PENNSTATE World Campus

SWENG568: Enterprise Integration

Lesson 6: Integrating Existing Applications

Learning Objectives

- Understand a variety of levels of approaches to integrating existing applications
- Master technical fundamentals of data-, method-, API-, and process-level integrations
- Get familiar with workflow- and process-aware types of integration

By the end of this week, make sure you have completed the readings and activities found in the Lesson 6 Course Schedule.

The Business Perspective of Integration

As we discussed in Lesson 1, because of the rich information linkages the right data and information in the right format in today's business world could be delivered to the right user (e.g., people, machine, device, software component, etc.) in the right place, at the right time (Figure 1). Ultimately, right enterprise integration would result 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.

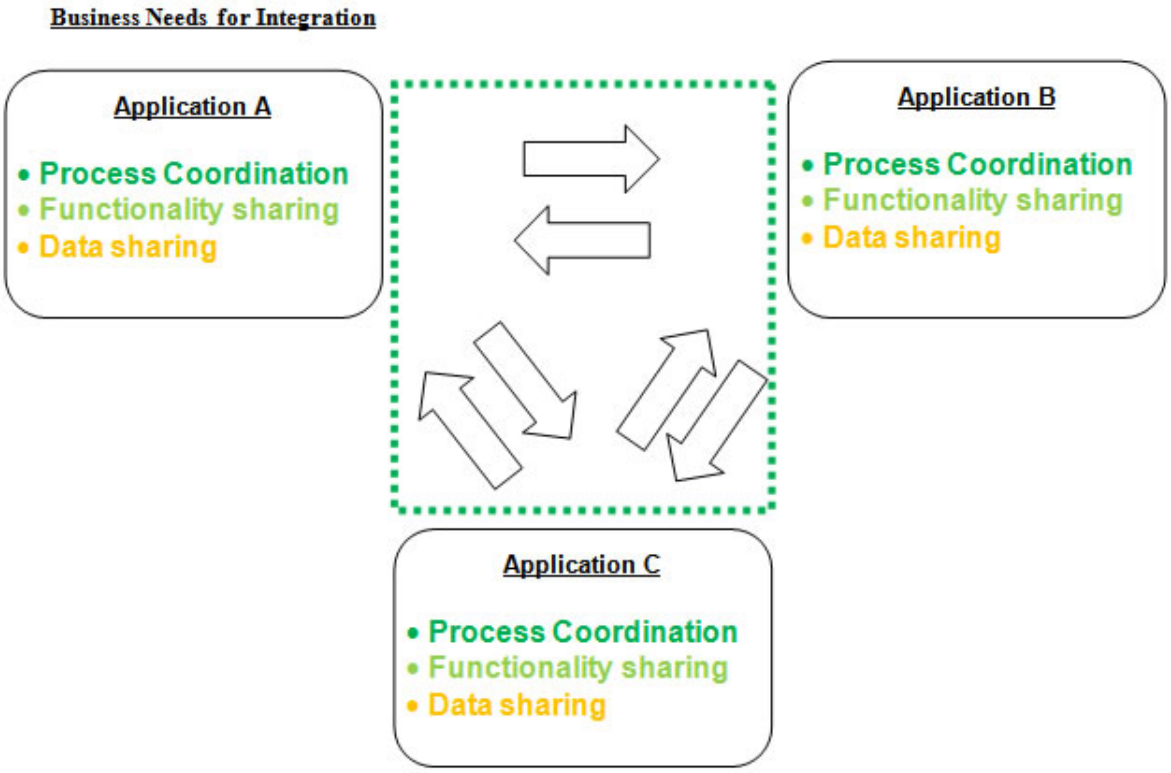


Figure 6.1 Business Needs for Integrations

As seen in Figure 6.1, it would be the best scenario if all applications can have their supported processes coordinated to function as a whole to serve the business needs throughout an enterprise. However, at a given time, some existing applications might be simply required for sharing data, while other existing applications demand functionality sharing. Technically, appropriate and implementable integration most likely depends on how all the functionality/supports are implemented and what integration supports are provided/enabled in a given application. Therefore, different levels of integration will be needed to meet the different needs, operationally and technically (Figure 6.2 and Figure 6.3) [2].

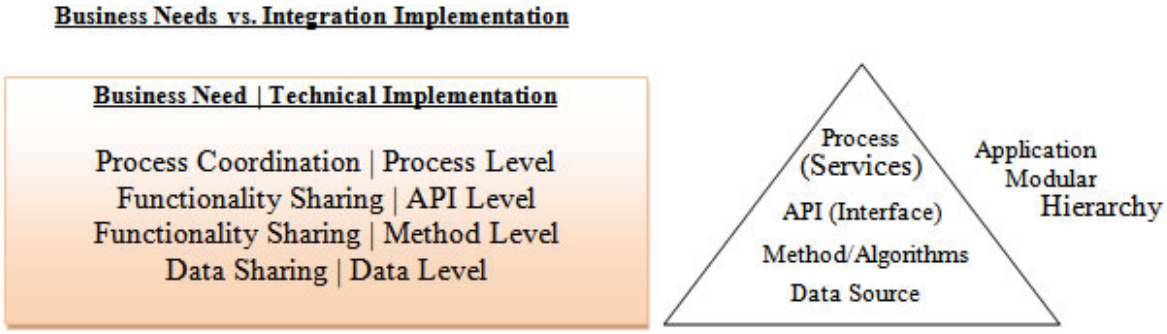


Figure 6.2 Business Needs vs. Integration Approaches

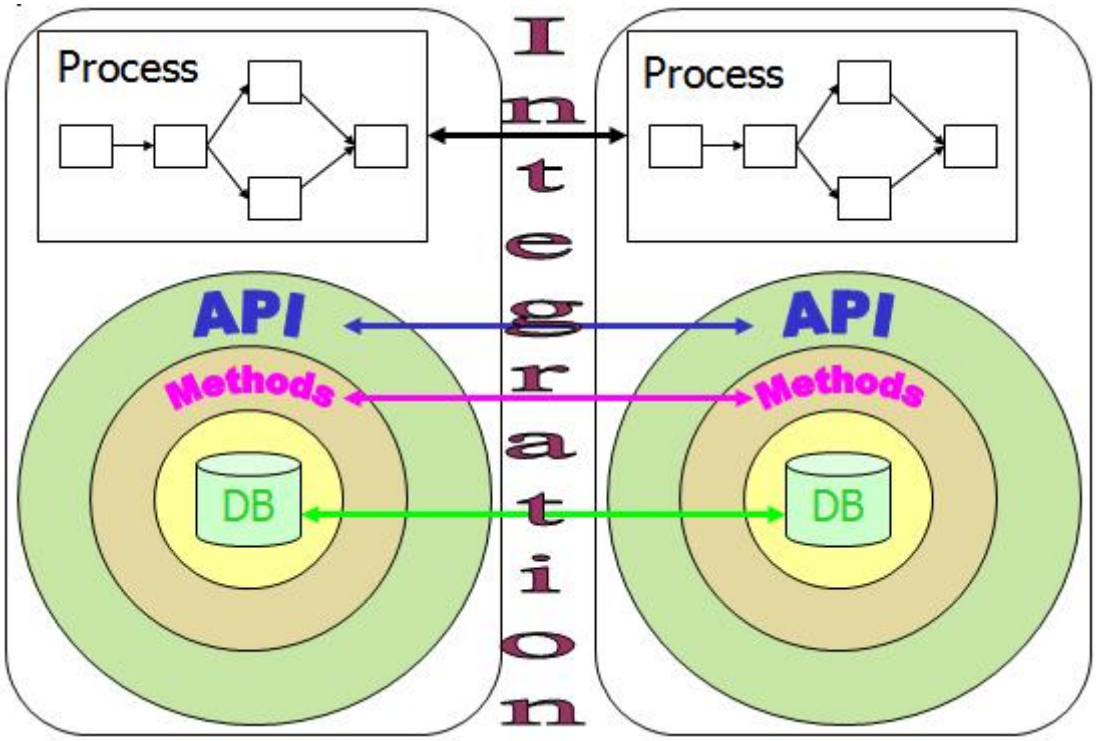


Figure 6.3 Graphical view of integrating methods at different levels between applications

The State-of-the-Art Connectivity Perspective of Integration (3 of 8)

The State-of-the-Art Connectivity Perspective of Integration

Without question, the quickly advanced ESB technologies can significantly ease the headaches of enterprise integration. As discussed in last lesson (Figure 6.4), state-of-the-art ESBs essentially combine the strengths of all other middleware, aimed at providing a solution to the various heterogeneity problems found in a large intra- and inter-enterprise setting. By fully leveraging the supports of standardized interfaces (i.e., IDL or the like), peer-to-peer relationship between applications, generalized stub/proxy/skeleton or adaptor which shields the programmer from system and network calls, marshalling/unmarshalling of arguments for transmission over the network, external data representation (XDR) which encodes data in a machine-independent format, synchronous and asynchronous interaction, language- and platform-independent programming, transformation, and content- and context-based routing, enterprise integration gets theoretically simplified, scalable, manageable, and adaptable.

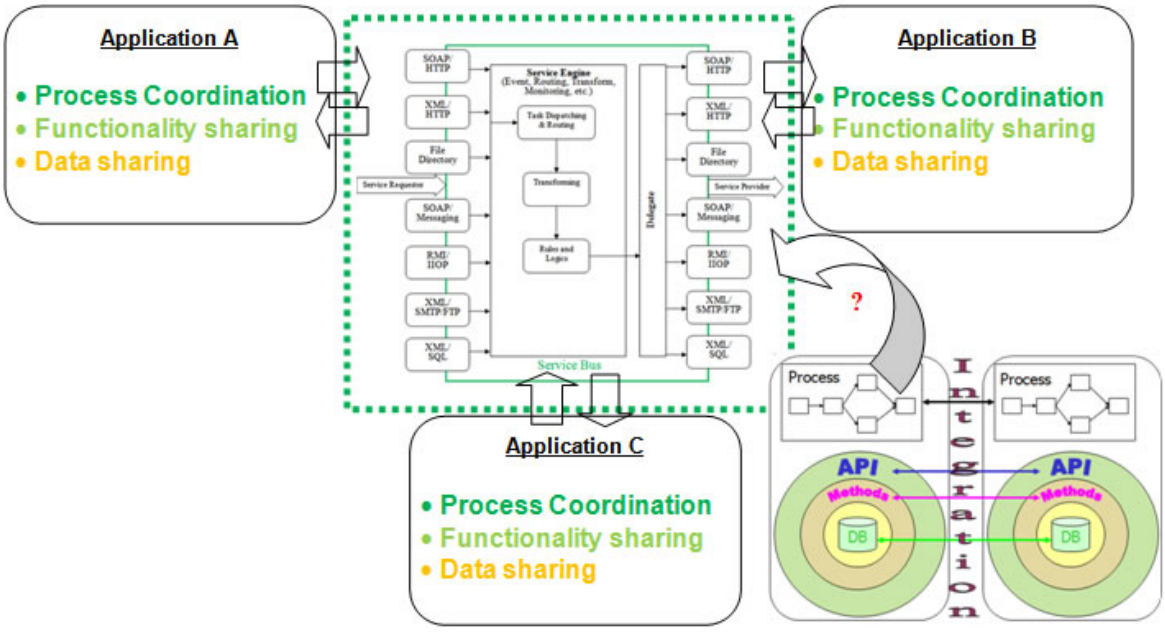


Figure 6.4 Integrating existing applications using ESBs

In practice, the question becomes how different levels of integration would be implemented given that there are many different types of ESBs. A typical level of integration may make more sense than others for a given business need under a given circumstance. Next section we will discuss these levels of integration in a great detail by focusing on their technical fundamentals.

Technical Fundamentals of Data-, Method-, API-, and Process-level Integrations (4 of 8)

Technical Fundamentals of Data-, Method-, API-, and Process-level Integrations

With or without middleware, traditional middleware or state-of-the-art ESBs, small- or large-scale EBSs, data-, method- and API-level integrations have been the most popular approaches used in addressing the challenges of legacy integration (Figure 6.5) [2]. This trend is gradually shifting as the process-level integration is recently gaining the popularity (detailed discussion will be in Lesson 7).

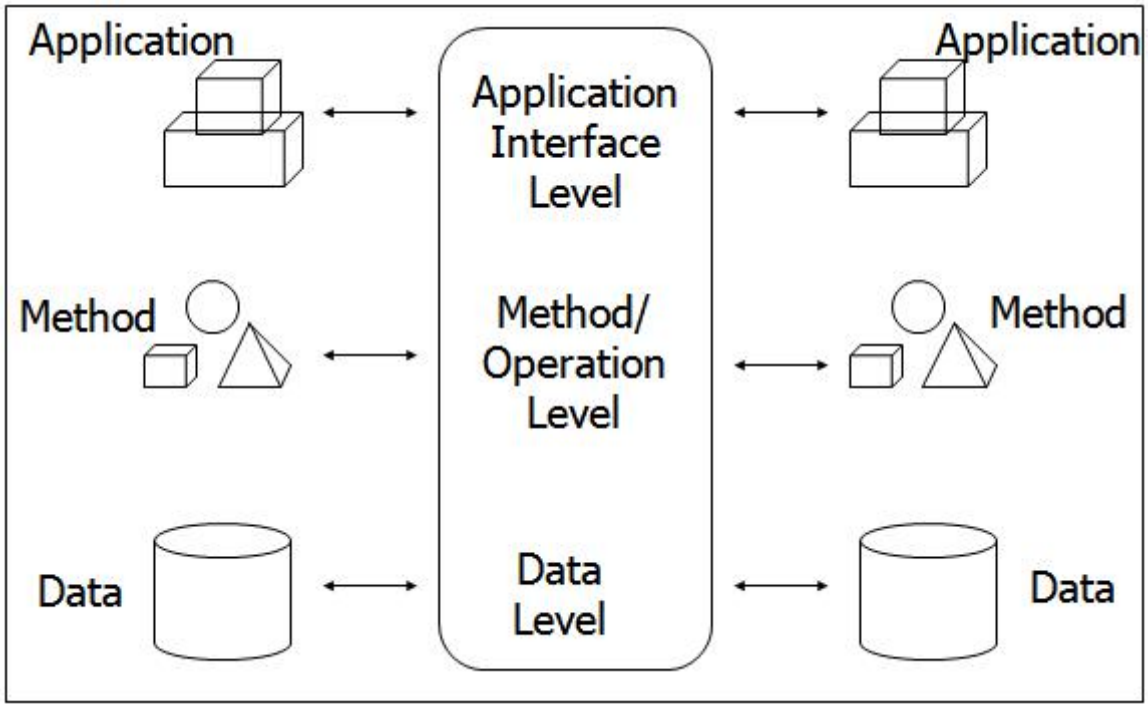


Figure 6.5 Three popular levels of integrations between applications

Data-level Integration

Regardless of how an application is technically implemented on each side of applications that are required for integration, if data sharing is the essential business need of the required integration and data sources are accessible to the integration engineers, a data-level integration scheme might be a more cost-effective choice as data-level integration is fairly straightforward and quick to implement.

Figure 6.6 shows a simple example of data-level integration. Please refer to Figure Ex1.1 and Ex 1.2, assume Roster System and Course System are required for meeting the following temporarily operational needs:

- Student contact information should be synchronized to avoid the discrepant information due to the fact that updating actions are allowed in both applications.
- As soon as a student finished his/her registration in the Course System, the registration data should be delivered to the Roster System. So the student invoice can be created.
- In case a student fails to pay the bill for two successive semesters, the student is not allowed for a new registration.

Figure 6.6 Example of data-level integration

Apparently, to capture and move the appropriate data from a source system to a target system is what this integration needs. The integration can be done by relying on all different means (database utilities, middleware, an ESB, or in-housing data transformer). The challenges of this type of integration include the following:

- Source and target systems may have different structure or format;
- An good understanding of data source (database, etc.) technologies is necessary;
- Data manipulation (e.g., inserting a new record or updating/deleting a record when triggers or macro-functions are heavily used by the database) might affect the implied information flow in a given application, which could complicate the integration implementation.

Although data-level integration is fairly straightforward and quick to implement, it has disadvantages that include increased data coupling between applications, the inability to access important application behavior such as data validation and critical business rules, and the need to write significant and proprietary data cleansing/formatting code if poorly designed data exists. Note that XML technology is now frequently used for this type of integration because XML is a platform-independent approach for sharing data.

There are many good packages (free or commercial) focusing on data integration [4], light-weighted (e.g., DB converter and **synchronizers** [5]) or full-fledged (e.g., Oracle Data Integrator [6]). According to [5], DBConvert tools include a number of powerful **transformation** and **synchronization utilities** that perform accurate and convenient way of data transfer from one database to another. DBConvert tools handle the process of transformation and replication data between the most of the leading database engines **MS Access, MS Excel, MS SQL, MS FoxPro, MySQL, PostgreSQL, SQLite, Oracle,** and others. Figure 6.7 shows the GUI of a DBConvert tool that convert and synchronize data between MS Access and MySQL.

Figure 6.7 Example of light-weighted data integration [5]

Exercise 6: Option 1 (6 points)

(Data-level Integration): Although DBSync is an easy choice, you are welcome to choose any other data integration tool to implement the following requirements discussed earlier in the Roster and Course example:

- *Student contact information should be synchronized to avoid the discrepant information due to the fact that updating actions are allowed in both applications.*
- *As soon as a student finished his/her registration in the Course System, the registration data should be delivered to the Roster System. So the student invoice can be created.*
- *In case a student fails to pay the bill for two successive semesters, the student is not allowed for a new registration.*

Instead of using MS SQL and Oracle databases, you can use any databases you like. You can manually insert some data into the databases. Then run the tool to see how you can synchronize these two databases when the defined data changing scenarios occur.

In your report, please clearly describe how data integration is implemented in your exercise, briefly introduce the tool and explain what you have done. Please use screenshots to show your test results.

Submit your zipped source codes via the Option 1 Dropbox (https://cms.psu.edu/section/content/default.asp?WCI=pgDisplay&WCU=CRSCNT&ENTRY_ID=15C6F6F187884E5CA40CCFBB14629D95) .

General Discussion #1 (Data-level integration challenges) [not-graded]

Purpose - In this discussion I would like you to share your understanding of data-level integration.

Tasks - To make it complicated, please image if you have to synchronize more than five (5) different data sources. Please share your understanding by making one original posting or response to one of your mates in carrying on the following tasks:

- Please share your working experience and understandings of the challenges when a data level integration approach is used to synchronize data across five (5) different data sources in a real life situation. If you had no working experience of implementing data integration, please try to finish Exercise 6 to learn some tools and challenging issues (considering this is a real project at work).
- Then share your understanding of the integration challenges when your data sources continue to increase in your enterprise.

To share your experience, please visit the General Discussion Forum (https://cms.psu.edu/section/content/default.asp?WCI=pgDisplay&WCU=CRSCNT&ENTRY_ID=5F3B9C75396B457B845C67A0B94937F8) .

'Data rich and information poor' is what we frequently use to describe the current status of enterprise information systems. To better utilize the available while overwhelming data, data warehousing technology arises in an enterprise's need for synchronized, consolidated, reliable, and unique reporting and analysis of the data at different enterprise levels of aggregation through data-level integration (Figure 6.8). Enterprise information integration becomes more appealing in industry as it essentially focuses on the usefulness of presented information from heterogeneous data sources through goal-oriented extraction, transformation, and aggregation.

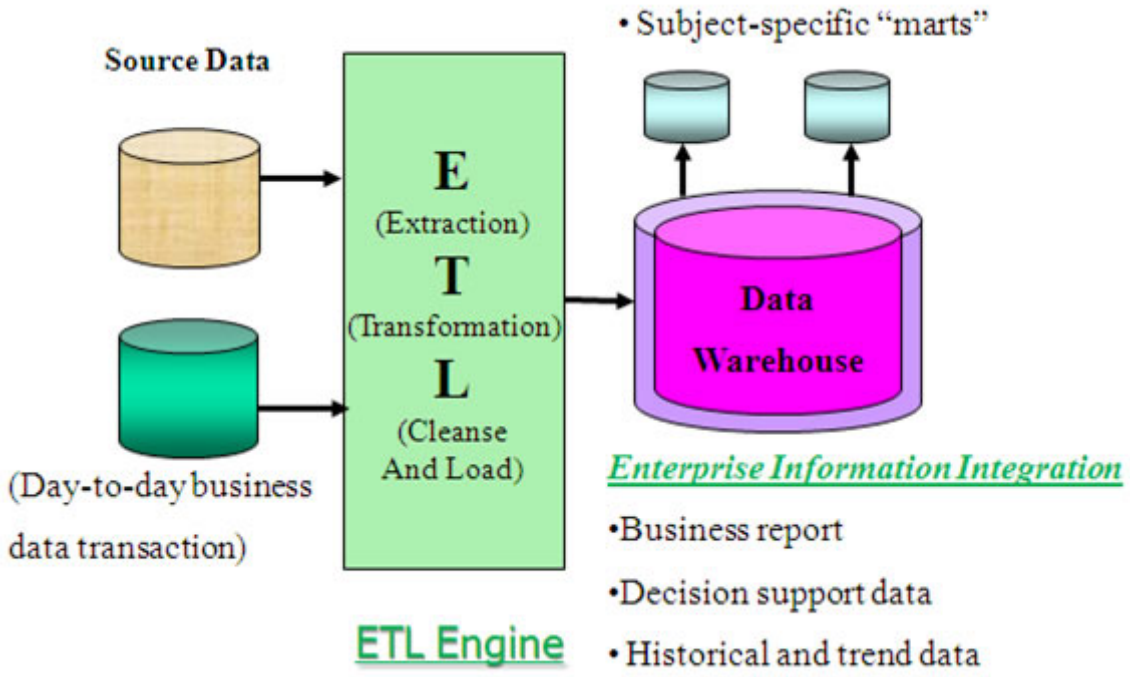


Figure 6.8 Schematic view of data warehousing

General Discussion Assignment #2 (Data warehousing) [not-graded]

Purpose – I would like you to enhance your understanding of data warehousing through participating this discussion forum.

Tasks - To make it complicated, please image if you have to synchronize more than five (5) different data sources. Please share your understanding by making one original posting or response to one of your mates in carrying on the following tasks:

- If you have used data warehousing tools at work, otherwise please read [8, 9] or other data warehouse papers or tutorials from the Internet first, then share your experience or/and understandings of the challenges in applying data warehousing technologies for enterprise information integration projects.
- If examples can be provided, be very specific and concise, and highlight the main challenging issues (e.g., architecture, performance, scalability, and/or security, etc.)..

To share your experience, please visit the General Discussion Forum (https://cms.psu.edu/section/content/default.asp?WCI=pgDisplay&WCU=CRSCNT&ENTRY_ID=5F3B9C75396B457B845C67A0B94937F8) .

Method-level and API-level Integration

Method level integration focuses on synchronizing and sharing business rules and logics through direct interaction among objects within applications (Figure 6.9). This approach minimizes the impact on data sources as the integration occurs at the level above the data sources. As a result, an enterprise can realize the needed coordination among applications at certain degree by sharing functions and synchronizing data across the applications. Method level integration, however, is viable only when the integrated applications allow integration practitioners to access each other's business objects and implementation.

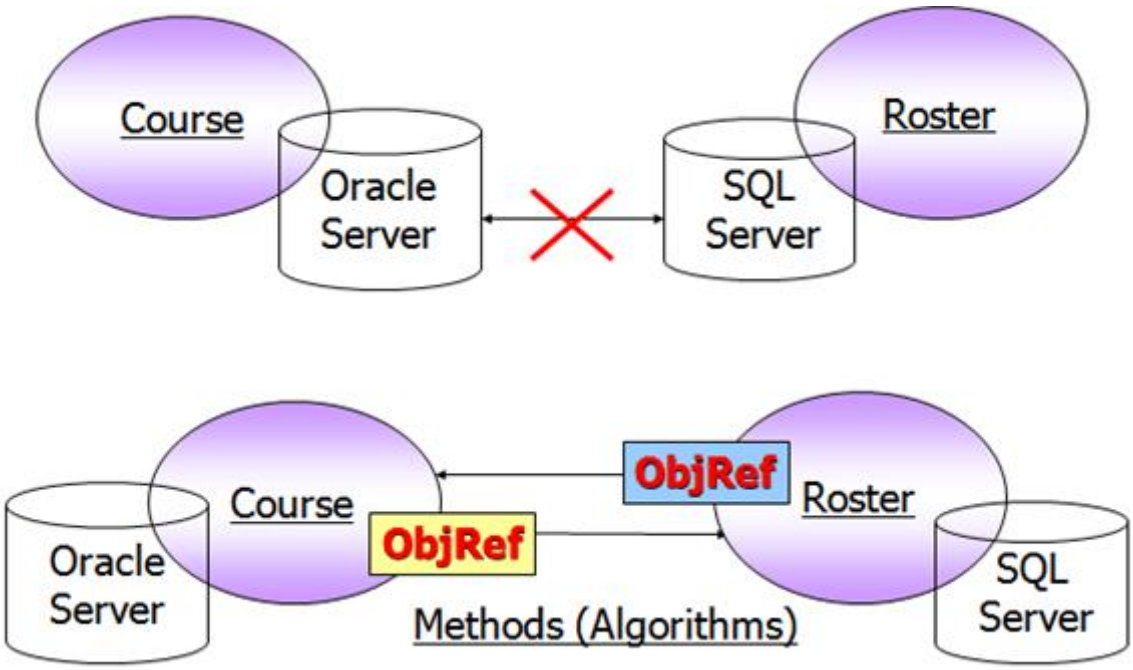
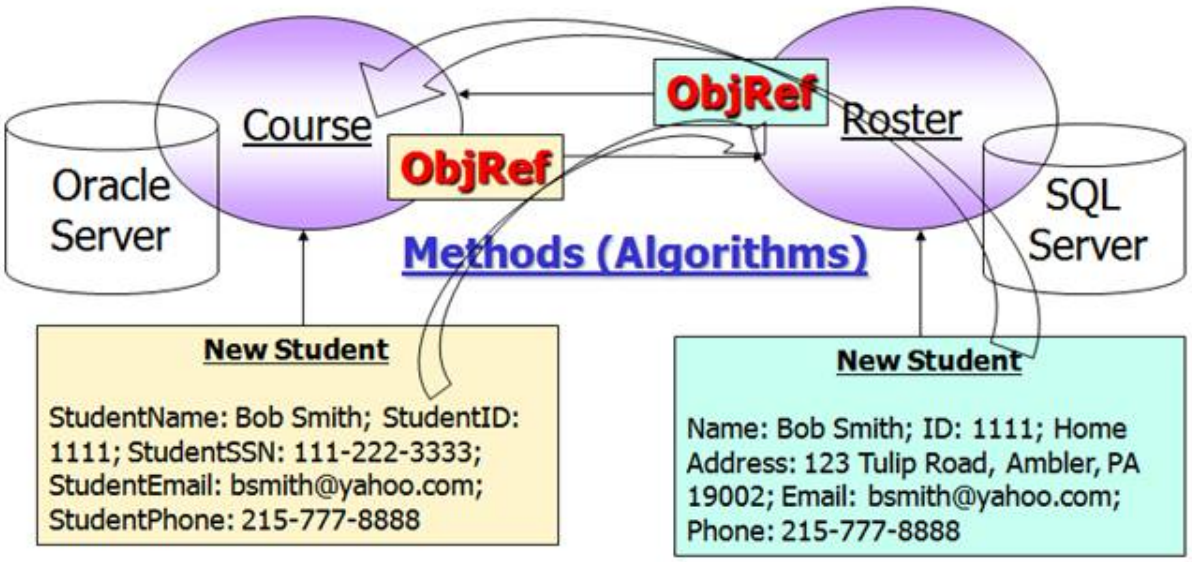


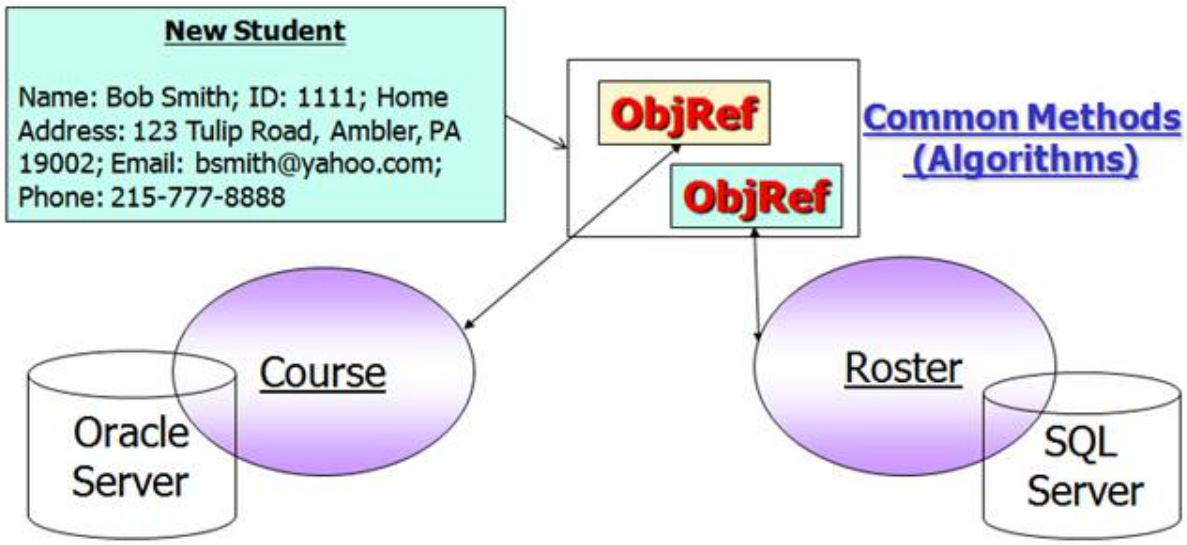
Figure 6.9 Example of method-level integration

As we discussed in previous lessons, the terms method, procedure, function, and operation are exchangeable in this . Depends on the technologies used, these terms essentially indicate the same logical and business support: a given predefined logic function enabled/provided by applications (Figure 6.10(a)). The advantages of method-level integration are that it often provides quite fine-grained access to business functions; and when appropriate middleware is used, a wide range of applications can access the operations and invoking the methods is simple and straightforward. The disadvantage is that the fine-grained nature of the methods can make it difficult to support transactions, run complex business logic, or support common technical services, such as security access control across applications [2]. Most likely, significant scaffolding in each method becomes necessary for a large-scale integration project if the method-level integration is adopted.

Under some circumstance, this method-level integration approach can work in an aggregated manner, the same as what data warehousing does but at a higher level. In other words, common business logic is put together and shared as a collection of common methods or operations that different applications can invoke (Figure 6.10(b)). For example, common operations to update student data, to validate a payment transaction, or to deposit money into a bank account may be made available in a common repository or reusable framework that all applications can access.



a. Each application can invoke a method of other applications



b. Each application can invoke a method of other applications

Figure 6.10 Examples of two different implementations at the method-level integration

Application level integration is also called application programming interface (API) level integration. An API is used when applications don't allow integration practitioners to directly access their business objects and data sources (Figure 6.11). An application is **packaged**, and instead of allowing users to have direct access to its internal algorithms and data, it provides a set of APIs that are essentially externally invoking mechanisms enabled by application providers. Using the provided APIs an applications communications with other applications to realize any needed functionality/data sharing encapsulated by legacy systems (Figure 6.12).

API level integration has many advantages, such as preserving the application's data integrity, no changes to the application itself during integration, and potential ease of use of new integration architecture and technologies. The main disadvantages are that many proprietary applications developed within enterprises rarely have a defined API, and the provided APIs could be very limited in scope and might not offer the behavior that an integration needs. Traditional APIs are often function-oriented (i.e., procedural) in nature and not object-oriented [2].

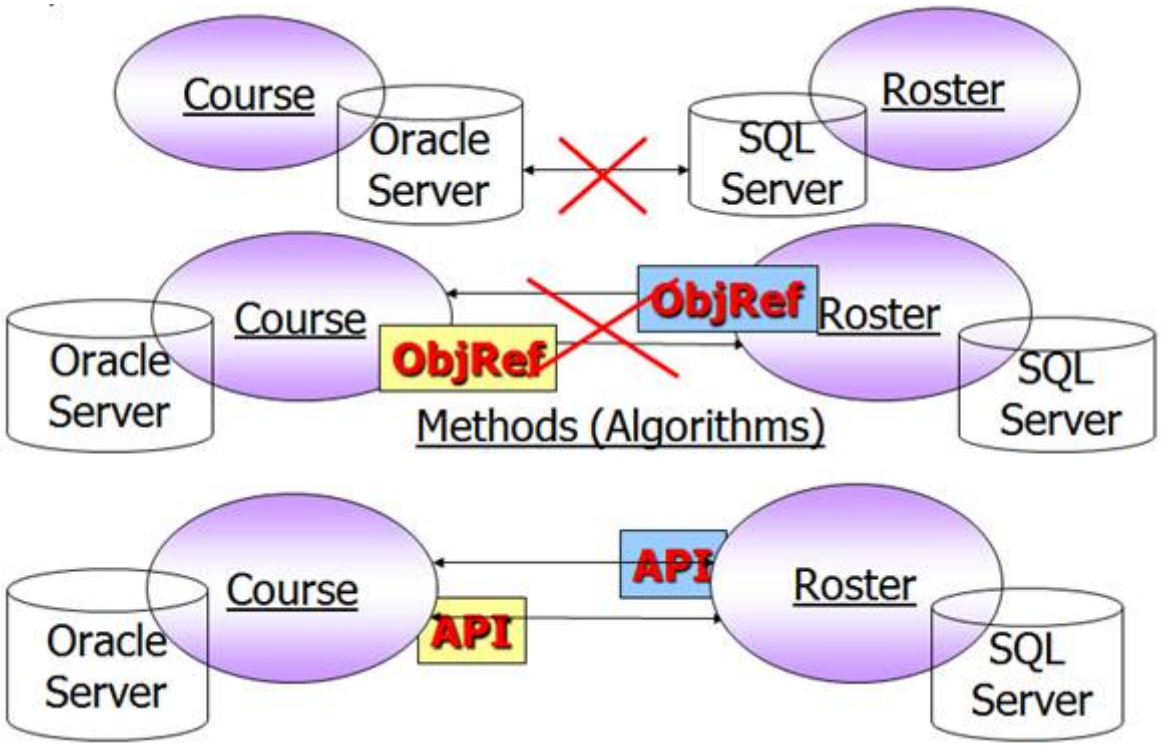


Figure 6.11 Schematic view of data warehousing

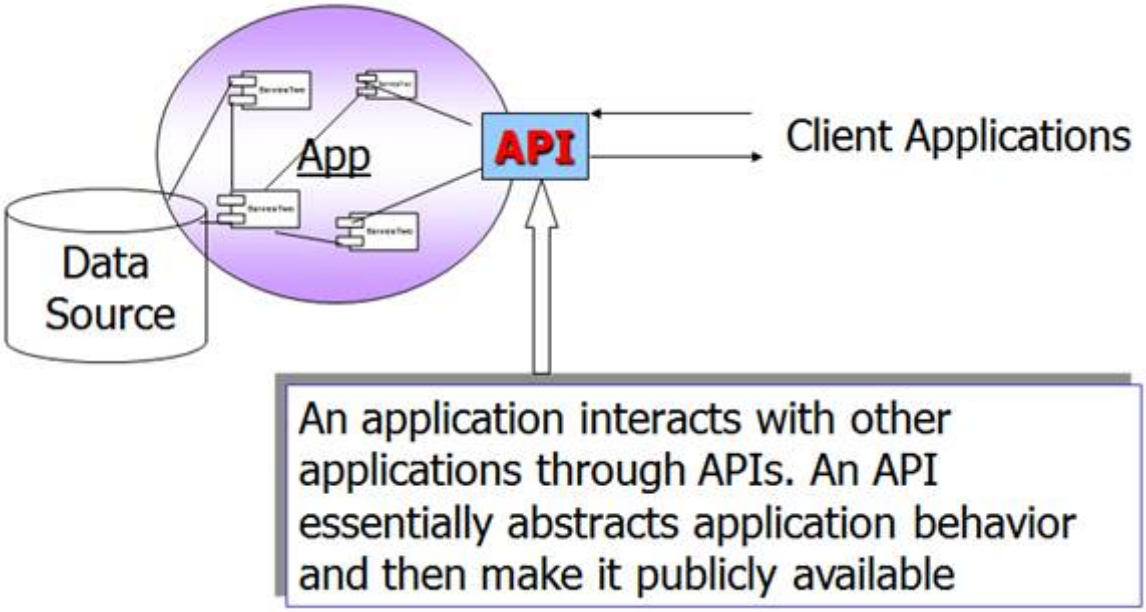


Figure 6.12 Schematic view of an API

EDI (Electronic Data Interchange) was originally developed in the mid-1970s and 1980s in order to reduce the procurement costs by minimizing delay and errors from suppliers. It has been in widespread use as a technique for conducting data level integration between heterogeneous systems for both intra-enterprise and inter-enterprise applications. EDI is essentially a group of defined standard communication protocols for applications, for instance, ANSI X12 from the American National Standard Institute and EDIFACT (i.e., Electronic Data Interchange for Administration, Commerce and Transport) from the United Nations (UN Economic Commission for Europe).

EDI messages contain distinct fields for each of the important pieces of information in a commercial transaction, such as transaction date, product purchased amount, sender and recipient's information. Since EDI was initially designed as a standard communication protocol for sharing business documents over a private communication network, it was typically customized for a specific industrial sector. As a result, an EDI used in one business domain will most likely not be able to communicate with one in a different business domain. There are a variety of EDIs available today; popular ones include X12, EDIFACT, SWIFT, HL7, HIPPA, RosettaNet, and ebXML [2]. Although the Internet technologies become pervasive across all business domains, EDI is still the data format used by the vast majority of electronic commerce transactions around the world. Of course, integration practitioners have incorporated Internet technologies in EDI-based applications aimed at further improving their interoperability.

Leveraging user interfaces to access application data and business logics is another integration method when dealing with existing systems (or a new application). You can consolidate different application user interfaces while having contents integrated and shared. For instance, portals and content management systems are gaining popularity in dealing with legacy systems as their user interfaces can be dynamically integrated at runtime including not only contents but also window panes, menus, toolbars, properties, and preferences.

Process-level Integration (7 of 8)

Process-level Integration

As we discussed in Lesson 4, whether data/functionality sharing at a micro- or macro-level (i.e., operational- or strategic-level) enabled by IT systems (i.e., integrated software systems) in an enterprise, enterprise integration essentially aims to provide better data/information/knowledge services to assist or serve employees and customers to meet their changing **needs** (i.e., **services** terms in a contemporary fashion).

According to the Workflow Management Coalition (WFMC) [9], a workflow is "the automation of a business process, in whole or in part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules". The participants can be human, machines, or software systems. The automation of processes is achieved with the help of a software system called a "workflow engine". A process controlled by a workflow engine is viewed as a collection of tasks executed by various resources within a value system comprising one or more integrating organization units. The reach of sub processes within a WF model can extend to inter-organizational processes, and the model of WF can contain not only automated processes but also human resources (Figure 6.13).

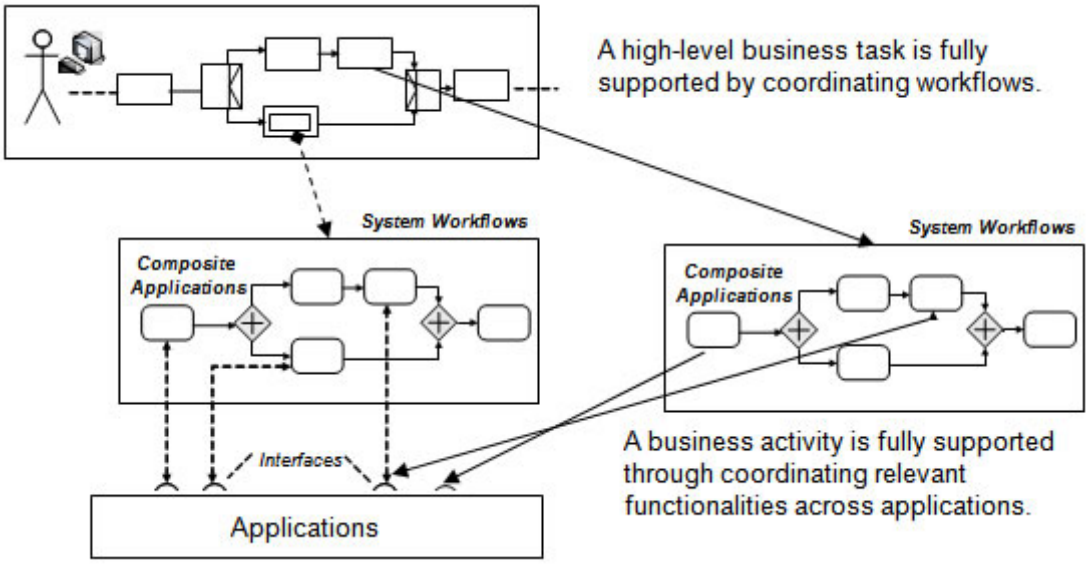


Figure 6.13 Illustration of Workflows in support of Business Services

To refresh your memory, a generic service-oriented IT computing architecture for the development of a state-of-the-art enterprise network is illustrated in Figure 6.13. The top two layers represent services operations from the business process perspective while the bottom three layers shows the value-adding services processes from the computing perspective. Apparently, how to optimally align enterprise-level business strategies with value-adding operations/activities is the key to the success of the deployment of an agile enterprise service-oriented IT system. Web services are standard runtime technologies over the Internet, providing best ever mechanisms for addressing heterogeneous computing issues.

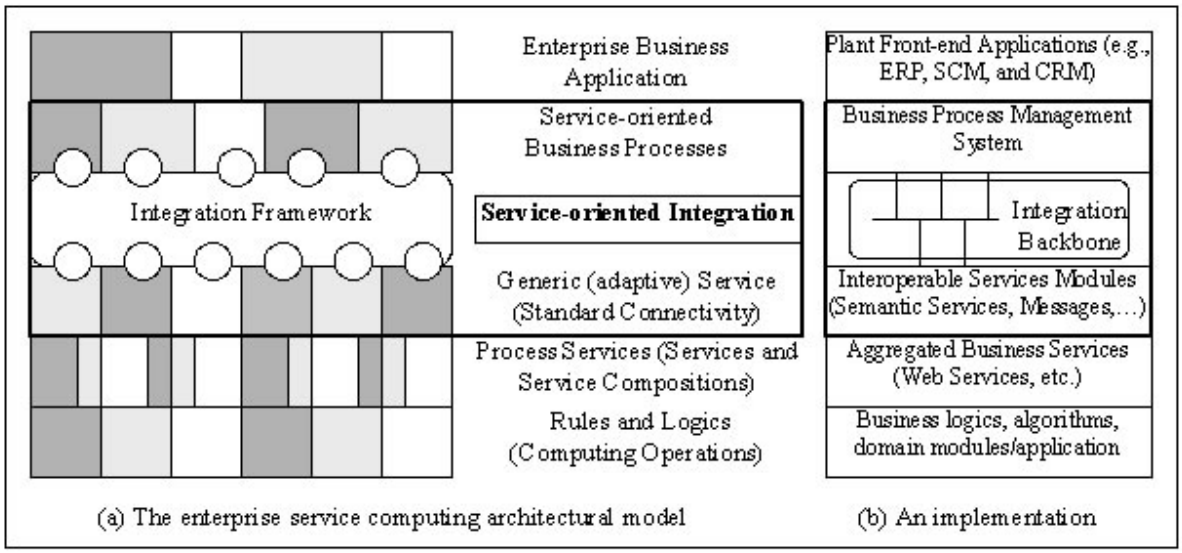


Figure 6.14 Service-Oriented Component Network Architectural Model (Adapted from [10])

Business process management (BPM) [11] emerges as a promising technology for integrating existing assets and future deployments through transforming the behavior of disparate and heterogeneous systems into standard and interoperable business process. BPM essentially provides a framework of applications to achieve effectively tracking and orchestrating business

process through enterprise integration. Beyond the workflow concept, BPM not only allows automation of processes, but also enables process self-management. Therefore, BPM systems position as the platform for the next generation of "process aware" integrating applications, providing the agility needs for today's uncertain business environment (see Lesson 7).

References

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

[2] Linthicum, D. 2000. Enterprise Application Integration. Addison-Wesley, New Jersey, USA

[3] Wikipedia:

- Data Integration: http://en.wikipedia.org/wiki/Data_integration,
- Data Transformation Services: http://en.wikipedia.org/wiki/Data_Transformation_Services,
- SQL Server Integration Services: http://en.wikipedia.org/wiki/SQL_Server_Integration_Services,
- Data Warehouse: http://en.wikipedia.org/wiki/Data_warehouse,
- Enterprise Information Integration: http://en.wikipedia.org/wiki/Enterprise_Information_Integration,
- EDI: http://en.wikipedia.org/wiki/Electronic_Data_Interchange.

[4] Data Integration tools:

- MS SQL DTS: http://www.sqldts.com,
- SQL Server Integration Services: http://msdn.microsoft.com/en-us/library/ms141026.aspx,
- MSDN Data Integration: http://msdn.microsoft.com/en-us/library/ms978572.aspx,
- DMSoft Technologies: http://dbconvert.com/,
- Datasync: http://datasyncorp.com/,
- Pentaho Data Integration: http://www.pentaho.com/products/data_integration/,
- iWay Data Hub: http://www.iwaysoftware.com/products/eii.html,
- IBM Information Integration: http://www-01.ibm.com/software/data/integration/,
- Informatica: http://www.informatica.com/Pages/index.aspx,
- Oracle Data Integrator: http://www.oracle.com/technetwork/middleware/data-integrator,
- SAS® Data Integration: http://www.sas.com/technologies/dw/,
- DataFlux
- SAP Data Integration : http://www.winshuttle.com/SAP-data-integration.html.

[5] DMSoft Technologies: http://dbconvert.com/.

[6] Oracle Data Integrator: http://www.oracle.com/technetwork/middleware/data-integrator.

[7] MS Data Warehousing: http://www.microsoft.com/en-us/server-cloud/solutions/modern-data-warehouse/.

[8] Data Warehousing With Oracle: http://www.oracular.com/white_paper_pdfs/DataWarehousingwithOracle.pdf.

[9] Qiu, R. Information Technology as a Service. Enterprise Service Computing: From Concept to Deployment. (Chapter 1) Ed. By R. Qiu. IGI Publishing, Hershey, PA.

[10] WFMC: www.wfmc.org.

[11] BPM: www.bpmi.org.
