HelpDesk (http://student.worldcampus.psu.edu/student-services/helpdesk) |