Application and Process Integration – Concepts, Issues, and Research Directions[1]

P. Johannesson
Department of Computer and Systems Science, Stockholm University/KTH
Stockholm, Sweden, email: pajo@dsv.su.se

B. Wangler
Department of Computer Science, University of Skövde
Skövde, Sweden, email: benkt@dsv.su.se

P. Jayaweera[2]
Department of Computer and Systems Science, Stockholm University/KTH
Stockholm, Sweden, email: prasad@dsv.su.se

Abstract

The need for integrating applications is growing as a consequence of organisational demands and enabling technologies, in particular the Web and enterprise software packages. In this Chapter, we introduce the basic concepts of application integration, discuss a number of the most important issues in the area and outline promising research directions: process libraries, methodologies for process integration, adaptive and flexible process enactment, and moving business logic from systems to processes.

1. Introduction

As organisations are becoming increasingly dependent on information technology, the need for integrating applications is growing. As an answer to this need, technologies in Enterprise Application Integration (EAI) have been proposed. EAI can be defined as "the unrestricted sharing of data and business processes among *any* connected applications and data sources in the enterprise", [1]. The demand for EAI is driven by many forces, where one of the most important is the move to process orientation in many organisations. Traditionally, organisations have been functionally divided, i.e. companies have been separated into departments such as marketing, sales, procurement, production, and service. However, such a functional

[1] This work was sponsored by NUTEK (Swedish National Board for Industrial and Technical Development) and SIDA (Swedish International Development Cooperation Agency)
[2] On leave from Institute of Computer Science, University of Ruhuna, Sri Lanka

organisation has been shown to have a number of weaknesses. In particular, it requires a huge administration to handle issues crossing functional borders. In order to overcome the problems of a functional organisation, companies have been concentrating on business processes, that is the connected activities that create value for the customers. These processes cross the internal borders of an organisation as well as the external borders to other organisations, thereby supporting supplier and customer relationship management, virtual enterprises, and extended supply chain management.

Supporting cross-functional and inter-organisational processes puts new demands on IT systems and applications. Traditionally, the applications have been built up around departments or functions in the companies. The result has been a "stovepipe like" relation between the functions and the applications, where every function in the company is supported by its own system or application (see Fig. 1), but where the applications work as "islands of automation" with limited communication among each other. This architecture is not satisfactory for process oriented organisations; to support the business processes in full the applications need to be integrated.

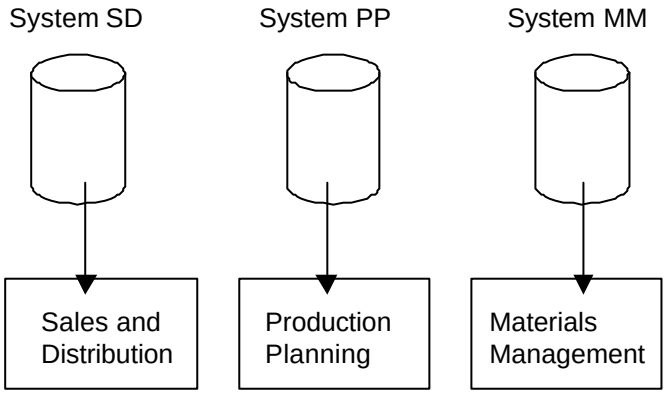

Fig. 1 IT systems supporting single functions

In addition to organisational forces, a number of enabling technologies drive the demand for EAI, in particular Internet and enterprise software packages. Internet provides an environment that can link a company's customers, suppliers, partners, and its internal users. Enterprise software packages offer an integrated environment for supporting business processes across the functional divisions in organisations. Some packages, like enterprise resource planning (ERP), for example SAP R/3 or Oracle Applications, manage back-office requirements, while other packages provide front-office capabilities, e.g. customer services. Common to Web applications as well as enterprise software packages is the need for application integration. Application integration is required to connect front office systems with back office systems, to transfer business processes to the Web, and to create extended supply chains involving customers, partners, and suppliers. Application

integration is also needed for wrapping legacy systems and for migrating to new environments.

The purpose of this Chapter is to introduce the basic concepts in EAI, to discuss a number of the most important issues in the area, and to point out promising research directions.

2. Technologies for EAI

Enterprise Application Integration can take place on different levels, which is a fact that has been recognised since a long time, [2] [3]. In [1], D. Linthicum identifies the following four levels:

Data level EAI. In data level integration, the integration takes place between data stores. More concretely, data is extracted from one database and used to update another database, possibly after appropriate modifications. A major advantage of this approach is that it does not require any modifications of the existing applications. Furthermore, the approach relies on inexpensive and established technology, i.e. database oriented middleware such as ODBC and JDBC.

Application interface level EAI. An application interface is an interface that gives access to services provided by a custom application or a standard package. It is possible to distinguish between three types of services, [1]: business, data, and object services. A business service provides access to some business logic, e.g. calculating prices or updating customer information. A data service provides a route to the logical or physical database and is in this similar to data-level access tools. An object service is the combination of business and data services packaged as an object. An advantage of objects is that integrity constraints cannot be violated as updates to data are always carried out by the appropriate methods of an object.

Method level EAI. In method level EAI, applications are integrated by being able to share a set of common methods. These methods can be stored on a central server, or they can reside as distributed objects in a network. Method level EAI is closely related to reuse. By introducing a number of common methods, different applications can reuse these methods.

User interface level EAI. User interface level EAI (also known as “screen scraping”) is the most primitive form of EAI. Applications are integrated through the user interfaces, i.e. information are accessed from user screens by programmatic mechanisms. This approach may seem quite unattractive, but in many cases it is the only one available for integration. Furthermore, it has the advantage of not requiring any changes to the applications to be integrated.

Integration of applications can be supported by many different architectures. One architecture for integrating applications is the point-to-point solution where every

application is directly connected to every other application, see Fig. 2. This solution could work for a small number of applications, but as the number of applications increases, the number of connections quickly becomes overwhelming. The Message Broker technology reduces this complexity, see Fig. 3. The main idea is to reduce the number of interfaces by introducing a central Message Broker and thereby make it easier to support the interfaces. If one of the applications changes format, only one connection has to be changed: the one to the Message Broker. The Process Broker, see Fig. 4, is an extension of the Message Broker. In addition to handling format conversions, the Process Broker also encapsulates the process logic for connecting applications. When all process logic resides in one place, it becomes possible to study, analyse, and change the processes using a graphical interface. This visualisation reduces the complexity and enables different categories of people to take part in the process design.

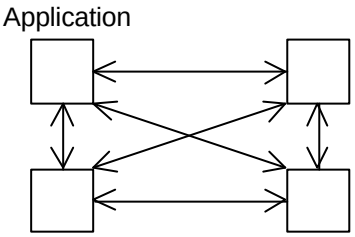

Fig. 2 Point to point integration

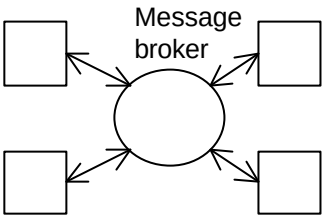

Fig. 3 Integration through a message broker

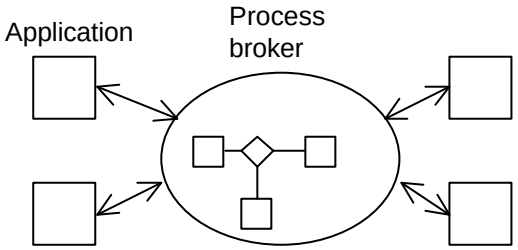

Fig.4 Integration through a process broker

The Process Broker technology can be seen as a continuation of workflow management systems. It is possible to categorise workflow systems into four generations starting from application specific, hard coded workflow capabilities realised as proprietary, closed systems. In the second generation, workflow capabilities factored out from the application domains to separate workflow applications with limited 3rd party application integration and tailorable process definitions via scripting languages. The current trends categorised into third generation can be distinguished as tailorable workflow services accessible to other

applications through APIs with open standards based architecture. Today proprietary workflow interfaces and interchange formats has the full interaction capability to 3rd party applications and tailorability to workflow definition via user friendly GUIs. In the next generation, workflow services will be fully embedded with other middleware services and towards standardised interfaces and interchange formats for workflow enabled applications where workflow capabilities are ubiquitous but invisible.

3. Research Directions

In this section, we identify a number of research directions within the area of application and process integration. Many of the problems and solutions are relevant also for other areas, but they take on an added significance when considered in the context of application integration.

3. 1 Process Libraries

A process library contains a large number of processes as well as support for navigation and customisation. A typical usage scenario is a business designer constructing a new process for sales. She could navigate the process library to identify a variety of alternative sales processes, e.g. sales by retail, mail order, or Internet. After having compared the alternatives, she can choose one or more processes, combine parts of them, and restructure the resulting process. Using a process broker, she can finally generate software for supporting the process and integrate with requested applications. Process libraries already exist as parts of commercial products, e.g. the Process Reference Model in SAP R/3 and the ARIS tool set, [4]. A research prototype of a large process library is the MIT Process Handbook, [5]. We believe that the most important research problems for process libraries are mechanisms for structuring the libraries in such a way that they become easy to browse and search. Below, we outline some promising structuring mechanisms.

Decomposition and Specialisation. A basic mechanism for structuring processes is decomposition. A process consists of subprocesses, which can be decomposed into other subprocesses and eventually into tasks. An example is shown in Fig. 5, where the "Sell product" process contains two subprocesses "Presales" and "Postsales", which are decomposed into tasks such as "Identify customer" and "Receive payment". In addition to being structured by decomposition, processes can also be organised in a specialisation hierarchy with the most general processes at the top and the most specialised processes at the bottom. As pointed out in [5], such a hierarchy will be similar to an object inheritance hierarchy in conventional object-oriented analysis and design. An object inheritance hierarchy consists of increasingly specialised objects that are associated with actions (i.e. methods). A process hierarchy, on the other hand, consists of increasingly specialised actions (i.e. processes and tasks) that are associated with objects. An example of a specialisation

of the process in Fig. 5 is given in Fig. 6. Process hierarchies provide two main benefits for a process library. First, they enable concise representations. When a new process is to be added, it can inherit large parts of its description from a more general process and only the differences to that process have to be explicitly stated. Secondly, hierarchies facilitate navigation in a process library: users can traverse the hierarchy upward in order to identify more general versions of a process and they can move across the hierarchy to find related processes.

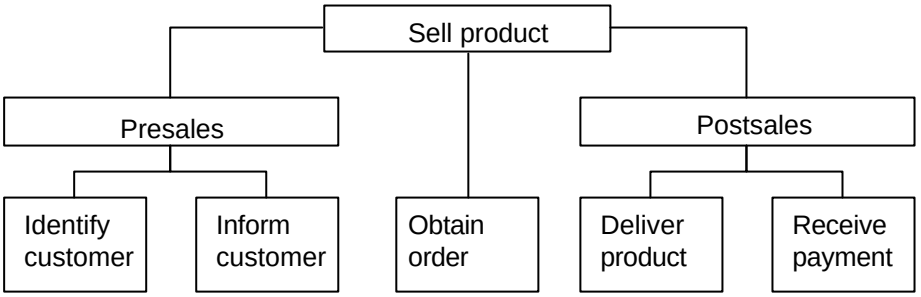

Fig. 5 A general sales process

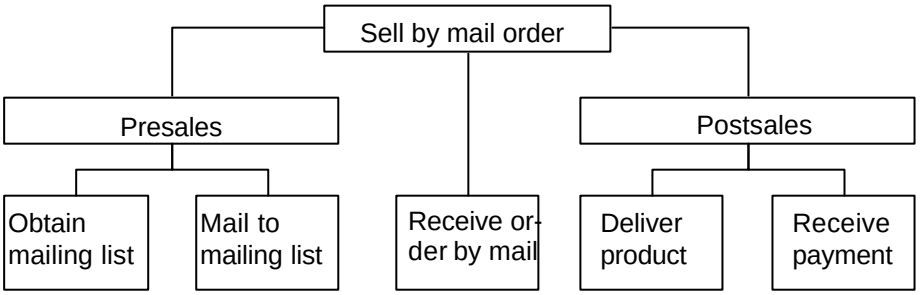

Fig. 6 A special sales process

Co-ordination. Another mechanism for structuring process libraries is based on co-ordination theory, [6]. Co-ordination is seen as the management of dependencies among activities, and three basic kinds of dependencies are identified: flow, sharing, and fit. *Flow dependencies* arise whenever one activity produces a resource that is used by another activity. *Sharing dependencies* occur when multiple activities all use the same resource. *Fit dependencies* arise when multiple activities together produce a single resource. Each dependency can be managed by different co-ordination processes. For example, a sharing dependency can be handled by co-ordination mechanisms such as first come/first serve, human decisions, bidding procedures, and priority orders. A flow dependency could be managed by, for example, a make to order or a make to inventory process. Dependencies and their possible co-ordination mechanisms can help structure a process library by allowing designers different views of the same process. A designer can switch back and forth between a dependency and the co-ordination mechanism by which it is handled. Furthermore, a designer can specify a particular dependency and browse through all the possible co-ordination mechanisms for that dependency.

Communication. Another structuring mechanism is based on a communicative language/action approach. This approach provides a way to structure the constituents of processes, see Fig. 7, [7]. At the bottom level, we find instrumental acts (e.g. producing goods or delivering products) as well as communicative acts (also called "speech acts"). The latter are used to make requests, commitments, and declarations through which obligations, permissions, and authorisations are established. The next layer contains the business transactions. A business transaction is the smallest sequence of acts that results in a new deontic state, e.g. a state in which an obligation to carry out some action has been created, or a state where an authorisation for certain actions has been established. A single speech act is in most cases not sufficient to achieve a deontic effect. In general, at least two messages are required. For example, a customer requesting a supplier to deliver a product and the supplier promising to do so; these two acts together constitute a business transaction. The top layer in Fig. 7 is the process level, where a process is built up by a number of business transactions. A language/action approach can help structuring a process library by providing a basis for decomposing processes into their constituents.

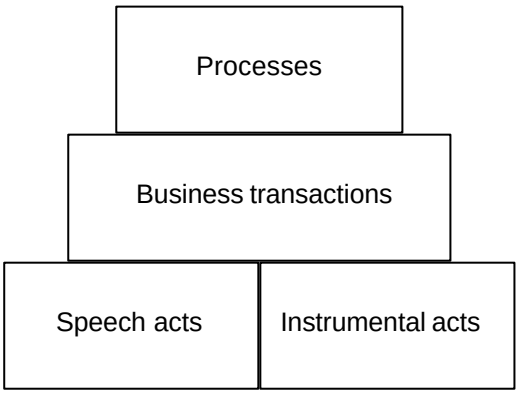

Fig. 7 Layers of actions

3. 2 Methodology

Effective EAI requires a comprehensive methodology including identification of data, processes, and application interfaces, performance considerations, process design, maintenance and much more. Three of the most important activities that would benefit from better methodological support are understanding the enterprise data, understanding the enterprise processes, and designing processes.

Understanding the enterprise data. Understanding data is obviously required for EAI at the data level, but an understanding of databases is also needed for EAI at the other levels. Understanding data starts by identifying the data by cataloguing which databases exist, who owns these databases, their physical location, data formats, etc. When data has been identified, an enterprise model of the data is to be built. This model will contain not only traditional data dictionary information, but also

additional information such as security information, connected processes, communication mechanisms, and integrity issues, [1]. Research on reverse engineering of databases will be relevant for this activity, especially when extended to the environment of the databases and not only their schemas.

Understanding the enterprise processes. Understanding processes also begins by identifying the processes in the enterprise. When the processes have been identified, a business model is constructed that describes the processes, their owners, the databases they interact with, and the technologies in which they are implemented. General process libraries, as introduced above, can be used as a means for organising the processes of a particular enterprise. Their navigation mechanisms can also be used for searching among the processes.

Designing processes. Application integration often results in highly unstructured and complex process models. One reason for this is that exception handling makes up a large part of an application integration specification and thereby easily obscures the main business logic. Furthermore, there is often extensive communication between a process broker and different applications, which also tends to conceal the business logic. Another characteristic of a process broker is that it does not maintain control over external applications, which means that these applications can be updated without the process broker being notified. As a consequence, it is often desirable to maintain redundant information that duplicates parts of the information in the external applications. This duplication of information requires some mechanisms for handling possible inconsistencies, which makes the process model even more complex. In order to overcome these problems of process modelling in the context of EAI, adequate design support is needed. One approach, as suggested in [8], is to model a process through a series of views starting with a customer oriented view on the business level, which models the interactions between the process broker and the customer. The succeeding views add more and more details moving from a business perspective to a technical perspective by adding interactions with external applications, exception handling, etc. Each view is an extension of the previous one, either through adding subprocesses or through introducing new components into the existing diagrams. This approach gives the designer increased control by allowing her to focus on different aspects of a process in different views.

3.3 Adaptive Process Management

Current software systems for supporting process management, such as process brokers and workflow management systems, are effective for predictable and repetitive processes. However, they are typically unable to adapt to a dynamic environment where unexpected situations have to be managed. It is possible to distinguish among four different types of exceptions that can occur during the execution of a process, [9]: *basic failures*, which are failures of the software system or its environment; *application failures*, which are failures of the applications that are invoked by the software system; *expected exceptions*, which are predictable

deviations from the normal flow of the process; *unexpected exceptions*, which are mismatches between the actual execution of a process and its definition in the software system.

Basic failures and application failures are usually handled by the underlying layers of the process management system, e.g. by a database management system that supports recovery. Expected exceptions and unexpected exceptions, on the other hand, are special for process management software and require their own techniques and methodologies, [10]. An expected exception is directly related to the process domain and can therefore be modelled as a part of the process, even if it represents a deviation from the normal, or desired, course of events. Expected exceptions can be caused by temporal events (e.g. the expiration of a deadline) or by actions by external agents (e.g. a customer cancels her reservation). Typically, these events take place asynchronously with the tasks in the process, which means that they are not easily modelled by a graph of tasks, which is the most common way of specifying processes. Instead, a more promising approach for representing expected exceptions is to use ECA (Event-Condition-Action) rules, as suggested by [10]. The event part in the rules specifies the occurrence of a possibly exceptional situation, the condition part checks that an exception has really taken place, and the event part specifies the response to the exception. An unexpected exception is caused by a change in the process that could not have been anticipated at design time, e.g. a new government regulation. Unexpected exceptions can be handled in two ways. First, a running process instance that encounters the exception can be modified. This means that the process definition is left unchanged, and other future process instances will execute according to that definition. Secondly, the process definition can be modified so that also future process instances will execute according to the new definition. The latter alternative gives rise to research issues in determining when a running instance can be migrated to a new process definition.

3.4 Moving Application Logic Out of Systems and into Processes

Integration between applications concerns not just the transfer of data, but also the business logic that controls the sequencing and ensures the integrity of the business transactions. Each step may have to complete before the next can commence. Data entered by people and systems communicating with the business process have to fulfil correctness criteria, usually expressed through (business) rules.

In object-oriented analysis and design, e.g. UML, rules are often captured during the definition of use cases, [11]. Later they may be represented e.g. as cardinality constraints or generalisation relationships of class diagrams, or as triggering event rules or guard conditions of state transition diagrams.

It is not obvious where the business rules that together determine the business logic should be coded. Are they part of a new application, part of existing applications, or should they be kept in a separate integration layer? In object orientation, it is

sometimes claimed that rules belong to the objects they control. However, many rules concern several objects. Hence, it is not obvious to which objects these rules should be assigned. In TEMPORA [12], a project aiming to develop a systems development platform based on the explicit representation of business rules, this was found to be a serious methodological problem.

If a business rule is embedded in the code of an application, irrespective of whether it is existing or new, any time that rule changes, the application must be carefully modified, tested and then re-deployed. In business process oriented integration, it is often found that the functional behaviour of the systems is required to be integrated, not just the data.. Therefore business logic should, as much as possible, be kept outside of applications. The logic that is specific to a certain subtask is kept within the component that implements that task and the activity is associated with an application definition, in which it only needs to specify the work to be done and the resources required by a client application. Service combinations are implemented through logic that is specific for each combination and hence kept in the service (combination) definition. When the application changes, or is replaced, the only alteration necessary is to change the definition to point to another application.

Moreover, customers should be able themselves to specify which service combination they want. This calls for the ability to augment the service definition with guard conditions that hinder the use of services that customers are not authorised to use or that are not possible to combine with other services selected.

For example, in the ordering process it should be possible to specify which combination of services as regards e.g. delivery and payment methods one wishes. Constraints that guard the gate (guard conditions) to task components should be put in the process definition.

The fact that many companies move to using standardized enterprise resource planning systems and other COTS products tend to make them more alike each other, hence removing individual differences that may represent competitive advantage. Moving logic out of the applications and into the process definitions is then becoming a means for companies to retain their uniqueness.

4 Concluding Remarks

In this Chapter, we have discussed the rationale for application and process integration as well as a number of possible research directions in the area. In the marketplace, a number of middleware vendors are adding process modelling and simulation capabilities to their products, thereby moving into the Process Broker market. Some of the major products in this market are: Viewlocity's Business Integration Modeler and Manager [13], Extricity Software's AllianceSeries [14], Vitria Technology's BusinessWare [15], and HP's Changeengine [16]. We believe that research as outlined in this Chapter will benefit the future development and use

of products in this category. Within the NUTEK (Swedish National Board for Industrial and Technical Development) sponsored project ProcessBroker [http://www.dsv.su.se/~pajo/arrange/index.html], we will pursue the research directions discussed here.

Acknowledgements

The authors are grateful to other participants of this project, especially Birger Andersson, S.J. Paheerathan, Erik Perjons, and Nasrin Shakeri at KTH and Christer Wåhlander at Viewlocity for contributing ideas and for commenting on earlier versions of this paper.

References

[1] Linthicum D. Enterprise Application Integration. Addison-Wesley, 2000

[2] Bubenko J A Jr, Wangler B. Research Directions in Conceptual Specification Development. Conceptual Modeling, Databases and CASE: An Integrated View of Information Systems Development. Eds. Loucopoulos P, Zicari R. John Wiley & Sons, Ltd. 1992

[3] Ahlsén M, Bubenko J A Jr. Interoperability in Federated Information Systems. Second International Workshop on Intelligent and Cooperative Information Systems: Core Technology For Next Generation Information Systems, Ed. Brodie M L, Ceri S. Villa Olmo, Como, Italy, Dipartimento di Ellettronica, Politecnico di Milano, 1991

[4] Curran T, Ladd A. SAP R/3 Business Blueprint. Prentice Hall. 2000

[5] Bernstein A, Klein M, Malone, T. The Process Recombinator: A Tool for Generating New Business Process Ideas. In: International Conference on Information Systems, 1999.

[6] Malone T, Crowston K. The Interdisciplinary Study of Coordination. ACM Computing Surveys 1994; 26

[7] Papazoglou M. Distributed, Interoperable Workflow Support for Electronic Commerce. Lecture Notes in Computer Science, volume 1402, 1998

[8] Johannesson P, Perjons E. Design Principles for Application Integration. In: International Conference on Advanced Information Systems. Springer. 2000

[9] Eder J, Liebhart W. The Workflow Activity Model WAMO. In: 3rd International Conference on Cooperative Information Systems. University of Toronto Press. 1995, pp. 87-98

[10] Casati F, Fugini M, Mirbel I. An Environment for Designing Exceptions in Workflows. Information Systems 1999; 24: 255-273

[11] Allen P, Frost S. Component-Based Development for Enterprise Systems. Addison-Wesley, 1998

[12] Loucopoulos P, McBrien P, Schumacker F, Theodoulidis B, Kopanas V, Wangler B. Integrating database technology, rule-based systems and temporal reasoning for effective information systems: the TEMPORA paradigm. Journal of Information Systems 1991, 129-152.

[13] Wåhlander C, Nilsson M, Skoog A. Introduction to Business Integration Model (BIM). Copyright Viewlocity, 1998

[14] Extricty AllianceSeries, Extricity Software, http://www.extricity.com/products/alli_series_over.html, 2000-02-21

[15] Atwood R. Bringing Process Automation to Bear on the Task of Business Integration. Vitria Technology 1999 http://www.vitria.com/products/whitepapers/seyboldwp.html, 1999-11-25

[16] HP Changengine Overview, Hewlett Packard Company, http://www.ice.hp.com/cyc/af/00/101-0110.dir/aovm.pdf, 1999-10-04