Enterprise Computer Applications & Integration

Lavon Wasson

First Edition, 2012

ISBN 978-81-323-3048-6

Published by:
Research World
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1

Enterprise Application Integration

Enterprise Application Integration (EAI) is defined as the use of software and computer systems architectural principles to integrate a set of enterprise computer applications.

Overview

Enterprise Application Integration (EAI) is an integration framework composed of a collection of technologies and services which form a middleware to enable integration of systems and applications across the enterprise.

Supply chain management applications (for managing inventory and shipping), customer relationship management applications (for managing current and potential customers), business intelligence applications (for finding patterns from existing data from operations), and other types of applications (for managing data such as human resources data, health care, internal communications, etc.) typically cannot communicate with one another in order to share data or business rules. For this reason, such applications are sometimes referred to as islands of automation or information silos. This lack of communication leads to inefficiencies, wherein identical data are stored in multiple locations, or straightforward processes are unable to be automated.

Enterprise application integration (EAI) is the process of linking such applications within a single organization together in order to simplify and automate business processes to the greatest extent possible, while at the same time avoiding having to make sweeping changes to the existing applications or data structures. In the words of the Gartner Group, EAI is the "unrestricted sharing of data and business processes among any connected application or data sources in the enterprise."

One large challenge of EAI is that the various systems that need to be linked together often reside on different operating systems, use different database solutions and different

computer languages, and in some cases are legacy systems that are no longer supported by the vendor who originally created them. In some cases, such systems are dubbed "stovepipe systems" because they consist of components that have been jammed together in a way that makes it very hard to modify them in any way.

Improving connectivity

If integration is applied without following a structured EAI approach, point-to-point connections grow across an organization. Dependencies are added on an impromptu basis, resulting in a tangled mess that is difficult to maintain. This is commonly referred to as spaghetti, an allusion to the programming equivalent of spaghetti code. For example:

The number of connections needed to have fully meshed point-to-point connections, with *n* points, is given by $\frac{n(n-1)}{2}$. Thus, for ten applications to be fully integrated point-to-point, $\frac{10 \times 9}{2}$, or 45 point-to-point connections are needed.

However, EAI is not just about sharing data between applications; it focuses on sharing both business data and business process. Middleware Analysts attending to EAI involves looking at the system of systems, which involves large scale inter-disciplinary problems with multiple, heterogeneous, distributed systems that are embedded in networks at multiple levels.

Purposes

EAI can be used for different purposes:

- Data integration: Ensures that information in multiple systems is kept consistent. This is also known as Enterprise Information Integration (EII).
- Vendor independence: Extracts business policies or rules from applications and implements them in the EAI system, so that even if one of the business applications is replaced with a different vendor's application, the business rules do not have to be re-implemented.
- Common facade: An EAI system can front-end a cluster of applications, providing a single consistent access interface to these applications and shielding users from having to learn to use different software packages.

Patterns

Integration patterns

There are two patterns that EAI systems implement:

Mediation (intra-communication)
: Here, the EAI system acts as the go-between or broker between (interface or communicating) multiple applications. Whenever an interesting event occurs in an application (e. g., new information created, new transaction completed, etc.) an integration module in the EAI system is notified. The module then propagates the changes to other relevant applications.

Federation (inter-communication)
: In this case, the EAI system acts as the overarching facade across multiple applications. All event calls from the 'outside world' to any of the applications are front-ended by the EAI system. The EAI system is configured to expose only the relevant information and interfaces of the underlying applications to the outside world, and performs all interactions with the underlying applications on behalf of the requester.

Both patterns are often used concurrently. The same EAI system could be keeping multiple applications in sync (mediation), while servicing requests from external users against these applications (federation).

Access patterns

EAI supports both asynchronous and synchronous access patterns, the former being typical in the mediation case and the latter in the federation case.

Lifetime patterns

An integration operation could be short-lived (e.g. keeping data in sync across two applications could be completed within a second) or long-lived (e.g. one of the steps could involve the EAI system interacting with a human work flow application for approval of a loan that takes hours or days to complete).

Topologies

There are two major topologies: hub-and-spoke, and bus. Each has its own advantages and disadvantages. In the hub-and-spoke model, the EAI system is at the center (the hub), and interacts with the applications via the spokes. In the bus model, the EAI system is the bus (or is implemented as a resident module in an already existing message bus or message-oriented middleware).

Technologies

Multiple technologies are used in implementing each of the components of the EAI system:

Bus/hub
: This is usually implemented by enhancing standard middleware products (application server, message bus) or implemented as a stand-alone program (i. e., does not use any middleware), acting as its own middleware.

Application connectivity
: The bus/hub connects to applications through a set of **adapters** (also referred to as **connectors**). These are programs that know how to interact with an underlying business application. The adapter performs two-way communication, performing requests from the hub against the application, and notifying the hub when an event of interest occurs in the application (a new record inserted, a transaction completed, etc.). Adapters can be specific to an application (e. g., built against the application vendor's client libraries) or specific to a class of applications (e. g., can interact with any application through a standard communication protocol, such as SOAP, SMTP or Action Message Format (AMF)). The adapter could reside in the same process space as the bus/hub or execute in a remote location and interact with the hub/bus through industry standard protocols such as message queues, web services, or even use a proprietary protocol. In the Java world, standards such as JCA allow adapters to be created in a vendor-neutral manner.

Data format and transformation
: To avoid every adapter having to convert data to/from every other applications' formats, EAI systems usually stipulate an application-independent (or common) data format. The EAI system usually provides a data transformation service as well to help convert between application-specific and common formats. This is done in two steps: the adapter converts information from the application's format to the bus's common format. Then, semantic transformations are applied on this (converting zip codes to city names, splitting/merging objects from one application into objects in the other applications, and so on).

Integration modules
: An EAI system could be participating in multiple concurrent integration operations at any given time, each type of integration being processed by a different integration module. Integration modules subscribe to events of specific types and process notifications that they receive when these events occur. These modules could be implemented in different ways: on Java-based EAI systems, these could be web applications or EJBs or even POJOs that conform to the EAI system's specifications.

Support for transactions
: When used for process integration, the EAI system also provides transactional consistency across applications by executing all integration operations across all applications in a single overarching distributed transaction (using two-phase commit protocols or compensating transactions).

Communication architectures

Currently, there are many variations of thought on what constitutes the best infrastructure, component model, and standards structure for Enterprise Application Integration. There seems to be consensus that four components are essential for a modern enterprise application integration architecture:

1. A centralized broker that handles security, access, and communication. This can be accomplished through integration servers (like the School Interoperability Framework (SIF) Zone Integration Servers) or through similar software like the enterprise service bus (ESB) model that acts as a SOAP-oriented services manager.
2. An independent data model based on a standard data structure, also known as a canonical data model. It appears that XML and the use of XML style sheets has become the *de facto* and in some cases *de jure* standard for this uniform business language.
3. A connector, or agent model where each vendor, application, or interface can build a single component that can speak natively to that application and communicate with the centralized broker.
4. A system model that defines the APIs, data flow and rules of engagement to the system such that components can be built to interface with it in a standardized way.

Although other approaches like connecting at the database or user-interface level have been explored, they have not been found to scale or be able to adjust. Individual applications can publish messages to the centralized broker and subscribe to receive certain messages from that broker. Each application only requires one connection to the broker. This central control approach can be extremely scalable and highly evolvable.

Enterprise Application Integration is related to middleware technologies such as message-oriented middleware (MOM), and data representation technologies such as XML. Other EAI technologies involve using web services as part of service-oriented architecture as a means of integration. Enterprise Application Integration tends to be data centric. In the near future, it will come to include content integration and business processes.

Implementation pitfalls

In 2003 it was reported that 70% of all EAI projects fail. Most of these failures are not due to the software itself or technical difficulties, but due to management issues. Integration Consortium European Chairman Steve Craggs has outlined the seven main pitfalls undertaken by companies using EAI systems and explains solutions to these problems.

1. Constant change: The very nature of EAI is dynamic and requires dynamic project managers to manage their implementation.

2. Shortage of EAI experts: EAI requires knowledge of many issues and technical aspects.
3. Competing standards: Within the EAI field, the paradox is that EAI standards themselves are not universal.
4. EAI is a tool paradigm: EAI is not a tool, but rather a system and should be implemented as such.
5. Building interfaces is an art: Engineering the solution is not sufficient. Solutions need to be negotiated with user departments to reach a common consensus on the final outcome. A lack of consensus on interface designs leads to excessive effort to map between various systems data requirements.
6. Loss of detail: Information that seemed unimportant at an earlier stage may become crucial later.
7. Accountability: Since so many departments have many conflicting requirements, there should be clear accountability for the system's final structure.

Other potential problems may arise in these areas:

- Emerging Requirements: EAI implementations should be extensible and modular to allow for future changes.
- Protectionism: The applications whose data is being integrated often belong to different departments that have technical, cultural, and political reasons for not wanting to share their data with other departments

Advantages and disadvantages

Advantages

- Real time information access among systems
- Streamlines business processes and helps raise organizational efficiency
- Maintains information integrity across multiple systems
- Ease of development and maintenance

Disadvantages

- High initial development costs, especially for small and mid-sized businesses (SMBs)
- Require a fair amount of up front business design, which many managers are not able to envision or not willing to invest in.

Most EAI projects usually start off as point-to-point efforts, quickly becoming unmanageable as the number of applications increase.

Future

EAI technologies are still being developed and there still is no consensus on the ideal approach or the correct group of technologies a company should use. There are many

ongoing projects that provide support to design EAI solutions. They may range from proprietary or open projects like Microsoft BizTalk Server or Apache Camel to academic projects like Guaraná DSL. The future is to provide technologies that allow the design of EAI solutions at a high level of abstraction and use MDA to automatically transform the design into an executable solution. A common pitfall is to use other proprietary technologies that claim to be open and extensible but create vendor lock-in.

Chapter 2

Enterprise Content Management

Enterprise Content Management (**ECM**) is a formalized means of organizing and storing an organization's documents, and other content, that relate to the organization's processes. The term encompasses strategies, methods, and tools used throughout the lifecycle of the content.

Definition

The Association for Information and Image Management (AIIM) International, the worldwide association for enterprise content management, defined the term *Enterprise Content Management* in 2000. AIIM has refined the abbreviation *ECM* several times to reflect the expanding scope and importance of information management:

Late 2005

> *Enterprise content management is the technologies used to Capture, Manage, Store, Preserve, and Deliver content and documents related to organizational processes.*

Early 2006

> *Enterprise content management is the technologies used to Capture, Manage, Store, Preserve, and Deliver content and documents related to organizational processes.*
>
> *ECM tools and strategies allow the management of an organization's unstructured information, wherever that information exists.*

Early 2008

> *Enterprise Content Management (ECM) is the strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to organizational processes. ECM tools and strategies allow the management of an organization's unstructured information, wherever that information exists.*

Early 2010

> *Enterprise Content Management (ECM) is the strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to*

organizational processes. ECM covers the management of information within the entire scope of an enterprise whether that information is in the form of a paper document, an electronic file, a database print stream, or even an email

The latest definition encompasses areas that have traditionally been addressed by records management and document management systems. It also includes the conversion of data between various digital and traditional forms, including paper and microfilm.

ECM is an umbrella term covering document management, web content management, search , collaboration, records management, digital asset management (DAM), work-flow management, capture and scanning. ECM is primarily aimed at managing the life-cycle of information from initial publication or creation all the way through archival and eventually disposal. ECM applications are delivered in three ways: on-premise software (installed on the organization's own network), Software as a Service (SaaS) (web access to information that is stored on the software manufacturer's system), or a hybrid solution composed of both on-premise and SaaS components.

ECM aims to make the management of corporate information easier through simplifying storage, security, version control, process routing, and retention. The benefits to an organization include improved efficiency, better control, and reduced costs. For example, many banks have converted to storing copies of old checks within ECM systems versus the older method of keeping physical checks in massive paper warehouses. Under the old system a customer request for a copy of a check might take weeks, as the bank employees had to contact the warehouse to have someone locate the right box, file and check, pull the check, make a copy and then mail it to the bank who would eventually mail it to the customer. With an ECM system in place, the bank employee simply searches the system for the customer's account number and the number of the requested check. When the image of the check appears on screen, they are able to immediately mail it to the customer—usually while the customer is still on the phone.

History

Enterprise Content Management, as a form of content management, combines the capture, search and networking of documents with digital archiving, document management and workflow. It specifically includes the special challenges involved in using and preserving a company's internal, often unstructured information, in all of its forms. Therefore, most ECM solutions focus on *Business-to-Employee (B2E)* systems.

As ECM solutions have evolved, new components have emerged. For example, as content is checked in and out, each use generates new metadata about the content, to some extent automatically; information about how and when the content was used can allow the system to gradually acquire new filtering, routing and search pathways, corporate taxonomies and semantic networks, and retention-rule decisions. Email and instant messaging are increasingly employed in decision-making processes; ECM can provide access to data about these communications, which can be used in business decisions.

Solutions can provide intranet services to employees (B2E), and can also include enterprise portals for *Business-to-Business (B2B)*, *Business-to-Government (B2G)*, *Government-to-Business (G2B)*, or other business relationships. This category includes most former document-management groupware and workflow solutions that have not yet fully converted their architecture to ECM, but provide a web interface. Digital asset management is a form of ECM concerned with content stored using digital technology.

The technologies that comprise ECM today are the descendants of late 1980s and early 1990s electronic document management systems (EDMS). The original EDMS products were stand-alone products, providing functionality in one of four areas: imaging, workflow, document management, or COLD/ERM.

The typical early EDMS adopter deployed a small-scale imaging and workflow system, possibly to just a single department, in order to improve a paper-intensive process and migrate towards the mythical paperless office. The first stand-alone EDMS technologies were designed to save time and/or improve information access by reducing paper handling and paper storage, thereby reducing document loss and providing faster access to information. EDMS could provide online access to information formerly available only on paper, microfilm, or microfiche. By improving control over documents and document-oriented processes, EDMS streamlined time-consuming business practices. The audit trail generated by EDMS enhanced document security, and provided metrics to help measure productivity and identify efficiency.

Through the late 1990s, the EDMS industry continued to grow steadily. The technologies appealed to organizations that needed targeted, tactical solutions to address clearly defined problems.

As time passed, and more organizations achieved "pockets" of productivity with these technologies, it became clear that the various EDMS product categories were complementary. Organizations increasingly wanted to leverage multiple EDMS products. Consider, for example, a customer service department—where imaging, document management, and workflow could be combined to allow agents to better resolve customer inquiries. Likewise, an accounting department might access supplier invoices from a COLD/ERM system, purchase orders from an imaging system, and contracts from a document management system as part of an approval workflow. As organizations established an Internet presence, they wanted to present information via the web, which required managing web content. Organizations that had automated individual departments now began to envision wider benefits from broader deployment. Many documents cross multiple departments and affect multiple processes.

The movement toward integrated EDMS solutions merely reflected a common trend in the software industry: the ongoing integration of point solutions into more comprehensive solutions. For example, until the early 1990s, word processing, spreadsheet, and presentation software products were standalone products. Thereafter, the market shifted toward integration.

Early leaders already offered multiple stand-alone EDMS technologies. The first phase was to offer multiple systems as a single, packaged "suite", with little or no functional integration. Throughout the 1990s, integration increased. Beginning in approximately 2001, the industry began to use the term *enterprise content management* to refer to these integrated solutions.

In 2006, Microsoft (with its SharePoint product family) and Oracle Corporation (with Oracle Content Management) joined established leaders such as EMC Documentum and entered the entry-level "value" market segment of ECM.

Open source ECM products are also available, including Campsite, WebGUI, Alfresco, Sense/Net, eZ Publish, KnowledgeTree, Jumper 2.0, Nuxeo, and Plone.

Government standards, including HIPAA, SAS 70, BS 7799 and ISO/IEC 27001, are factors in developing and deploying ECM. Standards compliance may make outsourcing to certified service providers a viable alternative to an internal ECM deployment.

Today, organizations can deploy a single, flexible ECM system to manage information in all functional departments, including customer service, accounting, human resources, etc.

Today's Adoption Drivers

There are numerous factors driving businesses to adopt an ECM solution, such as the need to increase efficiency, to improve control of information, and to reduce the overall cost of information management for the enterprise. ECM applications streamline access to records through keyword and full-text search allowing employees to get to the information they need directly from their desktops in seconds rather than searching multiple applications or digging through paper records.

These management systems can enhance record control to help businesses to comply with government and industry regulations such as HIPAA, Sarbanes-Oxley, PCI DSS, and the Federal Rules of Civil Procedure. Security functions including user-level, function-level and even record-specific security options protect your most sensitive data. In fact, even information contained on a specific document can be masked using redaction features, so the rest of the document can be shared without compromising individual identity or key data. Every action taken within the system is tracked and reportable for auditing purposes for a wide variety of regulations.

ECM systems can reduce storage, paper and mailing needs, make employees more efficient, and result in better, more informed decisions across the enterprise—all of which reduce the overhead costs of managing information. SaaS ECM services can convert expensive capital outlay for servers and network equipment into a monthly operating expense, while also reducing the IT resources required to manage enterprise records.

Characteristics

Content management includes ECM, Web content management (WCM), content syndication, and media asset management. Enterprise content management is not a closed-system solution or a distinct product category. Therefore, along with Document Related Technologies or Document Lifecycle Management, ECM is just one possible catch-all term for a wide range of technologies and vendors.

The content and structure of today's outward-directed web portal will be the platform for tomorrow's internal information system. In his article in *ComputerWoche*, Ulrich Kampffmeyer distilled ECM to three key ideas that distinguish such solutions from Web content management:

Enterprise content management as integrative middleware
: ECM is used to overcome the restrictions of former vertical applications and island architectures. The user is basically unaware of using an ECM solution. ECM offers the requisite infrastructure for the new world of web-based IT, which is establishing itself as a kind of third platform alongside conventional host and client/server systems. Therefore, EAI (enterprise application integration) and SOA (service-oriented architecture) will play an important role in the implementation and use of ECM.

enterprise content management components as independent services
: ECM is used to manage information without regard to the source or the required use. The functionality is provided as a service that can be used from all kinds of applications. The advantage of a service concept is that for any given functionality only one general service is available, thus avoiding redundant, expensive and difficult to maintain parallel functions. Therefore, standards for interfaces connecting different services will play an important role in the implementation of ECM.

enterprise content management as a uniform repository for all types of information
: ECM is used as a content warehouse (both data warehouse and document warehouse) that combines company information in a repository with a uniform structure. Expensive redundancies and associated problems with information consistency are eliminated. All applications deliver their content to a single repository, which in turn provides needed information to all applications. Therefore, content integration and ILM (Information Lifecycle Management) will play an important role in the implementation and use of ECM.

Enterprise content management is working properly when it is effectively "invisible" to users. ECM technologies are infrastructures that support specialized applications as subordinate services. ECM thus is a collection of infrastructure components that fit into a multi-layer model and include all document related technologies (DRT) for handling, delivering, and managing structured data and unstructured information jointly. As such, enterprise content management is one of the necessary basic components of the overarching e-business application area. ECM also sets out to manage all the information of a WCM and covers archiving needs as a universal repository.

Components

ECM combines components which can also be used as stand-alone systems without being incorporated into an enterprise-wide system.

The five ECM components and technologies were first defined by AIIM as **capture**, **manage**, **store**, **preserve**, and **deliver**.

Capture

Capture involves converting information from paper documents into an electronic format through scanning. Capture is also used to collect electronic files and information into a consistent structure for management. Capture technologies also encompass the creation of metadata (index values) that describe characteristics of a document for easy location through search technology. For example, a medical chart might include the patient ID, patient name, date of visit, and procedure as index values to make it easy for medical personnel to locate the chart.

Earlier document automation systems photographed documents for storage on microfilm or microfiche. Optical scanners now make digital copies of paper documents. Documents already in digital form can be copied, or linked to if they are already available online.

Automatic or semi-automatic capture can use EDI or XML documents, business and ERP applications, or existing specialist application systems as sources.

Recognition technologies

Various recognition technologies can be used to extract information from scanned documents and digital faxes, including:

Optical character recognition (OCR)
: Converts images of typeset text into alphanumeric characters

handprint character recognition (HCR)
: Converts images of handwritten text into alphanumerics. Gives better results for short text in fixed locations than for freeform text.

Intelligent character recognition (ICR)
: Extends OCR and HCR to use comparison, logical connections, and checks against reference lists and existing master data to improve recognition. For example, on a form where a column of numbers is added up, the accuracy of the recognition can be checked by adding the recognized numbers and comparing them to the sum written on the original form.

Optical mark recognition (OMR)
: Reads special markings, such as checkmarks or dots, in predefined fields.

Barcode recognition
: Decodes industry-standard encodings of product and other commercial data.

Image cleanup

Image cleanup features include rotation, straightening, color adjustment, transposition, zoom, aligning, page separation, annotations and despeckling.

Forms processing

In forms capture, there are two groups of technologies, although the information content and character of the documents may be identical. *Forms processing* is the capture of printed forms via scanning; recognition technologies are often used here, since well-designed forms enable largely automatic processing. *Automatic processing* can be used to capture electronic forms, such as those submitted via web pages, as long as the layout, structure, logic, and contents are known to the capture system.

COLD

Computer Output to Laser Disc (COLD) records reports and other documents on optical disks, or any form of digital storage for ongoing management by ECM systems. Another term for this is enterprise report management (ERM). Originally, the technology only worked with laserdiscs; the name was not changed after other technologies supplanted the laserdisc.

Aggregation

Aggregation combines documents from different applications. The goal is to unify data from different sources, forwarding them to storage and processing systems in a uniform structure and format.

Indexing components

Indexing improves searches, and provides alternative ways to organize the information.

Manual indexing assigns index database attributes to content by hand, typically used by the database of a "manage" component for administration and access. Manual indexing may make use of *input designs* to limit the information that can be entered; for example, entry masks may use program logic to restrict inputs based on other information known about the document.

Both automatic and manual attribute indexing can be made easier and better with preset input-design profiles; these can describe document classes that limit the number of possible index values, or automatically assign certain criteria.

Automatic classification programs can extract index, category, and transfer data autonomously. Automatic classification or categorizing, based on the information contained in electronic information objects, can evaluate information based on predefined

criteria or in a self-learning process. This technique can be used with OCR-converted faxes, office files, or output files.

Manage

The **Manage** category includes five traditional application areas:

- Document management (DM)
- Collaboration (or collaborative software, a.k.a. groupware)
- Web content management (including web portals)
- Records management
- Workflow and business process management (BPM)

The Manage category connects the other components, which can be used in combination or separately. Document management, web content management, collaboration, workflow and business process management address the dynamic part of the information's lifecycle. Records management focuses on managing finalized documents in accordance with the organization's document retention policy, which in turn must comply with government mandates and industry practices.

All Manage components incorporate databases and access authorization systems. Manage components are offered individually or integrated as suites. In many cases they already include the "store" components.

Document management

Document management, in this context, refers to document management systems in the narrow sense of controlling documents from creation to archiving. Document management includes functions like:

Check in/check out
: For checking stored information for consistency.

Version management
: To keep track of different versions of the same information with revisions and renditions (same information in a different format).

Search and navigation
: For finding information and its associated contexts.

Organizing documents
: In structures like files, folders, and overviews.

However, document management increasingly overlaps with other "Manage" components, office applications like Microsoft Outlook and Exchange, or Lotus Notes and Domino, as well as "library services" for administering information storage.

Collaboration

Collaboration components in an ECM system help users work with each other to develop and process content. Many of these components were developed from collaborative software, or *groupware*, packages; ECM collaborative systems go much further, and include elements of knowledge management.

ECM systems facilitate collaboration by using information databases and processing methods that are designed to be used simultaneously by multiple users, even when those users are working on the same content item. They make use of knowledge based on skills, resources and background data for joint information processing. Administration components, such as virtual whiteboards for brainstorming, appointment scheduling and project management systems, communications application such as video conferencing, etc., may be included.

Collaborative ECM may also integrate information from other applications, permitting joint information processing.

Web content management

The scope of Enterprise content management integrates web content management systems. WCM as ECM component is used to present information already existing and managed in the ECM repository. However, information presented via Web technologies - on the Internet, an extranet, or on a portal — uses the workflow, access control, versioning, delivery and authorization modules of the ECM instead of own integrated WCM functionality. There are only few examples of successful implementations whereby a shared repository for documents and web content are managed together..

Records management (file and archive management)

Unlike traditional electronic archival systems, records management refers to the pure administration of records, important information, and data that companies are required to archive. Records management is independent of storage media; managed information does not necessarily need to be stored electronically, but can be on traditional physical media as well. Some of the functions of records management are:

- Visualisation of file plans and other structured indexes for the orderly storage of information
- Unambiguous indexing of information, supported by thesauri or controlled wordlists
- Management of record retention schedules and deletion schedules
- Protection of information in accordance with its characteristics, sometimes down to individual content components in documents
- Use of international, industry-specific or company-wide standardized metadata for the unambiguous identification and description of stored information

Workflow/business process management

Workflow and **business process management** differ substantially.

Workflow

There are different types of workflow: *production workflow* uses predefined sequences to guide and control processes, whereas in an *ad-hoc workflow*, the user determines the process sequence on the fly.

Workflow can be implemented as *workflow solutions* with which users interact, or as *workflow engines*, which act as a background service controlling the information and data flow.

Workflow management includes the following functions:

- Visualisation of process and organization structures
- Capture, administration, visualization, and delivery of grouped information with its associated documents or data
- Incorporation of data processing tools (such as specific applications) and documents (such as office products)
- Parallel and sequential processing of procedures including simultaneous saving
- Reminders, deadlines, delegation and other administration functionalities
- Monitoring and documentation of process status, routing, and outcomes
- Tools for designing and displaying process

The objective is to automate processes as much as possible by incorporating all necessary resources.

Business process management

Business process management (BPM) goes a step further than workflow. Although the words are often used interchangeably, BPM aims to completely integrate all of the affected applications within an enterprise, monitoring processes and assembling all required information. Among BPM's functions are:

BPM offers complete workflow functionality, providing process and data monitoring at the server level. Enterprise application integration is used to link different applications. Business intelligence, with rule structures, integrates information warehouses and provides utilities that assist users in their work.

Store

Store components temporarily store information that isn't required, desired, or ready for long-term storage or preservation. Even if the Store component uses media that are suitable for long-term archiving, "Store" is still separate from "Preserve."

The Store components can be divided into three categories: **Repositories** as storage locations, **Library Services** as administration components for repositories, and storage **Technologies**. These infrastructure components are sometimes held at the operating system level (like the file system), and also include security technologies that work in tandem with the "Deliver" components. However, security technologies, including access control, are superordinated components of an ECM solution.

Repositories

Different kinds of ECM repositories can be used in combination. Among the possible kinds are:

File systems
: File systems are used primarily for temporary storage, as input and output caches. ECM's goal is to reduce the data burden on the file system, and make the information generally available through Manage, Store, and Preserve technologies.

Content management systems
: This is the actual storage and repository system for content, which can be a database or a specialized storage system.

Databases
: Databases administer access information, but can also be used for the direct storage of documents, content, or media assets.

Data warehouses
: These are complex storage systems based on databases, which reference or provide information from all kinds of sources. They can also be designed with global functions, such as document or information warehouses.

Library services

Library services are the administrative components of the ECM system that handle access to information. The library service is responsible for taking in and storing information from the Capture and Manage components. It also manages the storage locations in dynamic storage, the actual "Store," and in the long-term Preserve archive. The storage location is determined only by the characteristics and classification of the information. The library service works in concert with the Manage components' database to provide the necessary functions of search and retrieval.

While the database does not "know" the physical location of a stored object, the library service manages online storage (direct access to data and documents), nearline storage (data and documents on a medium that can be accessed quickly, but not immediately, such as data on an optical disc that is present in a storage system's racks but not currently inserted in a drive that can read it), and offline storage (data and documents on a medium that is not quickly available, such as data stored offsite).

If the document management system does not provide the functionality, the library service must have *version management* to control the status of information, and *check-in/check-out*, for controlled information provision.

The library service generates logs of information usage and editing, called an "audit trail."

Storage technologies

A wide variety of technologies can be used to store information, depending on the application and system environment:

Magnetic online media
: Hard drives, typically configured as RAID systems, may be locally attached, part of a storage area network (SAN), or mounted from another server (network-attached storage).

Magnetic tape
: Magnetic tape data storage, in the form of automated storage units called tape libraries, use robotics to provide nearline storage. Standalone tape drives may be used for backup, but not online access.

Digital optical media
: Besides the common Compact Disc and DVD optical media in write-once or rewritable forms, Storage systems may use other specialized optical formats like magneto-optical drives for storage and distribution of data. Optical jukeboxes can be used for nearline storage. Optical media in jukeboxes can be removed, transitioning it from nearline to offline storage.

Cloud computing
: Data can be stored on offsite cloud computing servers, accessed via the Internet.

Preserve

Preserve involves the long-term, safe storage and backup of static, unchanging information. Preservation is typically accomplished by the records management features of an ECM system and many are designed to help companies comply with government and industry regulations.

Eventually, content ceases to change and becomes static. The **Preserve** components of ECM handle the long-term, safe storage and backup of static information, as well as the temporary storage of information that does not need to be archived. *Electronic archiving*, a related concept, has substantially broader functionality than ECM Preserve components. Electronic archiving systems generally consist of a combination of administration software like records management, imaging or document management, library services or information retrieval systems, and storage subsystems.

Other forms of media are also suitable for long-term archiving. If the desire is merely to ensure information is available in the future, microfilm is still viable; unlike many digital

records, microfilm is readable without access to the specialized software that created it. Hybrid systems combine microfilm with electronic media and database-supported access.

Long-term storage systems require the timely planning and regular performance of data migrations, in order to keep information available in the changing technical landscape. As storage technologies fall into disuse, information must be moved to newer forms of storage, so that the stored information remains accessible using contemporary systems. For example, data stored on floppy disks becomes essentially unusable if floppy disk drives are no longer readily available; migrating the data stored on floppy disks to Compact Discs preserves not only the data, but the ability to access it. This ongoing process is called *continuous migration*.

The Preserve components contain special viewers, conversion and migration tools, and long term storage media:

Long term storage media

WORM optical disc
: Write Once Read Many (WORM) rotating digital optical storage media, including the 5.25-inch or 3.5-inch WORM disc in a protective sleeve, as well as CD-R and DVD-R. Recording methods vary for these media, which are held in jukeboxes for online and automated nearline access.

WORM tape
: Magnetic tapes used in special drives, that can be as secure as optical write-once, read-many media if used properly with specially secured tapes.

WORM hard disk drive
: Magnetic disk storage with special software protection against overwriting, erasure, and editing; delivers security similar to optical write-once, read-many media. This category includes content-addressable storage.

Storage networks
: Storage networks, such as network-attached storage and storage area networks, can be used if they meet the requirements of edit-proof auditing with unchangeable storage and protection against manipulation and erasure.

Microform
: Microforms like microfilm, microfiche, and aperture cards can be used to back up information that is no longer in use and does not require machine processing. It is typically used only to double-secure originally electronic information.

Paper
: Paper still has use as a long-term storage medium, since it does not require migration, and can be read without any technical aids. In ECM systems, however, it is used only to double-secure originally electronic information.

Long term preservation strategies

To secure the long term availability of information different strategies are used for electronic archives.

The continuous **migration** of applications, index data, metadata and objects from older systems to new ones generates a lot of work, but secures the accessibility and usability of information. During this process, information that is no longer relevant can be deleted. Conversion technologies are used to update the format of the stored information, where needed.

Emulation of older software allows users to run and access the original data and objects. Special viewer software can identify the format of the preserved objects and can display the objects in the new software environment.

Standards for interfaces, metadata, data structures and object formats are important to secure the availability of information.

Deliver

The **Deliver** components of ECM present information from the Manage, Store, and Preserve components. The AIIM component model for ECM is function-based, and doesn't impose a strict hierarchy; the Deliver components may contain functions used to enter information into other systems (such as transferring information to portable media, or generating formatted output files); or for readying information, such as by converting its format or compressing it, for the "Store" and "Preserve" components. The Deliver category's functionality is also known as "output"; technologies in this category are often termed *output management*.

The Deliver components break down into three groups: **transformation technologies**, **security technologies**, and **distribution**. Transformation and security, as services, are middleware and should be equally available to all ECM components. For output, two functions are of primary importance: *layout and design*, with tools for laying out and formatting output, and *publishing*, with applications for presenting information for distribution and publication.

In short, ECM delivery provides information to users. Secure distribution, collaboration, and version control take the forefront. In some cases, these components are still deployed as stand-alone systems without being incorporated into an enterprise-wide ECM system.

Delivery Methods

On-premise ECM was developed as a traditional software application that companies implemented on their own corporate networks. In this scenario, each individual company manages and maintains both the ECM application, and the network storage devices that store the data. Many on-premise ECM systems are highly customized for individual organizational needs. A note about Capture: since paper document capture requires the use of physical scanning devices, like scanners or multi-function devices, it is typically performed on-premise. However, it can be outsourced to businesses that provide scanning services. Known as Service Bureaus, these companies complete high-volume scanning

and indexing and return the electronic files to organizations via web transfer or on CDs, DVDs, or other external storage devices.

Software as a Service SaaS SaaS ECM means that rather than deploying software on an in-house network, users access the application and their data online. It is also known as cloud computing, hosted, and on demand. As SaaS distribution technologies mature, businesses can count on receiving the same features and customization capabilities they have come to expect from on-premise ECM applications. SaaS delivery allows companies to more quickly begin using ECM, since they do not have to purchase hardware or configure the applications, databases, or servers. In addition, organizations trade the capital costs associated with a hardware and software purchase for a monthly operating expense and storage capabilities that grow automatically to accommodate company growth.

Hybrid In some scenarios, companies find a hybrid composed of both SaaS and on-premise software work best for their situation. For example, hybrid ECM systems are being used to bridge the gap during company moves or to simplify information exchange following an acquisition. Hybrid is also being used when companies want to manage their own ECM on-premise, but also provide easy web access to certain information for business partners or customers using a SaaS model. Hybrid makes the most sense when the two technologies are provided by the same manufacturer, so that features and interfaces are an exact match.

Transformation technologies

Transformations should always be controlled and trackable. This is done by background services which the end user generally does not see. Among the transformation technologies are:

Computer Output to Laser Disc (COLD)
: Unlike its use in the Capture stage, when used for delivery COLD prepares output data for distribution and transfer to the archive. Typical applications are lists and formatted output (for example, individualized customer letters). These technologies also include journals and logs generated by the ECM components. Unlike most imaging media, COLD records are indexed not in a database table, but by absolute positions within the document itself (i.e. page 1, line 82, position 12). As a result, COLD index fields are not available for editing after submission unless they are converted into a standard database.

Personalization
: Functions and output can be customized to a particular user's needs.

XML (Extensible Markup Language)
: A computer language that allows the description of interfaces, structures, metadata, and documents in a standardized, cross-platform manner.

PDF (Portable Document Format)
: A cross-platform print and distribution format. Unlike image formats such as TIFF, PDFs permit content searches, the addition of metadata, and the embedding

of electronic signatures. When generated from electronic data, PDFs are resolution-independent, allowing crisp reproduction at any scale.

XPS (XML Paper Specification)
: An XML specification developed by Microsoft, describing the formats and rules for distributing, archiving, rendering, and processing XPS documents.

Converters and viewers
: Serve to reformat information to generate uniform formats, and also to display and output information from different formats.

Compression
: Used to reduce the storage space needed for pictorial information.

Syndication
: Used for presenting content in different formats, selections, and forms in the context of content management. Syndication allows the same content to be used multiple times in different forms for different purposes.

Security technologies

Security technologies are available to all ECM components. For example, electronic signatures are used not only when documents are sent, but also in data capture via scanning, in order to document the completeness of the capture. Public key infrastructure is a basic technology for electronic signatures. It manages keys and certificates, and checks the authenticity of signatures. Other electronic signatures confirm the identity of the sender and the integrity of the sent data, i.e., that it is complete and unchanged.

In Europe, there are three forms of electronic signatures, of different quality and security: simple, advanced, and qualified. In most European states the qualified electronic signature is legally admissible in legal documents and contracts.

Digital rights management and watermarking are used in content syndication and media asset management, to manage and secure intellectual property rights and copyrights. Digital rights management works with techniques like electronic watermarks that are integrated directly into the file, and seeks to protect usage rights and protect content that is published on the Internet.

Distribution

All of the above technologies serve to provide an ECM's contents to users by various routes, in a controlled and user-oriented manner. These can be active components such as e-mail, data media, memos, and passive publication on websites and portals where users can get the information themselves. Possible output and distribution media include:

- The Internet
  - extranets
  - intranets
  - E-business portals
  - Employee portals

- E-mail
- Fax
- Data transfer by EDI, XML or other formats
- Mobile devices, like mobile phones, PDAs, and others
- Data media like CDs and DVDs
- Digital TV and other multimedia services
- Paper.

The various Deliver components provide information to users in the best way for the given application, while controlling its use as far as possible.

ECM market development

Vendors recognized by the 2009 Gartner ECM Magic Quadrant include Alfresco, Autonomy, Digitech Systems, Day Software, EMC, Ever Team, Fabasoft, HP, Hyland Software, IBM, Laserfiche, Microsoft, Newgen Software Technologies, Objective Corporation, Open Text, Oracle, Perceptive Software, SAP, Saperion, Siav, SpringCM, SunGard, Systemware, Xerox and Xythos Software.

Prior to 2003, the ECM market was dominated by a number of medium-sized independent vendors that fell into two categories: those who had originated as Document Management companies (Advanced Processing & Imaging, Documentum, Laserfiche, FileNet, OpenText, Db technology) and had begun adding on management of other enterprise content, and those who had started as Web Content Management providers (Interwoven, Vignette, Stellent) and had begun trying to branch out into managing other types of content such as business documents and rich media. Larger vendors, such as IBM and Oracle, also had offerings in this space, and the market share remained largely fragmented.

In 2002, Documentum had added collaboration capabilities with its acquisition of eRoom while Interwoven and Vignette countered with their respective acquisitions of iManage and Intraspect. Similarly, Documentum purchased Bulldog for its Digital Asset Management (DAM) capabilities while Interwoven and OpenText countered with acquisitions of MediaBin and Artesia. OpenText also acquired European companies IXOS and Red Dot to shore up its software portfolio. In October 2003, EMC Corporation acquired Documentum. Soon EMC's primary competitors in the database space responded as IBM purchased FileNet and Oracle purchased Stellent in 2006. OpenText also purchased Hummingbird in 2006. Hewlett-Packard (HP) entered the ECM space with its acquisition of Australian company Tower Software in 2008. In March 2009, Autonomy purchased Interwoven, in July 2009 Open Text acquired Vignette, and in February, 2011 OpenText acquired MetaStorm.

In April 2007, independent analyst firm CMS Watch noted that "some of the biggest names in this business are undergoing substantial transformation that will lead to shifting road maps and product sets over the next few years". In addition, 2007 saw the emergence of open-source options for ECM supplied by Nuxeo and Alfresco, along with

a Software-as-a-Service offering from Spring CM. In 2008, Sense/Net released Sense/Net 6.0, an open source ECM and EPS solution.

There are a number of software companies that have sprung up to develop applications to complement ECM with specific functions and features. There are companies that provide third party document and image viewers such as LEAD Technologies, MS Technology, and Accusoft. There are companies that provide workflows such as Office Gemini, SpringCM, and docAssist. There are also several companies that provide plugins for ECMs.

The Web 2.0 wave brought new players to the market with strength in web-based delivery. Koral, Box.net, and EchoSign, AppExchange platform, are representative of this trend. Web 2.0 was also instrumental in bringing Cygnet ECM, an entirely web-based ECM product, to the market. Enterprises are increasingly implementing analytics tools to help present targeted content to users in order to improve productivity, sales and user engagement. This has been referred to by some as "web engagement management".

Gartner estimated that the ECM market was worth approximately $3.3 billion in 2008; this was expected to grow at a compound annual growth rate of 9.5 percent through 2013. After a plethora of industry consolidation, only three or four major companies are left in this space, and the industry as a whole is undergoing a significant transformation as Microsoft commoditizes content-management components.

According to Gartner's 2009 report, 75 percent of Global 2000 companies were highly likely to have a desktop-focused, process-focused content management implementation by 2008, and ECM would continue to absorb other technologies, such as digital asset management and e-mail management. Gartner also predicted that there will be further market consolidation, acquisition, and separation of vendors into platform and solution providers.

Currently, enterprise information management (EIM) is gaining more interest from organizations trying to approach information management (whether structured or unstructured) from an enterprise perspective. EIM combines ECM and business intelligence.

Cloud content management is emerging as a web-based alternative, combining the content focus of ECM with the collaborative elements of social business software.

Chapter 3

Event-Driven SOA

Event-driven SOA is a form of service-oriented architecture (SOA), combining the intelligence and proactiveness of event-driven architecture with the organizational capabilities found in service offerings. Before event-driven SOA, the typical SOA platform orchestrated services centrally, through pre-defined business processes, assuming that what should have already been triggered is defined in a business process. This older approach (sometimes called SOA 1.0) does not account for events that occur across, or outside of, specific business processes. Thus complex events, in which a pattern of activities—both random and scheduled—should trigger a set of services is not accounted for in traditional SOA 1.0 architecture.

SOA 2.0

SOA 2.0 architecture, ("event-driven SOA"), lets business users monitor, analyze, and enrich events to make the connections among disparate events that do not at first appear to be intuitively obvious. This makes these enriched events visible to others, especially business analysts or marketing directors, and also allows the SOA 2.0 system to possibly automate actions to take to address some unique pattern.

SOA 2.0 is the ability to create high-level business events from numerous low-level system events. Events are created by filtering real-time data (from middleware, applications, databases, and Web services, for example) and infusing it with defining detail such as dependencies or causal relationships discovered by correlating other events.

If it's clear, through the enriched events that are produced by an SOA 2.0 environment, that customer shopping cart abandonment rate has escalated in the last few days, a notification to the marketing department could initiate research into what competitors have done to cause customers to buy products elsewhere. Was there a common product in most shopping carts? If so, what are the prices that are being offered by the competition?

In practice, this relationship of streamed events is processed through a causal vector engine, which performs a lookup based on recently viewed events and assigns a causal vector to an event if a relationship is discovered. If A causes B, the causal vector engine checks if B's causal vector rule index contains a reference to A. The engine may handle events for different transactions simultaneously, perhaps in a different order than they occurred.

Unlike sequential or procedural systems (in which clients must poll for change requests), event-driven SOA allows systems and components to respond dynamically, in real time, as events occur. SOA 2.0 complements and extends SOA 1.0 by introducing long-running processing capabilities.

Long running processing capability enables the architecture to collect various asynchronous events over a long period of time and correlate these events into causal relationships. SOA 2.0 event patterns can be designed and implemented to look for event relationships that span days, weeks, or months; and when certain criteria are met, trigger a business process to address the event pattern.

SOA 2.0 event-driven programming is structured around the concept of decoupled relationships between event producers and event consumers: an event consumer doesn't care where or why an event occurs; rather, it's concerned that it will be invoked when the event has occurred. Systems and applications that separate event producers from event consumers typically rely on an event dispatcher, or channel. This channel contains an event queue that acts as an intermediary between event producers and event handlers.

Prototypical SOA 2.0 paradigm

The prototypical SOA 2.0 paradigm contains four essential elements: (1) multiple low-level system events that, separately, do not appear to have any relationship, but through pattern detection by comparing these many events some unusual or less obvious correlation becomes clear; (2) some amount of data enrichment by infusion of related information to each event to more clearly illustrate how the many events are related; (3) a trigger condition which when not met, the business-level event is not created, but when the trigger condition is met, the higher-level business event is created; and (4) some human or automated process that is invoked when the trigger event is reached.

SOA 2.0 Web Services can be composed in two ways: orchestration and choreography. In orchestration, a central process takes control over the involved web services and coordinates the execution of different operations on the web services involved in the operation. The involved SOA 2.0 services do not know (and do not need to know) that they are part of a composition or a higher business process. Only the central coordinator of the orchestration knows this, so the orchestration is centralized with explicit definitions of operations and the order of invocation of SOA 2.0 services.

Choreography on the other hand does not rely on a central coordinator. Rather, each SOA 2.0 service involved in the choreography knows exactly when to execute its operations

(based on defined trigger criteria) and whom to interact with. Choreography is a collaborative effort focused on exchange of messages. All participants of the choreography need to be aware of the business process, operations to execute, messages to exchange, and the timing of message exchanges.

BPEL follows the orchestration paradigm. Choreography is covered by other standards, such as WSCI (Web Services Choreography Interface) and (Web Services Choreography Description Language).

Multiple low-level system events

Causal relationships are inherent in the world around us and are intrinsic to our decision making. The human intelligence processes and gathers these relationships faster than current artificial computational capability can. One of the fundamental obstacles in artificial intelligence is the absence of an automated ability to relate events together as when a human uses human intuition.

Using a Causal Vector Engine, the perception of causality can be enhanced under appropriate spatiotemporal conditions based on structural and temporal rules written into the engine. Perception of complex causal semantics, such as additive, mediated, and bidirectional causalities need to be coded so that the engine can distinguish between events that are related and those that only appear to be related but, in fact, are not.

The engine uses preponderant causal vector rate-of-change propagation to code the relationship among the events and establishes a partial order in which it validates the causality perceived between multiple occurrences. The engine plays and replays the event sequence in different temporal orderd to infer what could be related topological connections and compares these replays to rules preprogrammed by an analyst.

Multiple low-level system events are processed by the Causal Vector Engine and compared against these rules to trigger higher-level Business Events. It does this through a Causality Vector Engine (CVE) console application which displays events in real-time to business analysts. Where streams of events can be observed as they occur, much like a stock ticker, the CVE console app has several windows that list the same events in different contexts, so the business analysts can see what the CVE is doing with the relationships between them.

The Sequential window shows events in date-timestamp order, one or more other windows in various orders as the CVE works through the list of rules and creates implied relationships between the events. Various buttons and controls exist in the console application that enable the business analysts to create relationships between events on-the-fly and define rules that respond to these relationships.

Business analysts can infuse additional defining detail through an SQL query statement attached to a rule or event context. The CVE app works much like a modern day stock trading application that mutual funds managers use to manage risk.

Data enrichment

Most ESB implementations contain a facility called "mediation". For example, mediation flows are part of the WebSphere enterprise service bus intercept. Mule also supports mediation flows. Mediation flows modify messages that are passed between existing services and clients that use those services. A mediation flow mediates or intervenes to provide functions, such as message logging, data transformation, and routing.

As messages pass through the ESB, the ESB enriches the messages destined for a channel that is monitoring for a high-level business event. That is, for each message, the ESB may query a database to obtain additional information about some data entity within the message. For example, based on Customer ID, the ESB mediation flow could get the zip code that the customer resides in. Or, based on IP address of the originating request by the end-user, the ESB mediation flow could lookup what country, state or county that IP address is in.

These examples represent data enrichment, the concept of adding additional value to existing data, based on the intent of the high-level business event to eventually be triggered.

Mediation flows

An ESB mediation flow is one of the component types in a Service Component Architecture (SCA). Like any SCA component, the program accesses a mediation flow through exports that it provides, and the mediation flow forwards messages to other external services via imports. Special kinds of imports and exports for JMS, called JMS bindings, enable developers to specify the binding configuration and write data handling code. The mediation flow consists of a series of mediation primitives that manipulate messages as they flow through the bus.

Once the developers have coded the custom binding for both export and import, they can start to focus on the mediation flow component. In the WebSphere Integration Developer assembly editor, this is done by the JMS Custom Binding Mediation Component where each operation on the flow component's interface is represented by a request and a response.

Service Data Objects (SDO) framework provides a unified framework for data application development. With SDO, developers do not need to be familiar with any specific API in order to access and utilize data. Through SDO, developers simply work with data from multiple data sources, such as relational databases, entity EJB components, XML pages, Web services, the Service Component Architecture, and JavaServer Pages pages.

Mediation flows are entirely independent from the bindings that are used in the imports and exports. In fact, the purpose of having a conversion into an SDO DataObject instance outside of the flow implementation is because mediation flows can then be built without

knowledge of the protocol and format with which messages are sent to and from the mediation module.

Business-level trigger condition

A business-level trigger condition enables the SOA 2.0 architecture to establish real-time customer intelligence, marketing automation and customer loyalty solutions, among other features. Business objects model real-world entities in the architecture such as customers, accounts, loans, and travel itineraries. When the state of one of these objects changes, and a monitoring agent notices this change is significant (when compared to the list of criteria to monitor), an event is created and passed to other monitoring agents.

For example, the detection of an actual business problem or opportunity could lead to increased revenue. If a customer cancels an order, extra manufacturing capacity could reduce the profitability of the production run. A SOA 2.0 event could notify marketing department to create a special sales campaign that would resell the excess capacity, thereby recapturing the original profitable cost-per-unit.

Automatic monitoring of events in operational business process activities as processes execute to see if any immediate action needs to be taken either inside or outside the enterprise. These monitoring agents continually test for specific business conditions and changes in business operations. If necessary, the agents alert people, make recommendations, send messages to other applications or invoke whole business processes when such conditions or changes occur.

Resulting business process

A triggered business process should directly support revenue growth with cost containment, responsiveness to business conditions, or ability to pursue new market opportunities. Resulting business processes could also measure operational progress toward achieving goals, control operational costs by communicating just what is needed to just who needs to know, or report performance status of key processes to key decision makers.

SOA 2.0 Conceptual Examples

Abandoned Shopping Cart

For example, you could construct a CRM event from an "abandoned shopping cart" message (parsing the transaction, customer ID, and time), using other filters to extract the value of goods in the cart and tapping the correlation capabilities of the system to add causal indicators such as whether the commerce site was suffering performance problems. Your CRM event might also include customer value or rank from the customer database.

Engineering Defect

For another example, based on the types of independent service calls received, the SOA 2.0 platform could identify a product defect by detecting the underlying pattern of the separate complaints, then triggering an alert to engineering or production of the possible defect.jou kale ma is sinan

Real-time Electricity Market

Example 3: A potential use of event-driven SOA could be a virtual electricity market where home clothes dryers can bid on the price of the electricity they use in a real-time market pricing system. The real-time market price and control system could turn home electricity customers into active participants in managing the power grid and their monthly utility bills. Customers can set limits on how much they would pay for electricity to run a clothes dryer, for example, and electricity providers willing to transmit power at that price would be alerted over the grid and could sell the electricity to the dryer.

On one side, consumer devices can bid for power based on how much the owner of the device were willing to pay, set ahead of time by the consumer. On the other side, suppliers can enter bids automatically from their electricity generators, based on how much it would cost to start up and run the generators. Further, the electricity suppliers could perform real-time market analysis to determine return-on-investment for optimizing profitability or reducing end-user cost of goods.

Event-driven SOA software could allow homeowners to customize many different types of electricity devices found within their home to a desired level of comfort or economy. The event-driven software could also automatically respond to changing electricity prices, in as little as five-minute intervals. For example, to reduce the home owner's electricity usage in peak periods (when electricity is most expensive), the software could automatically lower the target temperature of the thermostat on the central heating system (in winter) or raise the target temperature of the thermostat on the central cooling system (in summer).

The event-driven SOA software could shut off the heating element of water heaters to the pre-set response limits established by individual homeowners. For example, if the market price of electricity for a given hour exceeded the home owner's limit, the home owner could plan to go without recharging the water's hot temperature for that hour, when prices were high, and opt to delay the hot water temperature recharge to the next hour when electricity market prices might be lower.

All this criteria would be managed through the home owner's personal computer with internet connection, programming the various devices around the home to consume electricity only when the management software approves of the consumption. The savings represented by this technique, and enabled by event-driven SOA, is like improving the gas mileage in your vehicle. It makes your home energy use more efficient

by enabling the consumption of electricity when the real-time prices are lower and inhibiting the consumption of electricity when real-time prices are higher.

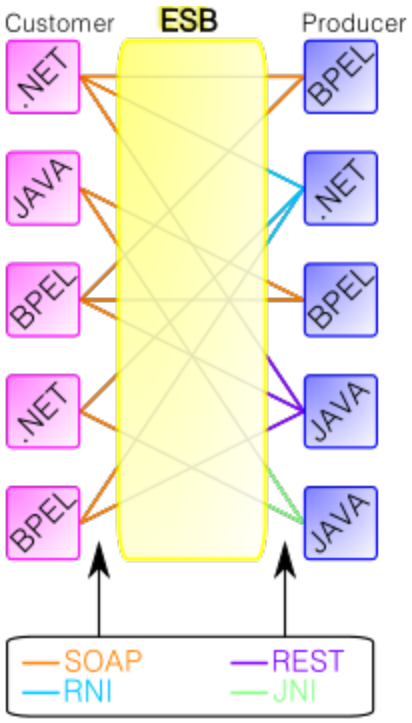
SOA 2.0 Implementations

One mechanism that can be used from most SOA 1.0 Enterprise Service Bus implementations is the publish/subscribe facility. By implementing ESB functionality as Pub/Sub messages, no advanced knowledge of system events is needed to create SOA 2.0 message patterns. After an enterprise has implemented many Publish functions, SOA middleware analysts can set about the task of strategizing which of the available Publish messages could be assembled into a unique pattern to detect an SOA 2.0-enriched trigger.

CVE mechanics are implemented simply, with an expandable view of SQL constructs written in stored procedures. If A causes B, and causality must occur within N number of transactions, then SQL ORDER BY timestamp clause creates a result set that increments a counter of all transactions that occurred within a timeframe, N number of matching B to occurrence A transactions. The creation of additional stored procedures is accomplished through the CVE console application or by using any standard database developer's toolkit.

Chapter 4

Enterprise Service Bus



All customer services communicate in the same way with the ESB: the ESB translates a message to the correct message type and sends the message to the correct producer service.

In computing, an **enterprise service bus** (ESB) is a software architecture construct which provides fundamental services for complex architectures via an event-driven and standards-based messaging engine (the bus). Developers typically implement an ESB using technologies found in a category of middleware infrastructure products, usually based on recognized standards.

An ESB generally provides an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit the value of messaging without writing code. Unlike the more classical enterprise application integration (EAI) approach of a monolithic stack in a hub and spoke architecture, an enterprise service bus builds on base functions broken up into their constituent parts, with distributed deployment where needed, working in harmony as necessary.

Definitions and scope

Commentators disagree over whether to define an Enterprise Service Bus (ESB) as an architectural style, a software product, or a group of software products. While use of an ESB certainly implies adherence to a particular architecture, the term "enterprise service bus" almost always denotes the software infrastructure that enables such an architecture, and in essence, the ESB is considered a platform to realize a service-oriented architecture.

An ESB brings flow-related concepts such as transformation and routing to a Service-Oriented Architecture. An ESB can also provide an abstraction for endpoints. This promotes flexibility in the transport layer and enables loose coupling and easy connection between services.

ESB architecture

Use of the word "bus" stems from the physical bus that carries bits between devices in a computer. The ESB serves an analogous function at a higher level of abstraction. In an enterprise architecture making use of an ESB, an application will communicate via the bus, which acts as a message broker between applications. Such an approach has the primary advantage of reducing the number of point-to-point connections required to allow applications to communicate. This, in turn, makes impact analysis for major software changes simpler and more straightforward. By reducing the number of points-of-contact to a particular application, the process of adapting a system to changes in one of its components becomes easier.

ESB as software

In such a complex architecture, the ESB represents the piece of software that lies between the business applications and enables communication among them. Ideally, the ESB should be able to replace all direct contact with the applications on the bus, so that all communication takes place via the ESB. To achieve this objective, the ESB must encapsulate the functionality offered by its component applications in a meaningful way. This typically occurs through the use of an enterprise message model. The message model defines a standard set of messages that the ESB will both transmit and receive. When the ESB receives a message, it routes the message to the appropriate application. Often, because that application evolved without the same message-model, the ESB will have to transform the message into a format that the application can interpret. A software "adapter" fulfills the task of effecting these transformations (analogously to a physical

adapter). Commentators disagree whether or not these adapters should be considered part of the ESB.

ESBs rely on accurately connecting the enterprise message model and the functionality offered by applications. If the message model does not completely encapsulate the applications' functionality, then other applications that desire that functionality may have to bypass the bus, and invoke the mismatched applications directly. Doing so violates all of the principles outlined above, and negates many of the advantages of using an ESB.

Salient characteristics

The phrase "Enterprise Service Bus" serves as a convenient catch-all term for a set of *capabilities*, which systems can implement in different ways. Commentators debate over whether to regard an ESB as a tangible product or as an architectural style, and on exactly how to implement an ESB (for example: centralised (broker or hub) versus decentralised (smart endpoints). For example, some SOA practitioners claim that SOAP + WS-Addressing **is** the bus. In any case, most observers accept certain core capabilities as functions of an ESB:

Category	Functions
Invocation	support for synchronous and asynchronous transport protocols, service mapping (locating and binding)
Routing	addressability, static/deterministic routing, content-based routing, rules-based routing, policy-based routing
Mediation	adapters, protocol transformation, service mapping
Messaging	message-processing, message transformation and message enhancement
Process choreography	implementation of complex business processes
Service orchestration[2]	coordination of multiple implementation services exposed as a single, aggregate service
Complex event processing	event-interpretation, correlation, pattern-matching
Other quality of service	security (encryption and signing), reliable delivery, transaction management
Management	monitoring, audit, logging, metering, admin console, BAM

[2] *While process choreography supports implementation of complex business processes that require coordination of multiple business services (usually using BPEL), service orchestration enables coordination of multiple implementation services (most suitably exposed as an aggregate service) to serve individual requests.*

In addition, an ESB is expected to exhibit the following characteristics:

- general agnosticism to operating-systems and programming-languages; for example, it should enable interoperability between Java and .NET applications
- general use of XML as the standard communication language
- support for web-services standards
- support for various MEPs (Message Exchange Patterns) (for example: synchronous request/response, asynchronous request/response, send-and-forget, publish/subscribe)
- adapters for supporting integration with legacy systems, possibly based on standards such as JCA
- a standardized security-model to authorize, authenticate and audit use of the ESB
- facilitation of the transformation of data formats and values, including transformation services (often via XSLT or XQuery) between the formats of the sending application and the receiving application
- validation against schemas for sending and receiving messages
- the ability to apply business rules uniformly
- enriching messages from other sources
- the splitting and combining of multiple messages and the handling of exceptions
- the provision of a unified abstraction across multiple layers
- routing or transforming messages conditionally, based on a non-centralized policy (without the need for a central rules-engine)
- queuing, holding messages if applications temporarily become unavailable

Key benefits

- Faster and cheaper accommodation of existing systems.
- Increased flexibility; easier to change as requirements change.
- Standards-based
- Scales from point-solutions to enterprise-wide deployment (distributed bus).
- Predefined ready-for-use service types.
- More configuration rather than integration coding.
- No central rules-engine, no central broker.
- Incremental patching with zero down-time; enterprise becomes "refactorable".

Key disadvantages

- Usually requires an Enterprise Message Model, resulting in additional management overhead. Potential difficulties integrating many disparate systems to collaborate via message standards.
- Requires ongoing management of message versions to ensure the intended benefit of loose coupling. Incorrect, insufficient, or incomplete management of message versions can result in tight coupling instead of the intended loose coupling.
- It normally requires more hardware than simple point-to-point messaging.
- Middleware analysis skills needed to configure, manage, and operate an ESB.
- Extra overhead and increased latency caused by messages traversing the extra ESB layer, especially as compared to point-to-point communications. The

increased latency also results from additional XML processing (the ESB normally uses XML as the communication language).

- Though ESB systems can require a significant effort to implement, they produce no commercial value without the subsequent development of SOA services for the ESB.

Chapter 5

Event-Driven Architecture

Event-driven architecture (**EDA**) is a software architecture pattern promoting the production, detection, consumption of, and reaction to events.

An *event* can be defined as "a significant change in state". For example, when a consumer purchases a car, the car's state changes from "for sale" to "sold". A car dealer's system architecture may treat this state change as an event to be produced, published, detected and consumed by various applications within the architecture.

This architectural pattern may be applied by the design and implementation of applications and systems which transmit events among loosely coupled software components and services. An event-driven system typically consists of event emitters (or agents) and event consumers (or sinks). Sinks have the responsibility of applying a reaction as soon as an event is presented. The reaction might or might not be completely provided by the sink itself. For instance, the sink might just have the responsibility to filter, transform and forward the event to another component or it might provide a self contained reaction to such event. The first category of sinks can be based upon traditional components such as message oriented middleware while the second category of sinks (self contained online reaction) might require a more appropriate transactional executive framework.

Building applications and systems around an event-driven architecture allows these applications and systems to be constructed in a manner that facilitates more responsiveness, because event-driven systems are, by design, more normalized to unpredictable and asynchronous environments.

Event-driven architecture can complement service-oriented architecture (SOA) because services can be activated by triggers fired on incoming events. This paradigm is particularly useful whenever the sink does not provide any self-contained executive.

SOA 2.0 evolves the implications SOA and EDA architectures provide to a richer, more robust level by leveraging previously unknown causal relationships to form a new event pattern. This new business intelligence pattern triggers further autonomous human or automated processing that adds exponential value to the enterprise by injecting value-added information into the recognized pattern which could not have been achieved previously.

Computing machinery and sensing devices (like sensors, actuators, controllers) can detect state changes of objects or conditions and create events which can then be processed by a service or system. Event triggers are conditions that result in the creation of an event.

Event structure

An event can be made of two parts, the event header and the event body. The event header might include information such as event name, timestamp for the event, and type of event. The event body is the part that describes the fact that has happened in reality. An event body must not be confused with the pattern or the logic that can be applied in reaction to the event itself.

Event flow layers

An event triggered architecture is built on four logical layers. It starts with the sensing of a fact, its technical representation in the form of an event and ends with a non-empty set of reactions to that event.

Event generator

The first logical layer is the event generator, which senses a fact and represents the fact into an event. Since a fact can be almost anything that can be sensed, so can an event generator. As an example, an event generator could be an email client, an E-commerce system or some type of sensor. Converting the different data collected from the sensors to one standardized data form that can be evaluated is a significant problem in the design and implementation of this layer. However, considering that an event is a strongly declarative frame, any transformational operations can be easily applied, thus eliminating the need for a high level of standardization.

Event channel

An event channel is a mechanism whereby the information from an event generator is transferred to the event engine or sink. This could be a TCP/IP connection or any type of input file (flat, XML format, e-mail, etc). Several event channels can be opened at the same time. Usually, because the event processing engine has to process them in near real time, the event channels will be read asynchronously. The events are stored in a queue, waiting to be processed later by the event processing engine.

Event processing engine

The event processing engine is where the event is identified, and the appropriate reaction is selected and executed. This can also lead to a number of assertions being produced. I.e., if the event that comes into the event processing engine is a “product ID low in stock”, this may trigger reactions such as, “Order product ID” and “Notify personnel”.

Downstream event-driven activity

This is where the consequences of the event are shown. This can be done in many different ways and forms; e.g., an email is sent to someone and an application may display some kind of warning on the screen.. Depending on the level of automation provided by the sink (event processing engine) the downstream activity might not be required.

Event processing styles

There are three general styles of event processing: simple, stream, and complex. The three styles are often used together in a mature event-driven architecture.

Simple event processing

Simple event processing concerns events that are directly related to specific, measurable changes of condition. In simple event processing, a notable event happens which initiates downstream action(s). Simple event processing is commonly used to drive the real-time flow of work, thereby reducing lag time and cost.

For example, simple events can be created by a sensor detecting changes in tire pressures or ambient temperature.

Event stream processing

In event stream processing (ESP), both ordinary and notable events happen. Ordinary events (orders, RFID transmissions) are screened for notability and streamed to information subscribers. Stream event processing is commonly used to drive the real-time flow of information in and around the enterprise, which enables in-time decision making.

Complex event processing

Complex event processing (CEP) allows patterns of simple and ordinary events to be considered to infer that a complex event has occurred. Complex event processing evaluates a confluence of events and then takes action. The events (notable or ordinary) may cross event types and occur over a long period of time. The event correlation may be causal, temporal, or spatial. CEP requires the employment of sophisticated event interpreters, event pattern definition and matching, and correlation techniques. CEP is commonly used to detect and respond to business anomalies, threats, and opportunities.

Extreme loose coupling and well distributed

An event driven architecture is extremely loosely coupled and well distributed. The great distribution of this architecture exists because an event can be almost anything and exist almost anywhere. The architecture is extremely loosely coupled because the event itself doesn't know about the consequences of its cause. e.g. If we have an alarm system that records information when the front door opens, the door itself doesn't know that the alarm system will add information when the door opens, just that the door has been opened.

Implementations and examples

Java Swing

The Java Swing API is based on an event driven architecture. This works particularly well with the motivation behind Swing to provide user interface related components and functionality. The API uses a nomenclature convention (e.g. "ActionListener" and "ActionEvent") to relate and organize event concerns. A class which needs to be aware of a particular event simply implements the appropriate listener, overrides the inherited methods, and is then added to the object that fires the event. A very simple example could be:

```
public class FooPanel extends JPanel implements ActionListener {
    public FooPanel() {
        super();

        JButton btn = new JButton("Click Me!");
        btn.addActionListener(this);

        this.add(btn);
    }

    @Override
    public void actionPerformed(ActionEvent ae) {
        System.out.println("Button has been clicked!");
    }
}
```

Alternatively, another implementation choice is to add the listener to the object as an anonymous class. Below is an example.

```
public class FooPanel extends JPanel {
    public FooPanel() {
        super();

        JButton btn = new JButton("Click Me!");
        btn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                System.out.println("Button has been clicked!");
            }
        });
```

Chapter 6

Enterprise Messaging System and Boot Image Control

Enterprise Messaging System

An **enterprise messaging system** (EMS) is a set of published Enterprise-wide standards that allows organizations to send semantically precise messages between computer systems. EMS systems promote loosely coupled architectures that allow changes in the formats of messages to have minimum impact on message subscribers. EMS systems are facilitated by the use of XML messaging, SOAP and web services.

EMS usually take into account the following considerations:

1. *Security*: Messages must be encrypted if they travel over public interfaces. Messages must be authenticated or digitally signed if the receiver is to have confidence that the messages have not been tampered with in transit.
2. *Routing*: Messages need to be routed efficiently from the sender to the receiver. Intermediate nodes may need to route the messages if the body of the message is encrypted.
3. *Metadata*: The body of the document contains information that must be unambiguously interpreted. Metadata registries should be used to create precise definitions for each data element.
4. *Subscription*: Systems should be able to subscribe to all messages that match a specific pattern. Messages with a specific content may be routed differently. For example some messages may have different priority or security policies.
5. *Policy*: Enterprise messaging systems should provide some consideration for a centralized policy of messages such as what classes or roles of users can access different fields of any message.

Separation of message header and message body

The design of an EMS is usually broken down into two sections:

1. *Message header design* – Message headers contain the information necessary to route messages. Message headers are usually coded in clear text so that intermediate nodes receive all the necessary information they need to route and prioritize the message. Message headers are analogous to the information printed on the outside of a letter (to, from, priority of message etc.)
2. *Message body semantics* – Message body semantics include the precise definition of all of the data elements in the body of the message. Message semantics can be aided by the use of a precise data dictionary that documents metadata.

Comparisons

Although similar in concept to an Enterprise Service Bus (ESB), an EMS places emphasis on design of messaging protocols, not the implementation of the services using a specific technology such as web services or Java Message Service.

Note that an Enterprise Messaging System should not be confused with an electronic mail system used for delivering human readable text messages to individual people.

An example of a specific application programming interface (API) that implements an Enterprise Messaging System is the Java Message Service (JMS). Although this is an API it embodies many of the same issues involved in setting up a full EMS.

Policy statements may also be extracted from a centralized policy server. These policy statements can be expressed in the XML Access Control Markup Language (XACML).

Boot Image Control

A **boot image control** strategy is a common way to reduce total cost of ownership in organizations with large numbers of similar computers being used by users with common needs, e.g. a large corporation or government agency. *This is considered part of enterprise application integration in larger shops that use that term since applications are part of the boot image, and modify the boot image, in most desktop OS.*

Windows Vista includes tools for boot image control, displacing third party tools. Mac OS has always had more flexible handling of boot drives, simplifying control and reducing the need to move boot images around between drives. Increasingly, boot image control is a network operating system function.

Economics

Very often a large computer vendor is required to explain in a bid in response to an RFP how they intend to simplify the purchaser's boot image control problems and the attendant service costs:

The total cost of ownership correlates strongly to the total number of different images, not the total number of computers, so this is a major cost concern. Three basic strategies are commonly advised:

- a single base boot image for each type of computer in the organization, customized by each user with no central control
- a thin client strategy where the smallest possible boot image is used, typically one that does not include a full operating system
- a departmental boot image strategy where a base boot image is customized with applications to fit each group of users, but, the users do not have the ability to upgrade or alter the configurations

Thin client strategies

Organizations that do not closely track, control and set common standards for, acquisition of new computer hardware, typically can only practice a thin client strategy.

Which strategy will reduce total cost of operations the most depends on several factors:

- whether the capabilities of a full operating system are required, or just those of a thin client
- whether applications with inflexible software licenses are in use that must be paid for not only if they are used, but even if they are only installed
- whether poorly-behaved applications that interact badly are in use
- LAN or removable disk limits that make it easy or difficult to do re-imaging on demand

More complex departmental boot images

While the departmental boot image strategy seems to be the most flexible, the complexity of creating and managing several large boot images, and determining when a department needs to upgrade its applications, can easily outweigh these. Especially if users object and try to subvert the discipline of waiting for a regular **boot turn** to upgrade all machines at once. If each user is allowed to do this on their own, then, the discipline soon degrades into effectively a bunch of home computer whose issues are not really diagnosable nor comparable to each other. In which situation thin clients may become the only practical answer:

Many organizations use thin clients for applications which require high security, involve unreliable users or repurpose older machines for continued use. This much simplifies

boot image control by facilitating centralized management of computers, and has many advantages:

- since servers manage clients and the local environment is highly restricted (and often stateless), providing protection from malware, support costs are reduced
- since no application data typically resides on the thin client (it is entirely rendered), it is securely stored on network drives upon its creation
- since disk, application memory, and processors are minimal in thin client hardware, they go obsolete slowly and cost much less
- since they are not as useful as ordinary computers they are of less interest to thieves

While control of the images is simpler, there are disadvantages. Thin clients:

- require more network bandwidth
- require more host computer power and must typically be served by much larger host boxes
- typically cannot run arbitrary Windows, Linux or Mac software
- perform poorly in multimedia applications or games - an advantage in many business environments

Many organizations try to gain the advantages of thin clients without the disadvantages by treating many very standard machines as if they were terminals, but with very much greater capabilities. As they buy new computers, they put the demanding applications on those.

Boot turns and re-imaging

Administrators perform a regular (often bi-annual) **boot turn** that re-images many older, off-spec machines at once so that new hardware can be deployed for higher-end use. This procedure is called cascading: the oldest hardware is repurposed with simpler software to let it continue in use for some less demanding or more access-controlled applications, but subjects it to much more rigorous **control** to minimize the number of images.

The total cost of operations correlates strongly to the total number of different images, not the total number of computers. To minimize the number of images requires additional discipline:

- Specify the computer hardware to minimize unneeded machine diversity and minimize the resultant number of boot images.
- Upgrade new machine specifications at low additional cost so they remain useful longer, reduce the incursion of off-spec machines later in the life-cycle, improve standardization, reduce support costs, minimize e-waste with longer lifecycles
- Organize the network so that boot images can be efficiently supported and swapped, independent of data.

    - Data must not be dependent on boot devices - use networks to store data on secure servers so that data recovery is literally never required even in a disaster recovery situation
- Confirm, by hardware acceptance testing on each new machine, that it runs the standard boot image
    - Any machine that does not must be considered to be **dead on arrival**
- A strict installation regime to ensure that only supportable standardized boot images are used and any machines that connect to the network for the first time with a nonstandard image are detected and rejected
- Diagnostics and troubleshooting so that help desk and other technical support staff can employ standardized tests to identify the source of problems: boot, software, or hardware
    - Ideally, backups on hand of the boot image, or even spare identical computers that can quickly be booted up from the boot device in question to determine if it is a hard disk, computer or software/image problem.
- Common desktop system recovery tools and procedures for failed desktop units, typically using backup copies of a boot image created with utilities
- Rapid network recovery procedures that replace a backup **boot image** in a few minutes or less, with considerable cost savings over using DVD, CD or floppy disk media which require human attention
- Ensure services for the disabled are on every departmental boot image that require them, or in the thin client hardware and software itself, to accommodate these users in a manner that is ubiquitous and cost effective.
- Support telework and secure off-site system access procedures in the standard boot image
    - Encourage teleworkers to buy identical machines to those in the office or use thin clients exclusively
- Facilitate worker transfer by changing boots or authorizations instead of moving the actual computer
- Install thin clients on all off-spec machines to eliminate the need for special boot images for them, and subsequent diagnostic problems and data risks.

Open configuration and semantic services

Desktop computing is increasingly relying on web services, making the thin client approach more viable. Departmental boot images may remain but simply instantiate part of a semantic service-oriented architecture, especially in larger organizations. A service component architecture would further simplify the implementation of control mechanisms, especially if a single application language like Java was used for all custom applications in the enterprise. More importantly, shift to software as a service by most large vendors means that applications are not tied to machines, so the number of variant boot images required (with the applications installed) is reduced.

Other open configuration technologies such as Bitfrost, OpenID and even XMPP would also simplify configuration of boot images, as authentication would no longer be dealt with on the desktop/laptop device.

Vendor support

Large system vendors increasingly provide DVDs with the **boot image** standard for the machine as shipped to the customer, which usually includes tools to diagnose changes to the machine and download drivers.

Chapter 7

IBM WebSphere

In computing, **IBM WebSphere** refers to a brand of software products in the genre of enterprise software known as "application and integration middleware". These software products are used by end-users to create applications and integrate applications with other applications. IBM WebSphere has been available to the general market for over a decade.

Occasionally, the term "IBM WebSphere" also refers in popular usage to one specific product: **IBM WebSphere Application Server** (WAS).

History

IBM introduced the first product in this brand, *IBM WebSphere Performance Pack*, in June 1998. As of 2010 it forms a part of IBM WebSphere Application Server Network Deployment.

IBM WebSphere software

The following complete list of IBM WebSphere software uses IBM classifications. Several tools appear in more than one category.

Application Servers

Distributed Application & Web Servers

Application integration, data access and integration, business processing and distributed transaction monitoring:

- IBM WebSphere Application Server

In Memory DataGrid or distributed cache

Partitioned, replicated and elastic storage of data in a grid of machines. Can also be used for distributed caching integration

- IBM WebSphere eXtreme Scale ()

Other Application Servers

Other platforms on which to run inter-operable applications:

- Remote Server
- WebSphere sMash (Project Zero)

Business Integration

Application Integration and Connectivity

Application Integration and Connectivity middleware reduces the complexity of connecting applications to applications.

- WebSphere Adapters (JCA-based) and WebSphere Business Integration Adapters
- Data Interchange
  - for MultiPlatforms
  - for z/OS
- IBM WebSphere DataPower SOA Appliances
  - DataPower Integration Appliance XI50
  - DataPower XML Security Gateway XS40
  - DataPower XML Accelerator XA35
- IBM WebSphere ESB
- IBM WebSphere Message Broker
  - for MultiPlatforms
  - for z/OS
  - with Rules and Formatter Extension for Multiplatforms
  - with Rules and Formatter Extension for z/OS
- IBM WebSphere MQ
  - Express
  - Extended Security Edition
  - for z/OS
  - V6
- WebSphere MQ Everyplace
  - Network Edition
  - Retail Edition
- IBM WebSphere Service Registry and Repository
- WebSphere Transformation Extender
- WebSphere Business Integration for Financial Networks

    - v2.2
        - for z/OS
        - for AIX

Process integration

Runtime and infrastructure for [real-time computing|real-time] application integration, event-driven processing and process automation.

- WebSphere Business Events
- Business Integration Server Express
- Business Integration Workbench Entry Edition
- Business Integration Workbench Server
- Business Modeler
    - Advanced
    - Basic
    - Publishing Server
- Business Monitor
- Business Services Fabric
- Event Broker
- IBM WebSphere Integration Developer
- WebSphere Lombardi Edition
- Partner Gateway
    - Advanced Edition
    - Enterprise Edition
    - Express
- IBM WebSphere Process Server

Commerce

Web commerce

Platform framework for e-commerce, including marketing, sales, customer and order processing functionality in a tailorable, integrated package.

- IBM WebSphere Commerce
    - Enterprise
    - Professional
    - Express

Mobile, Speech and Enterprise Access

Device Software

Device software consists of client-sMicro Environment - this support many platforms, for instance, see their WEME 6.1.1 evaluation platforms.

Mobile and Enterprise Access

- Everyplace Access
- Everyplace Deployment, which has evolved into IBM Lotus Expeditor
- Everyplace Mobile Portal Enable

Speech

Delivers Business-to-employee (B2E) and Business-to-consumer (B2C) services, including voice recognition and telephony speech processing:

- Embedded ViaVoice
- Unified Messaging for WebSphere Voice Response

WebSphere

- Everyplace Subscription Manager
- Voice Response for AIX
- Voice Server

Translation

Translation applications convert languages automatically and assist humans performing internationalization tasks.

- Translation Server, for Multiplatforms

Other Mobile, Speech and Enterprise Access

- Everyplace Device Manager
- Everyplace Mobile Portal
- Everyplace Server for Telecom
- IP Multimedia Subsystem Connector
- WebSphere Presence Server
- Radio-frequency identification (RFID) Premises Server
- WebSphere Telecom Web Services Server

Networking

Host Access

Provides multi-protocol transparency and control and connectivity (protocol stacks and terminal emulation) to host applications.

- WebFacing Deployment Tool with HATS Technology

- WebSphere Host Access Transformation Services
- WebSphere Host Integration Solution
- WebSphere Host On-Demand

Organizational productivity, portals and collaboration

Portals

Portals provide personalized access to a variety of applications and aggregate disparate content sources and services. Portals allow people to customize their user experience, with personalized applications based on role, context, actions, location, preferences and team-collaboration needs. There are many other things that can be brought into a portal site.

- WebSphere Portal
- WebSphere Portlet Factory

Software Development

Integrated Development Environment

Application development tools for Websphere

- IBM Rational Application Developer
- Application Server Toolkit

Analysis modeling and design

Assists in creating resilient architectures for Service-oriented architecture (SOA), programming specifications, business processes and rules.

- Branch Transformation Toolkit for WebSphere Studio
- WebFacing Deployment Tool with HATS Technology
- WebSphere Developer
- WebSphere Development Studio Client Advanced Edition for iSeries
- WebSphere Studio Asset Analyzer
- WebSphere Studio Device Developer

Problem-determination tools

Problem-determination and -debugging tools:

- Application Performance Analyzer for System z
- Debugger Tool for System z
- File Manager for System z
- Fault Analyzer for System z

- Optim Move for DB2
- Workload Simulator for OS/390 and System z

Process and portfolio management

Implements and manages enterprise processes and investments:

- IBM Asset Transformation Workbench
- IBM Workbench

Software quality management

Tools that address all dimensions of software quality: functionality, reliability and performance:

- Studio Workload Simulator for z/OS and OS/390

Traditional programming-languages and compilers

3GL and 4GL/RAD language-based tools and unified development environments.

- COBOL Family

Systems management

Application performance and availability

Define, measure, and manage to committed service-levels across complex heterogeneous environments with central control.

- Studio Application Monitor

Enterprise content management

- Information Integrator, Content Edition (IICE)
- IBM WebSphere Information Integration

Chapter 8

SAP NetWeaver

SAP NetWeaver is SAP's integrated technology platform and is the technical foundation for all SAP applications since the SAP Business Suite. SAP NetWeaver is marketed as a service-oriented application and integration platform. SAP NetWeaver provides the development and runtime environment for SAP applications and can be used for custom development and integration with other applications and systems. SAP NetWeaver is built using open standards and industry de facto standards and can be extended with, and interoperate with, technologies such as Microsoft .NET, Oracle Java EE, and IBM WebSphere.

SAP NetWeaver's release is considered as a strategic move by SAP for driving enterprises to run their business on a single, integrated platform that includes both applications and technology. Industry analysts refer to this type of integrated platform offering as an "applistructure" (applications + infrastructure). According to SAP, this approach is driven by industry's need to lower IT costs through an enterprise architecture that is at once (1) more flexible; (2) better integrated with applications; (3) built on open standards to ensure future interoperability and broad integration; and, (4) provided by a vendor that is financially viable for the long term.

SAP is fostering relationships with system integrators and independent software vendors, many of the latter becoming "Powered by SAP NetWeaver".

SAP NetWeaver is part of SAP's plan to transition to a more open, service-oriented architecture and to deliver the technical foundation of its applications on a single, integrated platform and common release cycle.

History

SAP announced first NetWeaver release, named Netweaver 2004, in January 2003, and it was made available on March 31, 2004.

NetWeaver 7.0, aka 2004s, was made available on October 24, 2005.

Composition

NetWeaver is essentially the integrated stack of SAP technology products. The SAP Web Application Server (sometimes referred to as WebAS) is the runtime environment for the SAP applications—all of the mySAP Business Suite solutions (SRM, CRM, SCM, PLM, ERP) run on SAP WebAS.

Products

The core products that make up SAP NetWeaver include:

- SAP NetWeaver Application Server
- SAP NetWeaver Business Intelligence
- SAP NetWeaver Composition Environment (CE)
- SAP NetWeaver Enterprise Portal (EP)
- SAP NetWeaver Identity Management (IdM)
- SAP NetWeaver Master Data Management (MDM)
- SAP NetWeaver Mobile
- SAP NetWeaver Process Integration (PI)

SAP has also teamed with hardware vendors like HP, IBM, Fujitsu-Siemens, and Sun to deliver appliances (i.e., hardware + software) to simplify and enhance the deployment of NetWeaver components. Examples of these appliances include:

- BW Accelerator
- Enterprise Search

Development Tools

- ABAP Workbench (SE80)
- SAP NetWeaver Developer Studio (NWDS) based on Eclipse for most of the Java part of the technology (Web Dynpro for Java, JEE, Java Dictionary, Portal Applications etc.)
- SAP Netweaver Development Infrastructure (NWDI)
- Visual Composer

Features

- SOAP and Web Services
- Interoperability with Java EE
- Interoperability with .NET (Microsoft)
- Integration of Business Intelligence
- xApps
- Duet

Specifically, ERP is being extended by Business Process Management Systems (BPMs) and, as BPMs takes hold as the pre-dominant technical platform for new applications, expect to see radical changes to ERP architecture in the years ahead. The technology has been applied to a wide range of industries and applications.

SAP's Netweaver platform is still backwards-compatible with ABAP, SAP's custom development language.

Chapter 9

SOALIB

Service Oriented Architecture Library (**SOALIB**) is used to distribute reusable Service Oriented Architecture (SOA) software in a manner similar to other computing libraries. SOA consists of loosely coupled interoperable services which use messaging based on both Simple Object Access Protocol (SOAP) and Representational State Transfer (REST). A library in computing is a set of compiled modules which are tested and ready for reuse. A similar concept is used for SOA, in that whatever technology is used to develop the service can also be distributed in library form. A Java-based SOA library may be distributed in Web ARchive (WAR) or Enterprise Archive (EAR) file formats. C, C++, and .NET applications may be distributed as a shared object (in Unix and Linux), a Dynamic Link Library (in Windows), or as an executable file.

History

Service Oriented Architecture is usually tied to the redesign of an entire software system and determines how to decompose the single software unit into loosely coupled components, in which each loosely coupled components acts as interoperable services. Such a task is enormous and may take significant amount of time, while on the atomic level (where *atom* is defined as a single loosely-coupled service that is self-contained), most services are reusable regardless of the application. As an example, all matter is built with atoms, yet all material things are different. At the atomic level, however, they appear uniform. Similarly, all software can be built on loosely-coupled services which serve as the "atoms" of the redesign process. Because loose coupling is difficult to determine, the opposite is not true. That means it is easier to build a complete software system by using available loosely-coupled services.

By building Service Oriented Architecture Libraries, each of which are loosely-coupled services, complex applications can be developed by making use of these services. Because new applications depend on all loosely-coupled services, as long as we stick to loose coupling, the final application is also loosely-coupled. While it is true that the final application depends on many hierarchical loosely-coupled systems, it remains loosely-coupled due to the fact that all the hierarchy is based on atomic services.

Objectives

Building SOA requires the loose coupling of services as a starting point. They are termed as *atomic services*. The first step is to determine the atomic services. Then build these atomic services for reuse. A large number of such atomic services could be created, upon which composite services will be built. Composite service are services that are built only upon atomic services.

Steps

- Identify, build, and test loosely coupled *atomic* services.
- Identify, build, and test loosely coupled composite services, where each composite services is made up of atomic services only.
- Build integrated services, where each of the integrated services are made up of composite and atomic services.
- Build complex software system via the reuse of atomic, composite, and integrated services. This complex system remains loosely-coupled.

Platform independence

Consideration for cross platform is important. Presently, there are many ways to make the services host platform to be platform independent. Examples are building services in Java, where JVM is available for the host in which the server will run as a service. Alternatives are building applications with full compliance to ANSI C/C++ so that none of the components of the code is reliant upon third party libraries. Typically, this means building the C/C++ applications using GNU tools and GNU C compilers because GNU compilers are ported to most operating systems and platforms. Another alternative is to use C# .NET as the language of the web service where the Common Language Runtime is ported to the target operating system and platform. Many other options are available, but the most common approaches to the platform independence has already been mentioned.

Steps

- Full platform independence to the extent possible.
- Automatic tests on each platforms.

Multi-vendor database

The scope of the library is limited if it does not have support for databases. Composite and integration services should be built to take advantage of loosely-coupled atomic services which operate on databases. Once the support for database access is added, a metadata layer should be added to map the various kinds of data format in the database into a uniform set of data types which will have equal meaning to all databases. This is the difficult part, but at this time, JDBC , ODBC, ADO and other standards database drivers already do part of the task. Each database may store data in different formats.

Some data may be encrypted, some may be stored in little endian Endianness, or others in big endian, and so on. Therefore, data must be transformed at some point by the services and they have to be done at runtime due to the changing nature of data. All databases are different, therefore, no single API can take advantage of all the features of all databases. Therefore, services should use database specific features as well.

Steps

- Full support for all major databases - mobile, PC, and server-based.
- Creation of a metadata layer for uniform data access.
- All database access through the service libraries.
- Automatic runtime data transformation.
- Support internal database specific features.

Data synchronization

Once multi-vendor databases are supported, services can be added so that each database may synchronize with any other database. This will now become possible as all data is now passed through a metadata layer and data is represented in some intermediate form. The intermediate form may be any native form and does not have to be standards-based. Usually, the native form should be portable, that means, representing it in XML may be the best approach. Because data sizes tend to be large, incremental change detection should also be added.

Steps

- Any-to-any synchronization.
- Change capture engine.
- Uni- and bi-directional data synchronization.
- Custom mapping with referential integrity.
- Heterogeneous synchronization.

Security

All libraries must be secure. If the libraries have SOA implementation as web services, then it should have WS-Security, WS-Policy and other WS-type standards compliance. If based on REST or other protocols, then it should follow the respective standards. All libraries must at least support SSL.

Steps

- Support all major standards-based security architectures.
- Secure Socket Layer support.
- Option for encrypted storage on the server.

Interoperability

Interoperability is one of the most important reasons why SOA has been so important. A SOA library must also include the required API that could be used for rapid platform specific development.

Steps

- Support all major programming languages: Java, Java ME, C, C++, C#.NET, VB.NET., PHP
- Supplied with all needed client APIs.
- All APIs must be tested against the binding SOA services.

Building applications atom by atom

An atomic service is a loosely-coupled service which is independent of any assumptions, absolutely predictable and have no other dependencies on services or other atomic services. As an example, a disk file operation may be considered as an atomic service in which the only operations performed by the service are read, write, delete, or append file operations. Because the only information the disk file would need is the full path to the file and possibly some access parameters (like username and password), there would be no other dependencies. If a composite service is designed based on atomic services, it is still loosely coupled, but not an atomic service. Each integrated service can then be built together to make a larger SOA application. By building layer by layer, a large SOA application may be created which will remain loosely-coupled.

General guidelines

To keep the integrated services loosely-coupled would require all the services to be built on atomic and composite services. As soon as the integrated service uses another service that is somewhat tightly-coupled, the entire service application becomes tightly-coupled. This is analogous to atomic structures becoming "charged" if there is just one electron missing. For this reason, when using third party services, the designer must ensure that the service remains loosely-coupled.

- If the service is reliant on third party services, those services must also be loosely-coupled.
- If third part offers atomic services, then composite services may be built by mixing Service Oriented Architecture Libraries as well as third party atomic services.
- If any of the services is deemed to be tightly-coupled, which may become necessary when there is an industrial appliance involved (e.g., Robotic Arms, Consumer Appliances, etc.), this should be the final service and no service should be derived from them. If other services are built on tightly-coupled services, the derived services are tightly-coupled as well. These derived services may be used

in specialized applications where tight coupling is required (e.g., in precision machines).

The following is the hierarchy in which all service oriented applications should be designed.

Hierarchy

Ideally, the following should be the SOA application design approach:

- Integrated Services - *based upon composite and atomic services*
    - Composite Services - *based only upon atomic services*
        - Atomic Services - *no dependency, this service is the atom*

Structures to avoid

In some cases, loose coupling may not be possible due to reliance on hardware, mechanical systems, or specialized instruments. For example, if there is a service built to move a robotic arm, monitor industrial generators, or emergency hospital equipment. Then, tightly-coupled services are required. Tightly-coupled services should be the top in the hierarchy such that no other service may reuse tightly-coupled services, if possible. If there are derived services based on tightly-coupled services, then all derived services also become tightly-coupled. Such a system, if designed, should be limited to the scope of the purpose of the application.

Chapter 10

Governance Interoperability Framework and Search-Based Application

Governance Interoperability Framework

The **Governance Interoperability Framework** (GIF) is an open, standards-based specification and set of technologies that describes and promotes interoperability among components of a service-oriented architecture (SOA). GIF integrates SOA ecosystem technologies to achieve heterogeneous service lifecycle governance and is supported by Hewlett-Packard Company and by GIF partners.

SOA Governance and creating a system-of-record

Governance is recognized as a foundational requirement for successful enterprise adoption of SOA: Gartner has stated that "governance isn't an option but an imperative", and predicts that the dominant mode of SOA project failure will be a lack of adequate governance.

The primary products used by most organizations to achieve SOA governance are based on an integrated registry-repository, and provide support for managing and communication information in an SOA as well as automating key governance activities. These SOA governance systems provide a central system-of-record for all services and related information in an SOA, and are the place where services can be advertised by providers and discovered by consumers. As such, they act as a key control point for governing service availability, versioning, service lifecycle management, and for ensuring compliance with business and technical policies.

To be effective, SOA governance systems need a mechanism for exchanging information between all the disparate technologies that support an SOA. Interoperability is a fundamental requirement for the visibility, trust and control required for effective SOA

governance. The objective of GIF is to drive interoperability through the adoption of standards and common approaches to modeling and exchanging information.

GIF Overview

GIF represents a collection of APIs defined by standards organizations, data mappings and classifications and leverages UDDI and WS-Policy standards, among others, as building blocks. In order to promote commonality of approaches and understanding of the information represented, GIF also defines vocabularies for the purpose of applying metadata to service information.

Integration with the Governance Interoperability Framework is based on two primary pillars of integration: Control Integration and Service Data Integration. These themes are based on the famed Model-View-Controller (MVC) pattern:

Control integration - Consists of alerting and notification integration; launching events and actions; and integration of business service governance and lifecycle.

Data integration - Consists of leveraging the Business Service Registry as the primary service description, characteristic, and policy catalog.

GIF provides control and data integration needed to support activities such as the governance of business service provisioning and lifecycle management. Aspects of this are:

Provisioning integration - Leverage the SOA governance system as part of the provisioning and deployment process of business services. Once integrated, bi-directional exchange of service information between participants is enabled.

Deployment integration - Upon deployment of services, any party should have the ability to alert others to the existence of the service and the need to put the service and its definitions under management.

Lifecycle management - Lifecycle management of all facets of a business service is required. This means collaborating and integrating components for the purpose of managing:

- service and artifact versioning
- lifecycle information (e.g. development, test, production, deprecated)
- lifecycle status and state of the service (availability)
- deployment information including up-time, first deployed date, last deployment date
- contact and support information about the service or template (owner, responsible organization, support contact, manager, etc.)
- compliance status
- dependencies and relationships between:

- services – those dependencies that the partner creates partner
- proxies and the services they proxy
- services consumed by another service

GIF has been driven by several use cases, including:

- SOA governance system acting as the authoritative source of service descriptions and metadata to other components of an SOA ecosystem.
- SOA governance system as the recipient of service descriptions, metadata and policy information, including but not limited to publishing service descriptions (typically WSDL documents and associated metadata) and lifecycle information.
- Publish service characteristics to the SOA governance system to enable the searching and reporting of services and their artifacts. This may include publishing performance data, historical trends and other facets of service information such as constraints and capabilities of the service.
- Make service characteristics, such as configuration, constraints and capabilities typically expressed as policy, discoverable.
- Publishing and making available protocol and binding information, such as the WSDLs of proxied services by security or management partners, and their associated business service (functional) WSDLs.

GIF organization and membership

GIF is not a standard itself, but rather leverages existing standards to support SOA governance interoperability. GIF is supported by Hewlett-Packard Company and by GIF partners. For more information about the GIF specification, existing GIF partners and how to join GIF, visit HP's website.

Search-Based Application

Search-based applications (**SBA**) are software applications in which a search engine platform is used as the core infrastructure for information access and reporting. SBAs use semantic technologies to aggregate, normalize and classify unstructured, semi-structured and/or structured content across multiple repositories, and employ natural language technologies for accessing the aggregated information.

Pre-Conditions

Search based applications are fully packaged applications that:

- Are built on a search backbone to enable sub-second access to information in multiple formats and from multiple sources
- Are delivered as a unified work environment to support a specific task or workflow, for example: eDiscovery, financial services regulatory compliance,

fraud detection, voice of the customer, sales prospecting, pharmaceutical research, anti-terrorism intelligence, or customer support.
- Integrate all the tools that are commonly needed for that a specific task or workflow, including:
- Multi-source information access
- Authoring
- Collaboration
- Business process
- Reporting and analysis
- Alerting
- Visualization
- Provide pre-configured data integration with multiple repositories of information in multiple formats as appropriate for the application domain.
- Integrate domain knowledge to support the particular task, including industry taxonomies and vocabularies, internal processes, workflow for the task, connectors to specialized collections of information, and decision heuristics typical of the field.
- Provide a compelling user interface and interaction design that eliminates the need for users to "pogo stick" or continually jump from one application to another. This buffers the user from the complexity of operating separate applications and enables them to focus on getting work done.
- Are quick to deploy, easy to customize or extend, and economical to administer

Source: Worldwide Search and Discovery 2009 Vendor Shares and Update on Market Trends, IDC #223926, July, 2010 by Susan Feldman and Hadley Reynolds.

Practical Uses

SBAs are used for a variety of purposes, including:

- **Enterprise Business Applications:** For example, Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Supply Chain Management (SCM), Compliance & Discovery, and Business Intelligence (BI)

- **Web Applications:** Typically, B2B, B2C and C2B applications that mash-up data and functionality from diverse sources (databases, Web content, user-generated content, mapping data and functions, etc.)

The use of a search platform as the core infrastructure for software applications has been enabled largely by two search engine features: 1) Scalability 2) Ad hoc access to multiple heterogeneous sources from a single point of access.

Search based applications have proven popular and effective because they provide a dynamic, scalable access infrastructure that can be integrated with other features that information workers need: task-specific, and easy to use work environments that integrate

features that are usually designed to be used as separate applications, collaborative features, domain knowledge, and security.

Search engines are not a replacement for database systems; they are a complement. They have been optimally engineered to facilitate access to information, not to record and store transactions. In addition, the mathematical and statistical processors integrated to date into search engines remain relatively simple. At present, therefore, databases still provide a more effective structure for complex analytical functions.Search applications also focus on providing quality results considering search relevancy.

Chapter 11

Guarana DSL and Pervasive Business Intelligence

Guarana DSL

Guaraná DSL is a Domain-Specific Language (DSL) to design enterprise application integration (EAI) solutions at a high-level of abstraction. The resulting models are platform-independent, so engineers do not need to have skills on a low-level integration technology when designing their solutions. Furthermore, this design can be re-used to automatically generate executable EAI solutions for different target technologies

Functionality and structure of an EAI solution are completely defined by using the language building blocks, ports, tasks, decorator, slots and integration links. Guaraná's tasks are based on the Enterprise Integration Patterns (EIP Patterns) by Gregor Hohpe and Bobby Woolf. It is possible to design the internal structure of all kinds of building blocks (wrappers and integration processes) and its communication ports (entry port, exit port, solicitor port and responder port) by using tasks; it is also possible, to create integration flows that allow applications to collaborate by connecting these building blocks by means of integration links. Applications that participate in the integration solution are documented using decorators as well as its layers being used as communication interface.

Main constructors in Guaraná DSL

Below you have a list of the main constructor of Guaraná DSL.

Decorators: to provide visual information about the participating applications in the EAI solution and their layer(s).

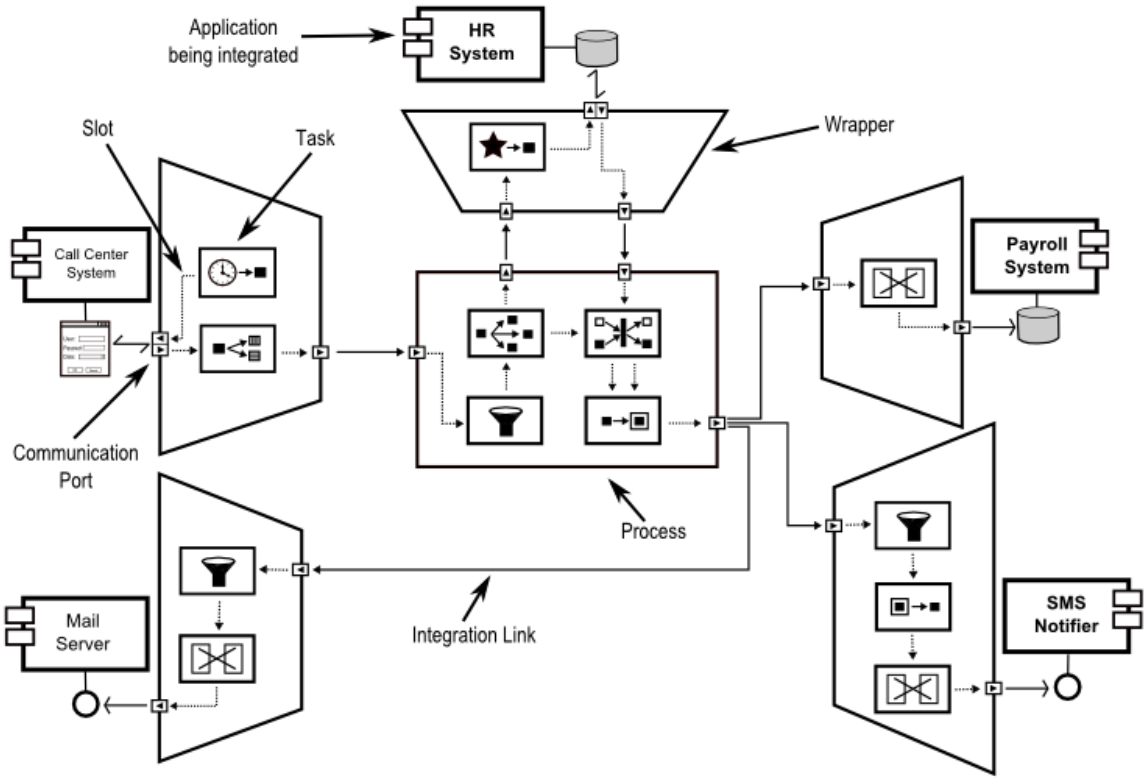
Processes: serve two purposes, namely: there are processes that allow to wrap applications and processes that allow to integrate them. The former are reusable processes that endow an application with a message-oriented API that simplifies interacting with it. Implementing such a wrapping process may range from using a JDBC driver to interact with a database to implementing a scrapper that emulates the behaviour of a person who interacts with a user interface. Generally speaking, this is known as wrapping an application in the literature. Integration processes, on the contrary, are intended to orchestrate the interactions with a number of wrapping processes and other integration processes. Processes rely on tasks to perform their wrapping or their orchestration activities. Simply put, a process can be viewed as a message processor.

Slots: are memory buffers used within building blocks for port to task and task to task internal communications.

Tasks: are message processing constructors and appear inside processes and wrappers. A task reads messages from incoming slots, processes them (e.g. enriches, translates, filters, etc.) and deposits the result in the outcome slot. Part of them are based on the enterprise integration patterns proposed by Gregor Hohpe and Bobby Woolf.

Ports: are used to communicate the internal building blocks of an EAI solution and the EAI solution with its applications. Integration links: are channels that transport messages between building blocks. They are used to connect the entry/exit ports used by building blocks.

Below you can see an example of EAI solution designed with Guaraná DSL:



Pervasive Business Intelligence

Pervasive Business intelligence is an architecture or frameworkfor working with Business Intelligence. It combines the practices of Enterprise Architecture with Service Oriented Architecture and Active Data Warehousing to provide enterprises with a more holistic approach to Business Intelligence. The aim is to facilitate tactical decisions by establishing a data warehouse which stores both historical data and real time data that is available instantly and accessible 24/7/365. It could be valuable to share access to the data warehouse with various stakeholders, clients and suppliers as they might gain and provide insight.

It is however important to note that the goal with Pervasive Business Intelligence is not only to provide near realtime data. The goal is to reduce lag or latency between events and the subsequent action.

Active Data Warehousing

An Active Data Warehouse is an important part of pervasive BI as it provides historical data infused with real time data on the fly either pushed or pulled to the user. The difference between a traditional data warehouse and one that is active is in the way it handles events. An active data warehouse will have pre-defined rules, schedules or event triggers which cause a certain automatic action such as notifying the user, giving a discount to a certain customer, etc.

The definitions of the ADW active elements are quoted from Teradata: Active Data Warehousing:

- Active Access

Front-line users access the data warehouse for operational decision making with a service level agreement of five seconds or less (also known as "web speed").

- Active Load

Near-real-time data enters the data warehouse via mini-batch loading, replication services or continuous streams of data from message queuing systems.

- Active Events

Event-driven architectures and business activity monitoring detect significant business events and issue alerts for timely, informed decisions.

- Active Workload Management

Dynamic priority management inside the data warehouse ensures service levels are achieved across multiple user communities and workload types.

- Active Enterprise Integration

Integration tools and designs connect the data warehouse to web sites, portals, SOA, web services, enterprise service busses, workflow and batch systems.

- Active Availability

Policies, procedures and redundant hardware ensure the entire information supply chain is protected when subsystem failure occurs.

Importance of Service Level Agreements

In order to take full advantage of Pervasive BI, Teradata suggests that the data and architecture is held up against a Service Level Agreements (SLA) which guarantees that the system lives up to certain minimal expectations. The categories for the Data SLA are;

- Data Freshness
- Data Completeness
- Data Cleansing and Accuracy

The categories for the BI SLA are;

- Scalability
- Performance
- Reliability

Usage

Pervasive Business Intelligence has been used in the following tasks:

- Inbound and outbound cross selling by Customer Service

Representatives

- Point-of-sale fraud detection
- Automated insurance claims triage and fraud detection
- Personalized next best offer on web sites, ATMs, and POS

devices

- Supply chain business activity monitoring
- Retail out-of-stock monitoring on promotional items
- Labor and crew scheduling
- Optimizing trailer and container loading and routing
- Anti-money laundering
- Dynamic pricing and yield management
- Real-time manufacturing line quality alerts

Chapter 12

Data Integration

Data integration involves combining data residing in different sources and providing users with a unified view of these data. This process becomes significant in a variety of situations both commercial (when two similar companies need to merge their databases) and scientific (combining research results from different bioinformatics repositories, for example). Data integration appears with increasing frequency as the volume and the need to share existing data explodes. It has become the focus of extensive theoretical work, and numerous open problems remain unsolved. In management circles, people frequently refer to data integration as "Enterprise Information Integration" (EII).

History

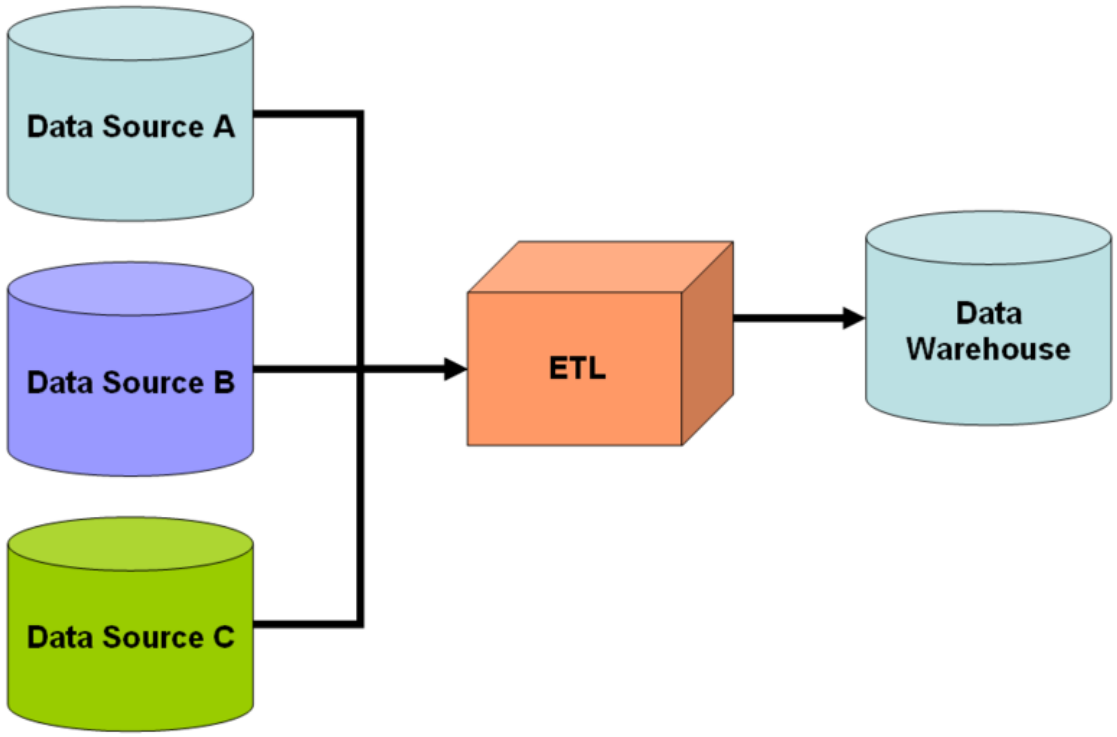


Figure 1: Simple schematic for a data warehouse. The ETL process extracts information from the source databases, transforms it and then loads it into the data warehouse.

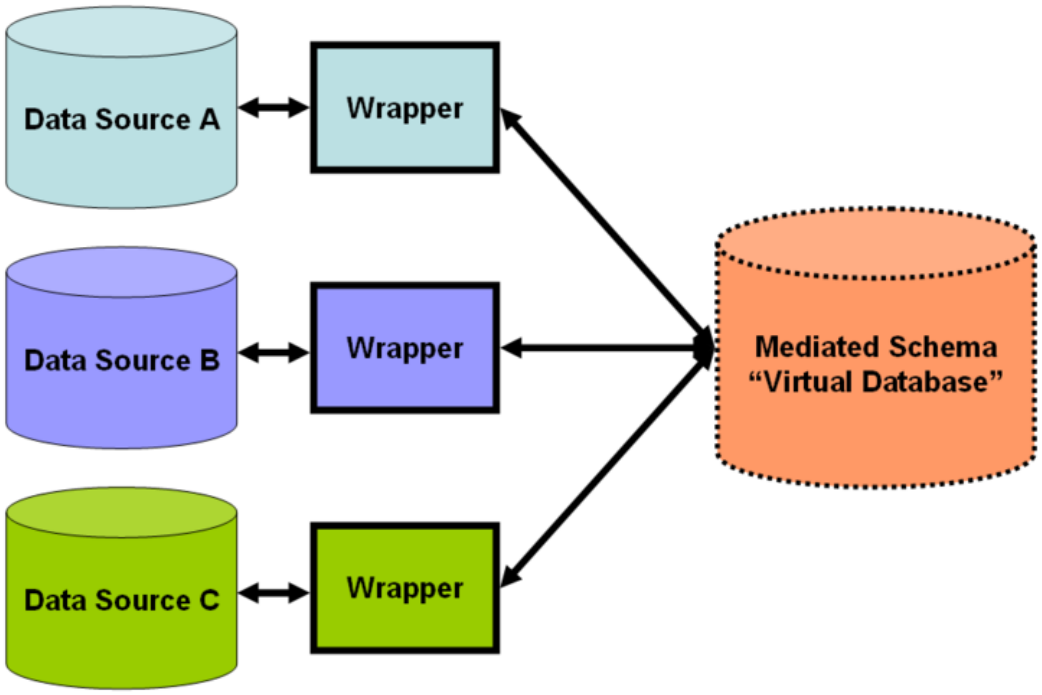


Figure 2: Simple schematic for a data-integration solution. A system designer constructs a mediated schema against which users can run queries. The virtual database interfaces with the source databases via wrapper code if required.

Issues with combining heterogeneous data sources under a single query interface have existed for some time. The rapid adoption of databases after the 1960s naturally led to the need to share or to merge existing repositories. This merging can take place at several levels in the database architecture. One popular solution involves data warehousing (see figure 1). The warehouse system extracts, transforms, and loads data from several sources into a single queriable schema. Architecturally, this offers a tightly coupled approach because the data reside together in a single repository at query-time. Problems with tight coupling can arise with the "freshness" of data; for example, when an original data source gets updated, but the warehouse still contains the older data and the ETL process needs re-execution. Difficulties also arise in constructing data warehouses when one has only a query interface to summary data sources and no access to the full data. This problem frequently emerges when integrating several commercial query services like travel or classified advertisement web applications.

As of 2009 the trend in data integration has favored loosening the coupling between data. This may involve providing a uniform query-interface over a mediated schema (see figure 2), thus transforming a query into specialized queries over the original databases.

One can also term this process "view-based query-answering" because each of the data sources functions as a view over the (nonexistent) mediated schema. Formally, computer scientists label such an approach "Local As View" (LAV) — where "Local" refers to the local sources/databases. An alternate model of integration has the mediated schema functioning as a view over the sources. This approach, called "Global As View" (GAV) — where "Global" refers to the global (mediated) schema — has attractions due to the simplicity involved in answering queries issued over the mediated schema. However, one must rewrite the view for the mediated schema whenever a new source gets integrated and/or an existing source changes its schema.

As of 2010 some of the work in data integration research concerns the semantic integration problem. This problem addresses not the structuring of the architecture of the integration, but how to resolve semantic conflicts between heterogeneous data sources. For example if two companies merge their databases, certain concepts and definitions in their respective schemas like "earnings" inevitably have different meanings. In one database it may mean profits in dollars (a floating-point number), while in the other it might represent the number of sales (an integer). A common strategy for the resolution of such problems involves the use of ontologies which explicitly define schema terms and thus help to resolve semantic conflicts. This approach represents ontology-based data integration.

Example

Consider a web application where a user can query a variety of information about cities (such as crime statistics, weather, hotels, demographics, etc). Traditionally, the information must exist in a single database with a single schema. But any single enterprise would find information of this breadth somewhat difficult and expensive to collect. Even if the resources exist to gather the data, it would likely duplicate data in existing crime databases, weather websites, and census data.

A data-integration solution may address this problem by considering these external resources as materialized views over a virtual mediated schema, resulting in "virtual data integration". This means application-developers construct a virtual schema — the *mediated schema* — to best model the kinds of answers their users want. Next, they design "wrappers" or adapters for each data source, such as the crime database and weather website. These adapters simply transform the local query results (those returned by the respective websites or databases) into an easily processed form for the data integration solution (see figure 2). When an application-user queries the mediated schema, the data-integration solution transforms this query into appropriate queries over the respective data sources. Finally, the virtual database combines the results of these queries into the answer to the user's query.

This solution offers the convenience of adding new sources by simply constructing an adapter or an application software blade for them. It contrasts with ETL systems or with a single database solution, which require manual integration of the entire new dataset into the system. The virtual ETL solutions leverage virtual mediated schema to implement

data harmonization; whereby the data is copied from the designated "master" source to the defined targets, field by field. Advanced Data virtualization is also built on the concept of object-oriented modeling in order to construct virtual mediated schema or virtual metadata repository, using hub and spoke architecture.

Theory of data integration

The theory of data integration forms a subset of database theory and formalizes the underlying concepts of the problem in first-order logic. Applying the theories gives indications as to the feasibility and difficulty of data integration. While its definitions may appear abstract, they have sufficient generality to accommodate all manner of integration systems.

Definitions

Data integration systems are formally defined as a triple $\langle G, S, M \rangle$ where *G* is the global (or mediated) schema, *S* is the heterogeneous set of source schemas, and *M* is the mapping that maps queries between the source and the global schemas. Both *G* and *S* are expressed in languages over alphabets composed of symbols for each of their respective relations. The mapping *M* consists of assertions between queries over *G* and queries over *S*. When users pose queries over the data integration system, they pose queries over *G* and the mapping then asserts connections between the elements in the global schema and the source schemas.

A database over a schema is defined as a set of sets, one for each relation (in a relational database). The database corresponding to the source schema *S* would comprise the set of sets of tuples for each of the heterogeneous data sources and is called the *source database*. Note that this single source database may actually represent a collection of disconnected databases. The database corresponding to the virtual mediated schema *G* is called the *global database*. The global database must satisfy the mapping *M* with respect to the source database. The legality of this mapping depends on the nature of the correspondence between *G* and *S*. Two popular ways to model this correspondence exist: *Global as View* or GAV and *Local as View* or LAV.

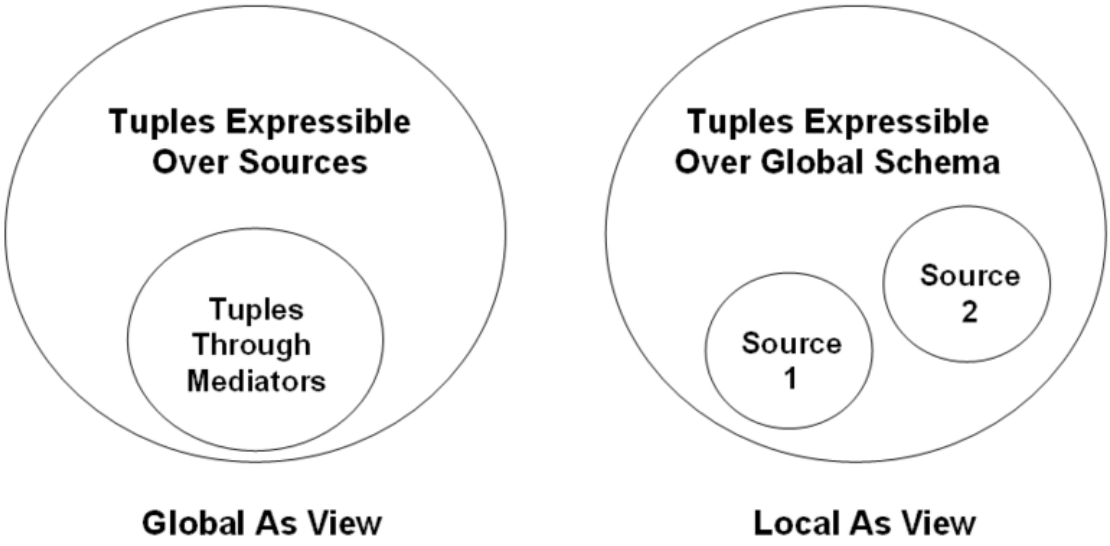


Figure 3: Illustration of tuple space of the GAV and LAV mappings. In GAV, the system is constrained to the set of tuples mapped by the mediators while the set of tuples expressible over the sources may be much larger and richer. In LAV, the system is constrained to the set of tuples in the sources while the set of tuples expressible over the global schema can be much larger. Therefore LAV systems must often deal with incomplete answers.

GAV systems model the global database as a set of views over *S*. In this case *M* associates to each element of *G* as a query over *S*. Query processing becomes a straightforward operation due to the well-defined associations between *G* and *S*. The burden of complexity falls on implementing mediator code instructing the data integration system exactly how to retrieve elements from the source databases. If any new sources join the system, considerable effort may be necessary to update the mediator, thus the GAV approach appears preferable when the sources seem unlikely to change.

In a GAV approach to the example data integration system above, the system designer would first develop mediators for each of the city information sources and then design the global schema around these mediators. For example, consider if one of the sources served a weather website. The designer would likely then add a corresponding element for weather to the global schema. Then the bulk of effort concentrates on writing the proper mediator code that will transform predicates on weather into a query over the weather website. This effort can become complex if some other source also relates to weather, because the designer may need to write code to properly combine the results from the two sources.

On the other hand, in LAV, the source database is modeled as a set of views over *G*. In this case *M* associates to each element of *S* a query over *G*. Here the exact associations between *G* and *S* are no longer well-defined. As is illustrated in the next section, the burden of determining how to retrieve elements from the sources is placed on the query processor. The benefit of an LAV modeling is that new sources can be added with far less

work than in a GAV system, thus the LAV approach should be favored in cases where the mediated schema is more stable and unlikely to change.

In an LAV approach to the example data integration system above, the system designer designs the global schema first and then simply inputs the schemas of the respective city information sources. Consider again if one of the sources serves a weather website. The designer would add corresponding elements for weather to the global schema only if none existed already. Then programmers write an adapter or wrapper for the website and add a schema description of the website's results to the source schemas. The complexity of adding the new source moves from the designer to the query processor.

Query processing

The theory of query processing in data integration systems is commonly expressed using conjunctive queries . One can loosely think of a conjunctive query as a logical function applied to the relations of a database such as "$f(A,B)$ where $A < B$". If a tuple or set of tuples is substituted into the rule and satisfies it (makes it true), then we consider that tuple as part of the set of answers in the query. While formal languages like Datalog express these queries concisely and without ambiguity, common SQL queries count as conjunctive queries as well.

In terms of data integration, "query containment" represents an important property of conjunctive queries. A query A contains another query B (denoted $A \supset B$) if the results of applying B are a subset of the results of applying A for any database. The two queries are said to be equivalent if the resulting sets are equal for any database. This is important because in both GAV and LAV systems, a user poses conjunctive queries over a *virtual* schema represented by a set of views, or "materialized" conjunctive queries. Integration seeks to rewrite the queries represented by the views to make their results equivalent or maximally contained by our user's query. This corresponds to the problem of answering queries using views (AQUV).

In GAV systems, a system designer writes mediator code to define the query-rewriting. Each element in the user's query corresponds to a substitution rule just as each element in the global schema corresponds to a query over the source. Query processing simply expands the subgoals of the user's query according to the rule specified in the mediator and thus the resulting query is likely to be equivalent. While the designer does the majority of the work beforehand, some GAV systems such as Tsimmis involve simplifying the mediator description process.

In LAV systems, queries undergo a more radical process of rewriting because no mediator exists to align the user's query with a simple expansion strategy. The integration system must execute a search over the space of possible queries in order to find the best rewrite. The resulting rewrite may not be an equivalent query but maximally contained, and the resulting tuples may be incomplete. As of 2009 the MiniCon algorithm is the leading query rewriting algorithm for LAV data integration systems.

In general, the complexity of query rewriting is NP-complete. If the space of rewrites is relatively small this does not pose a problem — even for integration systems with hundreds of sources.

Chapter 13

Enterprise Information Integration

Enterprise Information Integration (EII), is a process of information integration, using data abstraction to provide a single interface (known as uniform data access) for viewing all the data within an organization, and a single set of structures and naming conventions (known as uniform information representation) to represent this data; the goal of EII is to get a large set of heterogeneous data sources to appear to a user or system as a single, homogeneous data source.

Overview

Data within an enterprise can be stored in various formats, including relational databases (which themselves come in a large number of varieties), text files, XML files, spreadsheets and a variety of proprietary storage methods, each with their own indexing and data access methods.

Standardized data access APIs have emerged, that offer a specific set of commands to retrieve and modify data from a generic data source. Many applications exist that implement these APIs' commands across various data sources, most notably relational databases. Such APIs include ODBC, JDBC, OLE DB, and more recently ADO.NET.

There are also standard formats for representing data within a file, that are very important to information integration. The best-known of these is XML, which has emerged as a standard universal representation format. There are also more specific XML "grammars" defined for specific types of data, such as Geography Markup Language for expressing geographical features, and Directory Service Markup Language, for holding directory-style information. In addition, non-XML standard formats exist, such as iCalendar, for representing calendar information, and vCard, for business card information.

Enterprise Information Integration (EII) applies data integration commercially. Despite the theoretical problems described above, the private sector shows more concern with the

problems of data integration as a viable product. EII emphasizes neither on correctness nor tractability, but speed and simplicity. An EII industry has emerged, but many professionals believe it does not perform to its full potential. Practitioners cite the following major issues which EII must address for the industry to become mature:

simplicity of understanding
: Answering queries with views arouses interest from a theoretical standpoint, but difficulties in understanding how to incorporate it as an "enterprise solution". Some developers believe it should be merged with EAI. Others believe it should be incorporated with ETL systems, citing customers' confusion over the differences between the two services.

simplicity of deployment
: Even if recognized as a solution to a problem, EII as of 2009currently takes time to apply and offers complexities in deployment. People have proposed a variety of schema-less solutions such as "Lean Middleware", but ease-of-use and speed of employment appear inversely proportional to the generality of such systems. Others cite the need for standard data interfaces to speed and simplify the integration process in practice.

handling higher-order information
: Analysts experience difficulty — even with a functioning information integration system — in determining whether the sources in the database will satisfy a given application. Answering these kinds of questions about a set of repositories requires semantic information like metadata and/or ontologies. The few commercial tools that leverage this information remain in their infancy.

Applications

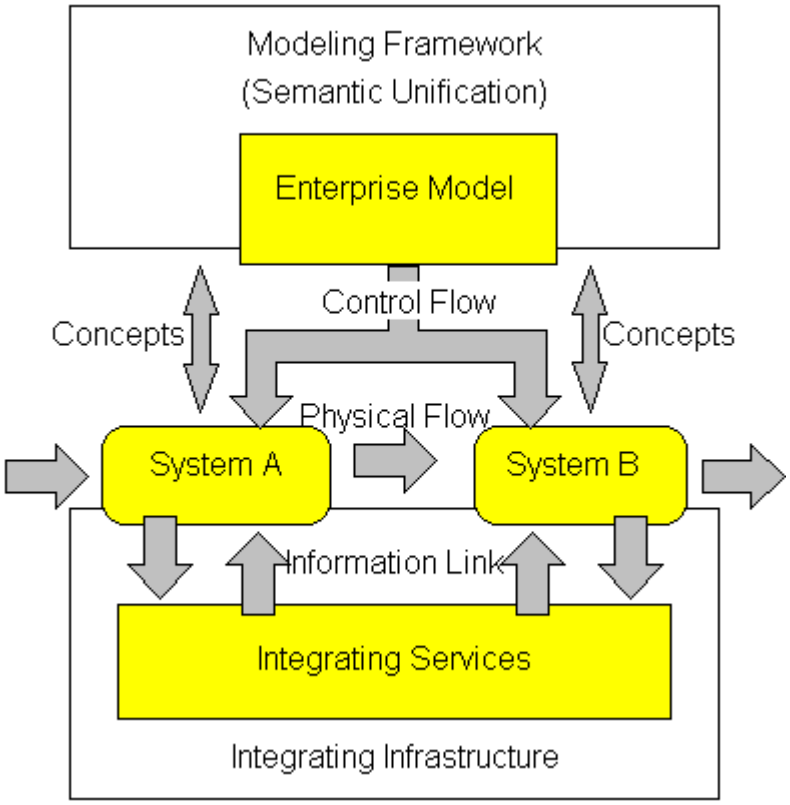
EII products enable loose coupling between homogeneous-data consuming client applications and services and heterogeneous-data stores. Such client applications and services include Desktop Productivity Tools (spreadsheets, word processors, presentation software, etc.), Development Environments and Frameworks (Java EE, .NET, Mono, SOAP or RESTian Web services, etc.), business intelligence (BI), business activity monitoring (BAM) software, enterprise resource planning (ERP), Customer Relationship Management (CRM), Business Process Management (BPM and/or BPEL) Software, and web content management (CMS).

Data access technologies

- ADO.NET
- JDBC
- ODBC
- OLE DB
- XQuery
- Service Data Objects (SDO) for Java, C++ and .Net clients and any type of data source

Chapter 14

Enterprise Integration



Concept of Enterprise Integration.

Enterprise integration is a technical field of Enterprise Architecture, which focused on the study of topics such as system interconnection, electronic data interchange, product data exchange and distributed computing environments.

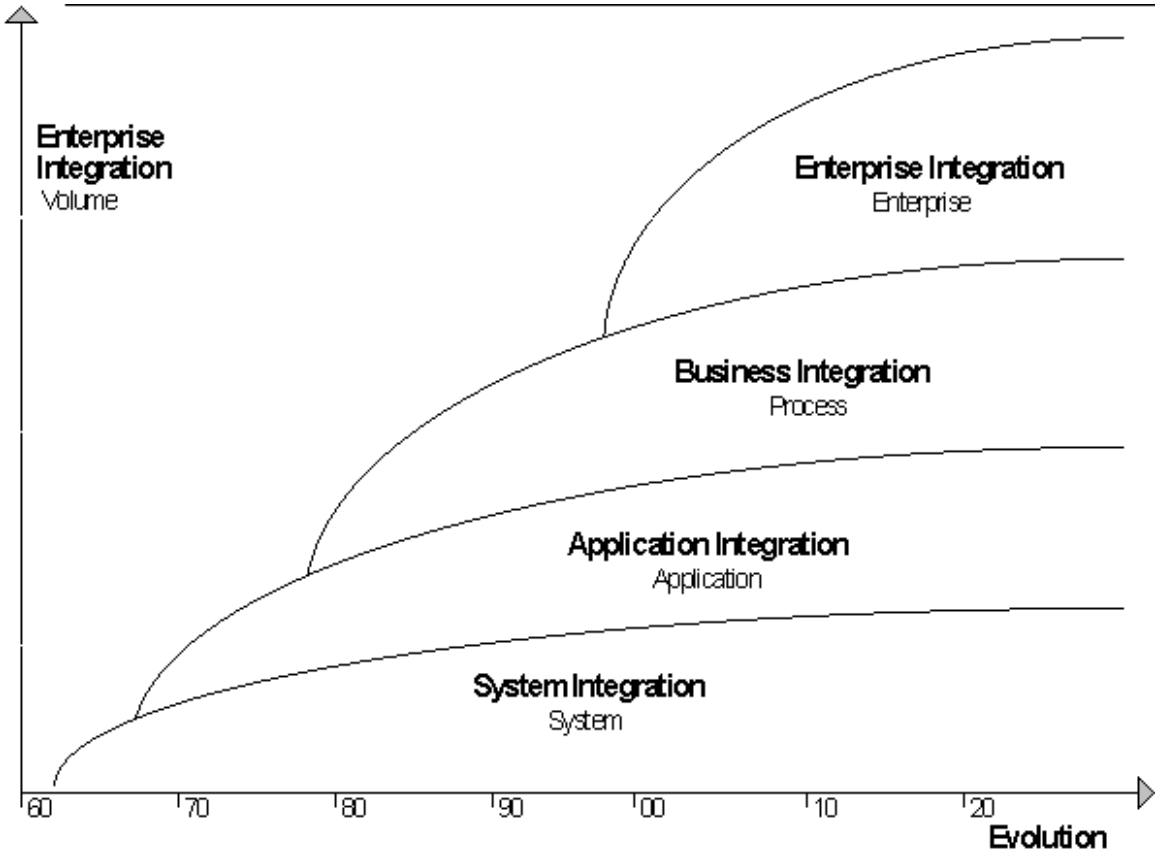
It is a concept in Enterprise engineering to provide the right information at the right place and at the right time and thereby enable communication between people, machines and computers and their efficient co-operation and co-ordination.

Overview

Enterprise Integration, according Brosey et al. (2001), "aims to connect and combines people, processes, systems, and technologies to ensure that the right people and the right processes have the right information and the right resources at the right time".

Enterprise Integration is focused on optimizing operations in a world which could be considered full of continuous and largely unpredictable change. Changes occur in single manufacturing companies just as well as in an "everchanging set of extended or virtual enterprises". It enables the actors to make "quick and accurate decisions and adaptation of operations to respond to emerging threats and opportunities".

History



Evolution in Enterprise Integration: This figure summarizes these developments indicating the shift of emphasis from systems integration to enterprise integration with increasing focus on inter enterprise operations or networks.

Enterprise integration has been discussed since the early days of computers in industry and especially in the manufacturing industry with Computer Integrated Manufacturing (CIM) as the acronym for operations integration. In spite of the different understandings of the scope of integration in CIM it has always stood for information integration across at least parts of the enterprise. Information integration essentially consists of providing the right information, at the right place, at the right time.

In the 1990s enterprise integration and enterprise engineering has become a focal point of discussions with active contribution of many disciplines. The state of the art in enterprise engineering and integration end 1990s, according to Jim Nell and Kurt Kosanke (1997), has been rather confusing. On one hand it claims to provide solutions for many of the issues identified in enterprise integration. On the other hand the solutions seem to compete with each other, use conflicting terminology and do not provide any clues on their relations to solutions on other issues. Workflow modelling, business process modelling, business process reengineering (BPR), and concurrent engineering all aim toward identifying and providing the information needed in the enterprise operation. In addition, numerous integrating-platforms concepts are promoted with only marginal or no recognition or support of information identification. Tools claiming to support enterprise modelling exist in very large numbers, but the support is rather marginal, especially if models are to be used by the end user, for instance, in decision support.

Enterprise integration topics

Enterprise modeling

In his 1996 book "Enterprise Modeling and Integration: Principles and Applications" François Vernadat states, that "enterprise modeling is concerned with assessing various aspects of an enterprise in order to better understand, restructure or design enterprise operations. It is the basis of business process reengineering and the first step to achieving enterprise integration. Enterprise integration according to Vernadat is a rapidly developing technical field which has already shown proven solutions for system interconnection, electronic data interchange, product data exchange and distributed computing environments. His book combines these two methodologies and advocates a systematic engineering approach called Enterprise Engineering, for modeling, analysing, designing and implementing integrated enterprise systems".

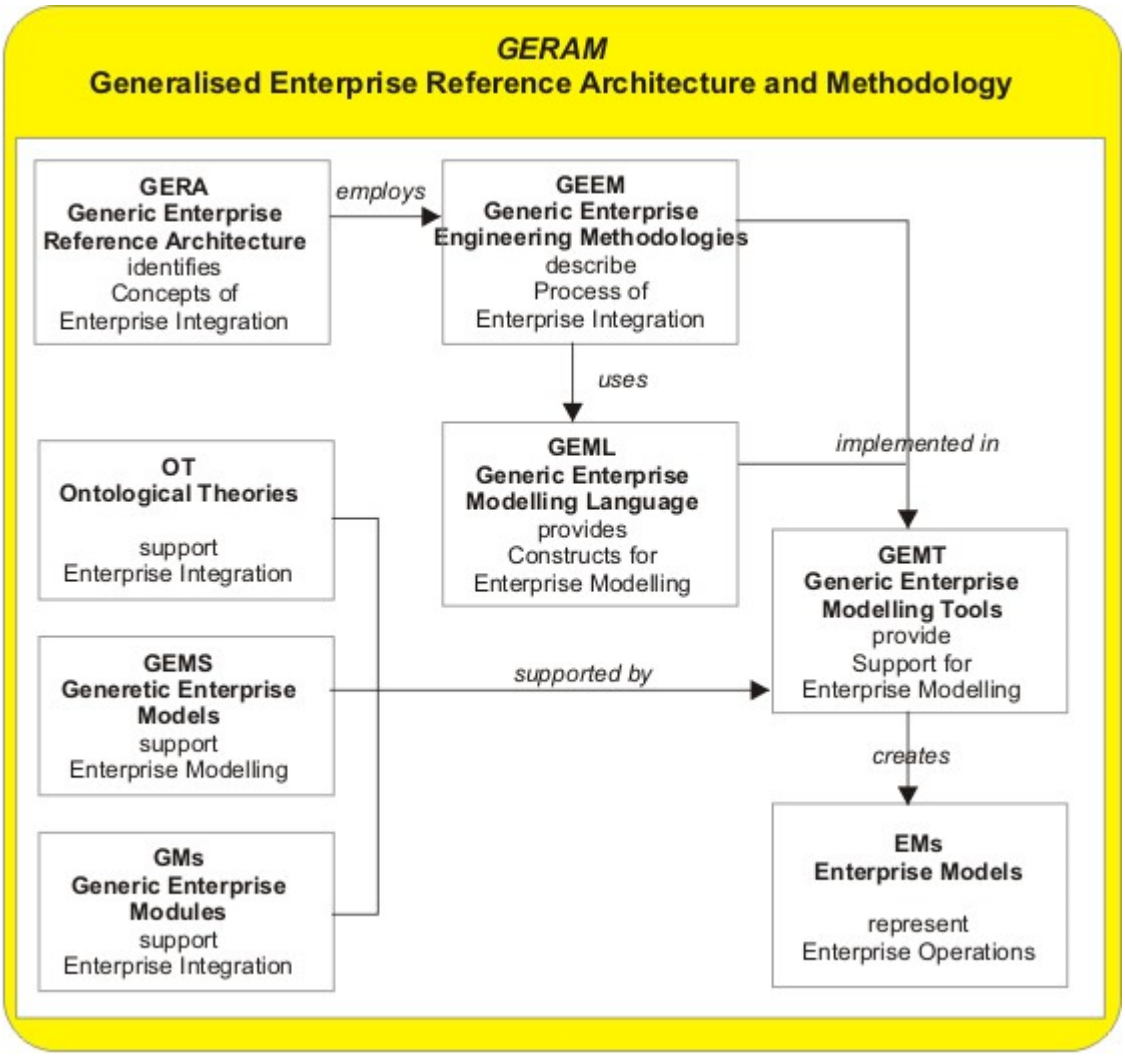
Enterprise integration needs

With this understanding the different needs in enterprise integration can be identified:

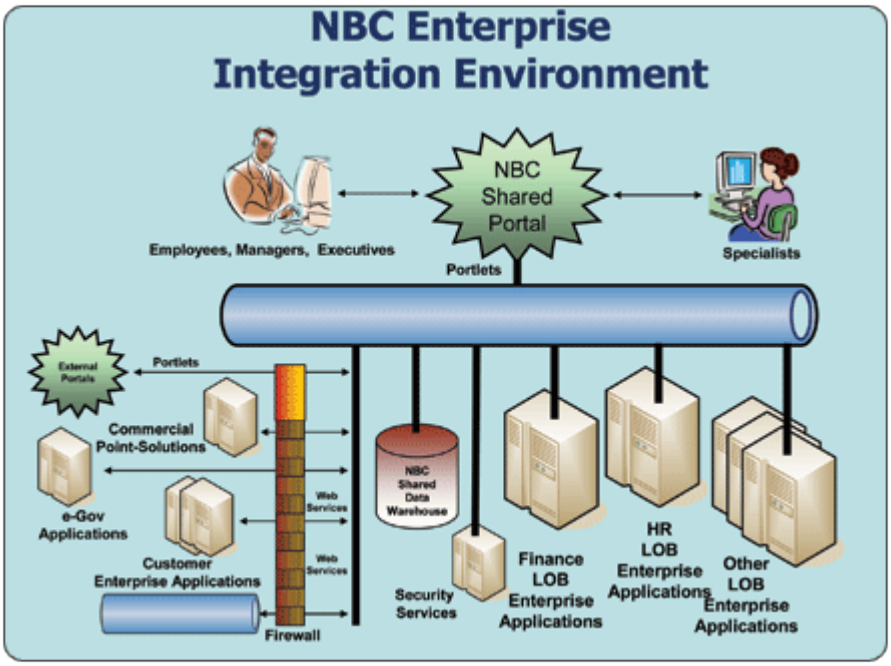
- *Identify the right information*: requires a precise knowledge of the information needed and created by the different activities in the enterprise operation. Knowledge which has to be structured in the form of an accurate model of the enterprise operation. A model which describes product and administrative information, resources and organisational aspects of the operational processes and allows what-if analysis in order to optimize these processes.

- *Provide the right information at the right place*: requires information sharing systems and integration platforms capable of handling information transaction across heterogeneous environments. Environments which consists of heterogeneous hardware, different operating systems and monolithic software applications (legacy systems). Environments which cross organizational boundaries and link the operation of different organisations on a temporal basis and with short set-up times and limited time horizon (extended and virtual enterprises).
- *Up-date the information in real time to reflect the actual state of the enterprise operation*: requires not only the up-date of the operational data (information created during the operation), but adapting to environmental changes as well. Changes which may originate from new customer demands, new technology, new legislation or new philosophies of the society at large. Changes which may require modification of the operational processes, the human organization or even the overall scope and goals of the enterprise.
- *Co-ordinate business processes*: requires precise modelling of the enterprise operation in terms of business processes, their relations with each other, with information, resources and organisation. This goes far beyond exchange of information and information sharing. It takes into account decisional capabilities and know-how within the enterprise for real time decision support and evaluation of operational alternatives.
- *Organize and adapt the enterprise*: requires very detailed and up-to-date knowledge of both the current state of the enterprise operation and its environment (market, technology, society). Knowledge which has to be available a priori and very well structured to allow easy identification of and access to relevant information.

Identification and use of information



Generalised Enterprise Reference Architecture and Methodology (GERAM) Framework for Enterprise Integration.



Example of Enterprise integration: the US National Business Center's Human Resources Line of Business, Innovative Future Direction.

Explicit knowledge on information needs during the operation of the enterprise can be provided by a model of the operational processes. A model which identifies the operational tasks, their required information supply and removal needs as well as the point in time of required information transactions. In order to enable consistent modelling of the enterprise operation the modelling process has to be guided and supported by a reference architecture, a methodology and IT based tools.

The Generalised Enterprise Reference Architecture and Methodology (GERAM) framework defined by the IFAC/IFIP Task Force provides the necessary guidance of the modelling process, see figure, and enables semantic unification of the model contents as well. The framework identifies the set of components necessary and helpful for enterprise modelling. The general concepts identified and defined in the reference architecture consist of life cycle, life history, model views among others. these concept help the user to create and maintain the process models of the operation and use them in her/his daily work. The modelling tools will support both model engineering and model use by providing an appropriate methodology and language for guiding the user and model representation, respectively.

Transfer of information

To enable an integrated real time support of the operation, both the process descriptions and the actual information have to be available in real time for decision support, operation monitoring and control, and model maintenance.

The figure illustrates the concept of an integrating infrastructure linking the enterprise model to the real world systems. Integrating services act as a harmonising platform across the heterogeneous system environments (IT and others) and provide the necessary execution support for the model. The process dynamics captured in the enterprise model act as the control flow for model enactment. Therefore access to information and its transfer to and from the location of use is controlled by the model and supported by the integrating infrastructure. The harmonising characteristics of the integrating infrastructure enables transfer of information across and beyond the organisation. Through the semantic unification of the modelling framework interoperability of enterprise models is assured as well.

Enterprise Integration Act of 2002

The Public Law 107-277 (116 Stat. 1936-1938), known as the Enterprise Integration Act of 2002, authorizes the National Institute of Standards and Technology to work with major manufacturing industries on an initiative of standards development and implementation for electronic enterprise integration, etc. It requires the Director of the National Institute of Standards and Technology (NIST) to establish an initiative for advancing enterprise integration within the United States which shall:

- involve the various units of NIST, including NIST laboratories, the Manufacturing Extension Partnership program, and the Malcolm Baldrige National Quality Program, and consortia that include government and industry;
- build upon ongoing efforts of NIST and the private sector; and
- address the enterprise integration needs of each major U.S. manufacturing industry at the earliest possible date.

Chapter 15

Integration Competency Center

The **Integration Competency Center** (ICC), sometimes referred to as an Integration Center of Expertise (COE), is a shared service function within an organization, particularly large corporate enterprises as well as public sector institutions, for performing methodical Data Integration , System Integration or Enterprise Application Integration.

Data integration allows companies to access their enterprise data and functions, fragmented across disparate systems, in order to create a combined, accurate, and consistent view of their core information as well as process assets and leverage them across the enterprise to drive business decisions and operations. System Integration is the bringing together of component subsystems into one system and ensuring that they function together effectively. Enterprise Application Integration enables efficient information exchanges and business process automation across separate computer applications in a cohesive fashion.

Overview

The term may be better understood by examining each of the three words that comprise the acronym. **Integration** refers to the objective of the ICC to take a holistic perspective and optimize certain qualities such as cost efficiency, organizational agility and effectiveness, operational risk, customer (internal) experience, etc. across multiple functional groups. **Competency** refers to the expertise, knowledge or capability that the ICC offers as services. **Center** means that the service is managed or coordinated from a common (central) point independent from the functional areas that it supports.

Large organizations are usually sub-divided into functional areas such as marketing, sales, distribution, finance, human resources to name just a few. These functional groups have separate operations and are *vertically integrated* and are therefore sometimes referred to as "silos" or "stovepipes". From an organizational perspective, an ICC is a

group of people with special skills, who are centrally coordinated, and offer services to accomplish a mission that requires separate functional areas to work together.

Key objectives of an ICC are:

- Lead and support enterprise integration (data, system and process) projects with the cooperation/coordination of subject matter experts
- Promote Enterprise integration as a formal discipline. For example, data integration will include data warehousing, data migration, data quality management, data integration for service oriented architecture deployments, and data synchronization. Similar system integration will include common messaging services, business service virtualization etc.
- Develop staff specialist in integration processes and operations and leverage their expertise company-wide
- Assess and select integration technology and tools from the market place
- Manage integration pilots and projects across the organization
- Optimize on the investment costs at enterprise level
- Leverage the economies of scale for integration portfolio at enterprise level

ICCs allow companies to:

- Optimize scarce resources by combining integration skills, resources, and processes into one group
- Reduce project delivery times and development and maintenance costs through effectiveness and efficiency
- Improve ROI through creation and reuse of enterprise assets like source definitions, application interfaces, and codified business rules
- Decrease duplication of integration related effort across the enterprise.
- Build on past successes instead of reinventing the wheel with each project
- Lower total technology cost of ownership by leveraging technology investments across multiple projects

An ICC may be a temporary group in support of a program or a permanent part of the organization. Furthermore, ICC's can be established at various scales or levels; within a division of a company, at the enterprise level, or across multiple companies in a supply chain.

History

The term Integration Competency Center and its acronym ICC was popularized by Roy Schulte of Gartner in a series of articles and conference presentations beginning in 2001 with ‘’’’The Integration Competency Center’’’’ [Ref SPA-14-0456]. He picked up the term from one of his colleagues, Gary Long, who found some of his clients using it (they took the established term “competency center” and applied it to integration). Prior to that (from 1997 to 2001) Gartner had been referring to it as the ‘’central integration team’’.

The concept itself (even before it was given a label) goes back to 1996 in one of Gartner's first reports on integration.

A major milestone was the publication in 2005 of the first book on the topic: ***Integration Competency Center: An Implementation Methodology*** by *John Schmidt* and *David Lyle*. The book introduced five ICC organizational models and explored the people, process and technology dimensions of ICC's. Several reviews of the book can be found at IT Toolbox and at Amazon. The concept of integration as a competency in the IT domain has now survived for over 10 years and appears to be picking up momentum and broad-based acceptance. The concept of ICC was further institutionalized by Infosys Technologies with the creation of a standard framework for establishing ICCs.

These days, ICC's are often called, Integration Center of Excellence, SOA Center of Excellence, the Data Management Center of Excellence and other variants. The most advanced ICC's are using Lean Integration practices to optimize end-to-end processes and to drive continuous improvements. Universities are also beginning to include integration topics in their MBA programs and Computer Science curricula. For example, The College of Information Sciences and Technology at Penn State University has established a Enterprise Informatics and Integration Center with the following mission:

"*The Enterprise Informatics and Integration Center (EI²) will actively engage industry, non-profit, and government agency leaders to address critical issues in enterprise processes, knowledge management, and decision making.*"

ICC Operating Models

There are a number of ways an ICC can be organized and a wide range of responsibilities with which it can be chartered. The ICC book introduced five ICC organizational models and explored the people, process and technology dimensions of ICCs. They include:

Best Practices ICC

The primary function of this ICC model is to document best practices. It does not include a central support or development team to implement those standards across projects, and probably not metadata either. To implement a Best Practices ICC, companies need a flexible development environment that supports diverse teams and that enables the team to enhance and extend existing systems and processes. Such a team might be a subset of an existing Enterprise Architecture capability and generally consists of a small number of staff (1-5).

Standard Services ICC

A Standard Services ICC provides the same knowledge leverage as a Best Practices ICC, but enforces technical consistency in software development and hardware choices. A Standard Services ICC focuses on processes, including standardizing and enforcing naming conventions, establishing metadata standards, instituting change management

procedures, and providing standards training. This type of ICC also reviews emerging technologies, selects vendors, and manages hardware and software systems. This style of ICC is often tightly linked with the Enterprise Architecture team and may be slightly larger than a typical Best Practices ICC.

Shared Services ICC

A Shared Services ICC provides a supported technical environment and services ranging from development support all the way through to a help desk for projects in production. This type of ICC is significantly more complex than a Best Practices or Standard Services model. It establishes processes for knowledge management, including product training, standards enforcement, technology benchmarking, and metadata management, and it facilitates impact analysis, software quality, and effective use of developer resources across projects. The organizational structure of a Shared Services ICC is sometimes referred to as a hybrid or federated model which often includes a small central coordinating team plus dotted-line reporting relationships with multiple distributed teams.

Central Services ICC

A Central Services ICC controls integration across the enterprise. It carries out the same processes as the other models, but in addition usually has its own budget and a charge-back methodology. It also offers more support for development projects, providing management, development resources, data profiling, data quality, and unit testing. Because a Central Services ICC is more involved in development activities than the other models, it requires a production operator and a data integration developer. The staff in a Central Services ICC does not necessarily need to be a central location and may be distributed geographically; the important distinction is that the staffs have a solid-line reporting relationship to the ICC Director. The size of these teams can vary and may be as large as 10%-15% of the IT staff in an organization.

Self Service ICC

The Self Service ICC represents the highest level of maturity in an organization. The ICC itself may be almost invisible in that its functions are so ingrained in the day-to-day systems development life-cycle and its operations are so tightly integrated with the infrastructure that it may require only small central team to sustain itself. This ICC model achieves both a highly efficient operation and provides an environment where independent development and innovation can flourish. This goal is achieved by strict enforcement of a set of application integration standards through automated processes enabled by tools and systems.

Key Challenges in ICC establishment

ICC as a concept is fairly simple. It is embodiment of the IT management best practices to deliver shared services. However, being an organizational concept, it is far more challenging to implement in practice than the conceptual view because every

organization has different DNA and it takes specific personalization/customization effort for ICC that makes the ICC initiative successful. Here are some of the common challenges in ICC establishment journey:

- Change management in terms of technology, processes, organization structure
- Ability of the organization to deal with the pace and quantum of change
- Alignment of stakeholders and process owners for ICC strategy
- Inappropriate ownership level for ICC program and lack of senior management sponsorship
- Highly tactical focus and business program level constraints
- Ignoring foundation elements and jumping to implementation directly
- Inappropriate funding

These issues are important to consider when embarking on the ICC investment since the last leg of the implementation of ICC that's what matters most. Intellectual definition of ICC that is not implemented in the organisation has no real value for the enterprise.