MANAGING INFORMATION FLOW DYNAMICS WITH AGILE ENTERPRISE ARCHITECTURES

A. Alexopoulou, P. Kanellis, D. Martakos
Informatics and Telecommunications Department
National and Kapodistrian University of Athens
University Campus, Athens Greece - 157 84

Keywords: Enterprise systems, integration, information flow, flexibility

Abstract: New organization forms and ways of conducting business require architectures for enterprise systems that can support and not hinder entrepreneurial activities. Primarily this means that the information flow between both internal as well as cross-enterprise processes must be managed by underlying systems that offer a high level of automation as well as being highly flexible and integrated. In this respect, we present an agile architecture that offers a coherent and high level conceptualisation of the above properties that enterprise information systems should display, consider a number of technologies as potential implementation candidates and demonstrate how the architecture addresses node density, velocity, viscosity and volatility as parameters for managing and controlling the dynamics of information flows.

1 INTRODUCTION

The Internet has established itself as the medium through which companies channel the largest percentage of new entrepreneurial activities. As new opportunities are being sought through this medium, new ways of doing business are being defined and in this context the value of temporary and ephemeral commercial relationships is being assessed. Today, enabled via the utilization of the new technologies, companies in electronic marketplaces are able to form partnerships only for the duration of the transaction, as opposed to long-term hierarchical supply chain collaborations of yesteryear (Hammer, 2001; Yang and Papazoglou, 2000). “On demand” partner relationships can be formed with enterprises that have published their profiles on the web and best satisfy ones’ own requirements, regarding price, quality standards, delivery schedules and other attributes. As firms continuously sense opportunities for competitive action in their product-market spaces, it is agility which underlies firms’ success in continuously enhancing and redefining their value creation (Sambamurthy et al., 2003). It follows that agile enterprise infrastructures that can meet the performance criteria in terms of efficiency required for the execution of both inter and intra-organizational processes are a prerequisite. By ‘efficiency’ we mean smooth business process execution that does not suffer from delays or errors and, ease in altering the business logic of a process and adjusting it to the needs of the moment. In turn, both of the above need seamless integration of internal enterprise processes (private processes) with external ones (public processes).

According to Krovi et al (2003), to attain such levels of performance, it is imperative to enable and manage agility in terms of the flow of information in an organization. These parameters that affect directly the information flow, namely node density, velocity, viscosity and volatility should be taken into consideration when designing enterprise system architectures.

The purpose of this paper is to present, primarily at a conceptual level, such an enterprise systems architecture that offers the required flexibility whilst enabling full automation and high integration. Its design caters for the criteria set by the information flow parameters as defined by Krovi et al. (2003) and, is based on a Business Process Engine (BPE)
that acts as a coordinator for end-to-end processes. The term ‘end-to-end’ is used in a holistic manner to denote processes that comprise both internal enterprise activities, as well as external Business-to-Business (B2B) transactions between one or more trading partners. The paper proceeds as follows. In the next section we provide a brief discussion on information flow parameters whilst in the section that follows we present the architecture. In section 4, we propose a number of enabling technologies as possible candidates for the implementation of the architecture. In section 5 we demonstrate how flexibility, automation and integration as provided by the architecture can help in the management of the enterprise information flow in terms of node density, velocity, viscosity and volatility. Section 6 offers our conclusions.

2 INFORMATION FLOW PARAMETERS

The brief discussion in this section is based on the work of Krovi et al. (2003) where the reader is referred for a detailed description and explanation of the parameters affecting information flow dynamics.

Node density as an information flow parameter is determined by the number of intermediate nodes in the information processing channel, where a node is used to describe an entity or a group of entities capable of altering the properties of information flow. The number of intermediate nodes appears to be an important factor for two reasons. Firstly, if decision making at each node depends on information from other nodes, then the presence of a large number of nodes along the processing channel should result in an increase in uncertainty. Secondly, a large number of nodes may impede the speed of information flow. If the extreme case of manual processing (human node) is assumed, then an increase in the number of intermediate nodes would also negatively affect the processing efficiency of the entire system. Thus, architectures for enterprise systems should be agile, allowing organizations to manage internal and external flows by altering the number of intermediate nodes.

Velocity refers to the speed of incoming information at a node. It is inferred that architectures which are designed to facilitate fully and not partially the automation of information exchange, help to streamline the organisational processes and thus increase efficiency in this respect.

Viscosity refers to the degree of conflict that can be observed at a node. The conflict arises due to the presence of contradictory information components known as information particles – the smallest component of information that can exist independently and still retain the characteristics of information. In such cases, viscosity appears in the form of multiple values of information that must be resolved before the node can begin processing. If there is lesser conflict between the multiple values, then a quicker resolution can occur – a situation characterised by low viscosity. However, a high degree of conflict will likely delay the resolution time – a situation characterised by high viscosity. It is inferred that a tightly integrated infrastructure can help to eliminate, for example, various functional information silos and thus eliminate the presence at a node of contradictory information components being present at any one time.

The associated uncertainty in information content, format and/or timing is expressed by the value of volatility. External forces having their roots in industry or economy-wide factors can impact the degree of volatility creating in terms of transaction volume either laminar or turbulent information flows. Knowledge of relationships between external forces and internal processes can help manage the effect on the system which in any case should be flexible to the extent that it can accommodate information flow variances without affecting negatively efficiency levels.

3 AN AGILE ARCHITECTURE FOR ENTERPRISE SYSTEMS

In this section we present an architecture for enterprise information systems that enables flexibility, full automation and high integration. Flexibility, in general, means the ability to make changes easily, i.e. in a timely and cost-effective manner. Full automation means that the flow of information between activities, processes and nodes is carried out electronically with no manual intervention. Integration is the process whose ultimate aim is to create an infrastructure where different entities (applications, databases, etc.) can communicate efficiently with each other. Integration, as well as flexibility, can be approached at three different levels: business processes, data and application components. Integration at business process level means that business processes can span multiple applications, whether these applications belong to a single or to different companies. The latter case is about the integration of private with public processes. Integration at data level means that data reside in any data source anywhere and can be used by any application or system anywhere.
Integration of application components means that components can communicate efficiently with each other as well as with legacy applications. A system that is integrated at all three levels is a highly integrated system. This 3-level approach to integration and flexibility is depicted in Table 1. Flexibility at business process level means that a business process definition can be altered without requiring modification of the application components. A process definition specifies activities, roles, routes and rules. Roles refer to the users or the systems that are responsible for carrying out the activities within a business process. Routes specify the way activities can be combined to form a business process and rules define alternative paths of activities based on certain conditions. Flexibility at the data level denotes the efficient transformation of data from one format to another that can be realised at run-time. Finally, at application components level, flexibility means that new components can be easily embodied into the existing architecture and also that components can be re-used across multiple business scenarios.

Integration of application components means that components can communicate efficiently with each other as well as with different applications. A system that is integrated at all three levels is a highly integrated system. This 3-level approach to integration and flexibility is depicted in Table 1. Flexibility at business process level means that a business process definition can be altered without requiring modification of the application components. A process definition specifies activities, roles, routes and rules. Roles refer to the users or the systems that are responsible for carrying out the activities within a business process. Routes specify the way activities can be combined to form a business process and rules define alternative paths of activities based on certain conditions. Flexibility at the data level denotes the efficient transformation of data from one format to another that can be realised at run-time. Finally, at application components level, flexibility means that new components can be easily embodied into the existing architecture and also that components can be re-used across multiple business scenarios.

Based on the above and the discussion on information flow parameters in the previous section, we derive that a flexible architecture can satisfy performance criteria associated with node density and volatility, while a fully automated and integrated system can satisfy criteria associated with velocity and viscosity. The former stands because with a flexible infrastructure, alterations in the number of nodes within a business process can be performed fast and easily. Similarly, any operational changes imposed by external factors can be accommodated in a timely and cost effective manner. Automation and integration on the other hand, mean that information is not error-prone, keeping thus the value of viscosity low at nodes. In terms of velocity, it means the ability to accommodate variances in the flow of information without bottlenecks.

Our proposed architecture is presented in figure 1. At the heart of the architecture is the Business Process Engine (BPE), which interacts through the exchange of messages with (a) users via a Document-based Worklist Browser, (b) customers via a Web Browser, (c) trading partners via the B2B engine, and (d) applications and components via the Component Management Service (CMS).

The BPE acts as a coordinator of activities spanning across the enterprise entities (users, applications, trading partners, etc.) invoking for each activity the entity that is responsible for performing it. The BPE reads and executes business logic defined in process definition documents. This implies that the process definition is expressed in a business process definition language that is machine-readable.

The role of the document-based worklist browser is to inform the user about the tasks that need to be accomplished within the context and sequence of a specific business process. In general, it provides a graphical user interface that helps the user with his everyday tasks.

The B2B engine is responsible for handling communication (transport, security, etc.) with trading partners and other external entities (financial institutions, insurance agencies, etc.) through the implementation of any open B2B protocol (previous version). A typical architecture of B2B engines, together with a description of its constituent elements is presented in Bussler (2002).

Finally, the CMS finds and invokes the

<table>
<thead>
<tr>
<th>Business Processes</th>
<th>Integration</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business processes span multiple applications whether these applications belong to a single or to different companies.</td>
<td>A business process definition can be altered without requiring modification of the application components.</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>Data reside in any data source anywhere and can be used by any application or system anywhere.</td>
<td>Data can be easily transformed from one format to another at run time.</td>
</tr>
<tr>
<td>Application Components</td>
<td>Components can communicate efficiently with each other as well as with legacy applications.</td>
<td>New components can be easily embodied into the existing architecture and also components can be re-used across multiple business scenarios.</td>
</tr>
</tbody>
</table>

Table 1: 3-level definition of integration and flexibility
appropriate application components that deliver the requested business service. The components can intercommunicate over a common communication infrastructure. Legacy applications can be connected to the communication infrastructure via adapters. In essence, the CMS together with this infrastructure constitute an Enterprise Application Integration (EAI) implementation that follows some of the principles of the NGOSS (New Generation Operations Systems and Software) framework (TMF, 2001). NGOSS is an initiative of the TeleManagement Forum set to develop a framework for rapid and flexible integration of operations and business support systems in telecommunications, but it can be equally applied to other business areas as well. NGOSS defines a service-oriented system framework, which is based on a collection of loosely coupled, re-usable components that perform business services. According to the framework, all software entities that provide services to other entities do so through a contract-defined interface. This interface is a precise specification of the service, i.e., a description of the service to be provided, the semantics of the interface, etc. Contract-defined interfaces are divided into two types. The first is business service contracts, which are used to perform activities in support of the automation of business processes. The second is
framework service contracts, which are used to perform ancillary services that are used by the software entities to carry out their operations (e.g. a registry service that records information used during system execution, such as the location and state of all available software services).

Some other primary principles of the NGOSS framework, apart from the common communication infrastructure and the contract-defined interfaces, are the separation of business process flow from software implementation (which is achieved in our architecture through the use of the BPE and the executable business process specification documents), a shared information environment and a registry of run time information.

Finally, we should mention that whenever messages sent by the BPE need to be transformed into another format, a transformation mechanism is used. For example, if a message is to be directed to a worklist browser, it must be first transformed into HTML. Likewise, at the application components level, if for example Common Object Request Broker Architecture (CORBA) is used, then the messages sent by the BPE will have to be transformed into CORBA IDL messages. Overall, the B2B engine will have to transform them into the format required by the protocol used in the specific business collaboration.

Based on the above description, it becomes clear how the architecture enables full automation; business processes are described in a machine-readable document, which is executed by the BPE. The machine-readable document is generated at build time through a process definition tool. Of course, there are cases at this level that human intervention may be inevitable, hence the inclusion of the document-based worklist browser as an element of the architecture.

Integration at business process level is attained because of the fact that the BPE operates with users, applications and trading partners in a transparent manner. This is feasible due to the existence of the CMS that is responsible for invoking the necessary applications for the accomplishment of a business process, as well as the existence of the B2B engine that hides the communication and B2B protocol details from the BPE. As a result, the BPE can efficiently execute end-to-end processes, since the B2B transactions are integrated with the internal enterprise activities. Also, regarding integration at the application component level, it is reminded that this is addressed in the architecture by an EAI implementation based on the principles of the NGOSS framework. As far as data integration is concerned, this is achieved through the transformation mechanism described above.

Flexibility at process level ensues from the abstraction of the business process flow into an entity separate from the components themselves. As a result, business process steps can be easily rearranged or altered, since the only action required is to update the process definition document. The underlying component interactions will be automatically reconfigured. As far as data format transformations are concerned, the transformation mechanism we mention in the next section enables also run time transformations. Finally, in a NGOSS infrastructure, new components can be easily embodied into the infrastructure and communicate with the already existing applications via the common communication vehicle. Due to this abstraction, components can also be re-used across multiple business scenarios.

We will end this section by presenting an example of a specific business process in order to illustrate how the BPE works. For this example, we will assume a company that produces custom-made furniture. Suppose that whenever an order is received from a customer it is first checked and if accepted a new order is created for the acquisition of the wood that is necessary for the construction of the furniture ordered. The order is then communicated to the wood supplier.

This example is depicted in figure 2. As shown, the BPE executes at run time the process definition generated at build time and forwards the appropriate messages to the relevant users or systems. More specifically, the first step of the business process is the “Receive Order”, which is triggered by a “Send Order” message received from a customer. This order must first be examined by an authorized employee, so the BPE forwards the customer’s order to this person. As a result, a new task is added to his task list. After the acceptance of the order the next step is the creation of a new order for the acquisition of the raw materials. For this the BPE sends a relevant message to the CMS to request the specific business service. The CMS in turn invokes the appropriate components for the order creation. After the order is created, the BPE forwards it to the B2B engine, which is responsible for sending it to the wood supplier. In figure 2, it is also illustrated that transformations take place during the execution of this process. In particular, at the worklist browser, the business process definition messages are transformed into HTML format. At the application components level, they are transformed into CORBA IDL messages because we assume that the company uses this particular middleware technology. Finally, the B2B engine transforms them according to the B2B protocol (e.g. RosettaNet/www.rosettanet.org) used for this specific business collaboration with the wood supplier.
4 ENABLING TECHNOLOGIES

The key enabling technology for the architecture presented in the previous section is the eXtensible Markup Language (XML) (Bray et al., 2000). XML is a meta-language that offers a mechanism to represent data in an application-independent way. It provides a new perspective on information modeling, which enables the modeling of the data flow of a business process instance. Also it facilitates data exchange between systems within an enterprise, as well as between systems of different trading partners. Therefore, XML can help (a) to automate the execution of business processes, and (b) to form the foundation for both EAI and B2B integration.

Automation of business process execution is enabled by the fact that XML-based business process definitions are machine-readable and thus can be executed by a BPE. More specific, XML acts
as the bridge between the human-readable versions required for modeling activities and the machine-readable versions required by the run time environment, filling thus the gap between business processes and application components. The procedure of mapping a business process onto application components was depicted graphically in figure 2. In that example, a process definition is generated at build time, which is described in a machine-readable language. Currently, there are various XML-based business process definition languages, such as the Business Process Modelling Language (BPML) (BPMI, 2001), Web Services Flow Language (WSFL) of IBM (Leymann, 2001) and XLANG of Microsoft (Satish, 2001). Also, ebXML (2001b) has defined a Business Process Specification Schema (BPSS) (ebXML, 2001a) that provides a shared view of the interactions between trading partners regardless of the actions that lead to any particular interaction. The issue here is whether each of the three business process definition languages suffices for the description of an end-to-end process or they will have to include BPSS for the implementation of B2B collaborations. As a matter of fact, BPML, WSFL and XLANG do not support basic business notions such as mutual non-repudiation and authentication between parties. Nickull et al (2001) present how BPSS can be bound to each of the three leading business process specification languages (BPML, WSFL and XLANG).

As shown in figure 2, during the execution of the business process, the BPE communicates with the CMS, which in turn invokes the appropriate components. At the application components level, messages sent by the BPE are transformed into an appropriate format, in this example into CORBA IDL. An efficient transformation mechanism that can be used for such transformations is XSLT (Clark, 1999). XSLT is used for the transformation of an XML format to another XML format, to aware non-XML or to any arbitrary format (Holman, 2000).

As far as integration is concerned, XML is not an integration solution in itself – it is just a definition language, as explained earlier. For XML messages to be interpreted by other companies, both trading parties need to agree on common XML-based B2B standards, which will specify the document structures and the process descriptions. Such standards have already been developed by various B2B initiatives. Two major B2B initiatives are RosettaNet and ebXML (2001b). Hence, the B2B engine must be able to support any B2B protocol so as to provide for more flexibility.

For EAI, a candidate XML-based technology is Web Services (Fremantle et al., 2002). According to the Stencil Group (2001), Web Services are loosely coupled, re-usable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols. Web Services enable interoperability via a set of open standards. The most important Web Services standards are:

- WSDL (Web Services Description Language), which allows for formal description of the interface of a web service, i.e. the format of XML messages that the service processes and exchanges. Thus, WSDL allows a web service interface to be published in a repository where users can find the service and use the description to access it.
- SOAP (Simple Object Access Protocol), which is an XML based object protocol for the exchange of information in a decentralized, distributed environment. It consists of an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types and a convention for representing remote procedure calls and responses. All encoding is in XML.
- UDDI (Universal Description Discovery and Integration), which represents a set of protocols and a public directory for the registration and real time lookup of Web Services and other business processes. A simple scenario based on the use of these protocols could be as follows.

The CMS accepts a request from the BPE for a specific business service and gets information about the web services offering the specific business service, by doing a look-up in the private UDDI registry. The location and WSDL binding information of web services is sent to the CMS which formats an XML message accordingly and sends it via SOAP to the application responsible for delivering the requested service. Then the application offers the service by performing the relevant operation.

Clearly, Web Services can be aligned with the NGOSS framework, since it is a loosely coupled and a service oriented technology. WSDL can be used for the description of contract-defined interfaces, while UDDI can be used for the implementation of the run time registry. Moreover, Web Services can cater for the separation of the business logic from the implementation and the infrastructure details, because, firstly, in a Web Services enabling architecture the interface is separated from the implementation, and secondly, business logic can be expressed in WSFL.
5 MANAGING INFORMATION FLOW DYNAMICS

We have used the terms ‘agility’ and ‘agile’ to denote enterprise system architectures that offer the flexibility, integration and level of automation necessary for the management and control of information flow dynamics and, in particular via node density, velocity, viscosity and volatility. In this section we further elaborate on the ways that our proposed architecture addresses those parameters.

5.1 Node Density

Node density, according to Krovi et al., (2003), refers to the number of nodes included in a business process, where a node is used to describe an entity or a group of entities capable of altering the properties of information flow. In the proposed architecture, both internal entities and external constituencies are regarded as nodes within an end-to-end process and the BPE interacts with them without discrimination. The abstraction of business process flow into an entity (BPE) separate from the application components themselves allows an easier and more flexible way to adjust node density, i.e. to add or remove nodes from business process sequence, whenever new circumstances arise or a modification is needed. The only action required in such a case is a reconfiguration in the business process definition that is executed by the BPE, while no modification is needed at the application component level. Separating process control removes the need for individual components to have knowledge of the business logic associated with process operation. When invoked by process control, a component simply performs the service offered through its interface.

To better understand the above, consider the business process scenario of figure 2 and suppose for example that the company decides to add another node in the process right after the first step (receive order), which will perform error check in the order form. This node will probably be supported by a set of components. The only action needed in such a case is to extend the XML code appropriately in order to include the new function. After this, the BPE will be able to send the relative message to the CMS, which, in turn, will invoke the necessary components. In reverse, whenever the company decides to remove the new steps and reverse to the initial structure of the business process, the only action required is to remove the respective piece of code from the XML definition. In a similar manner, the same procedure can be followed in case of the addition or removal of nodes that correspond to users, partners, etc.

5.2 Velocity

The decoupling of business process flow from application components leads also to easier system integration. A highly integrated system, in turn, allows for high velocity, as all its entities can intercommunicate fast and in a seamless manner. Moreover, in the proposed architecture, the control of information flow is completely automated, since the BPE has the overall ‘supervision’ of business processes and is always aware of where to forward the information. In effect, the execution of business processes is much faster. All necessary information is available at the respective edge (user, application, and partner) in a timely manner. The information flow is smooth and conflicts, discontinuities or unnecessary delays, are prevented.

In addition, the fact that the BPE does not discriminate in the way it handles operations between internal and external entities helps in the integration of internal processes with B2B transactions. As a result, the internal enterprise activities are synchronised with the B2B transactions and therefore any temporal misalignment between them is eliminated ensuring at the same time the accommodation of high velocity levels. At another level, open communication protocols implemented internally through Web Services or externally via the B2B engine, ensure a high level of integration providing thus for the accommodation of high velocity levels in the flow of information.

5.3 Viscosity

The high level of automation and integration offered by the proposed architecture helps also in the attainment of low viscosity, since it leads to more accurate and streamlined information and ensures lower probability of error occurrence. As a result, conflicts that may arise at the nodes due to the arrival of contradictory information particles are avoided.

Again, referring to the example in figure 2, it is obvious that each step has well defined inputs and outputs, and in general the possible routes of the information flow are predetermined. Since the BPE offers this automation and also ensures that the correct routes will be followed for the required information when this is needed by the various nodes along the value chain, the appearance of errors and contradictory information particles will ultimately depend on how well the business process has been designed by the business process engineer.
5.4 Volatility

Volatility denotes the associated uncertainty in information content, format and/or timing (Krovi et al., 2003). Generally speaking, to cope with volatility in system terms means to develop a flexible system that can be easily adjusted so as to accommodate the extent of turbulence. This turbulence is of a polymorphous nature and one cannot claim without the benefit of hindsight that any enterprise architecture or system could by design accommodate all its manifestations. However, as far as content and format are concerned, we believe that both the design and the underlying implementation technologies as described in the previous section provide the highest possible flexibility. For example, nodes can be added easily, new application components can be embodied into the infrastructure and communicate with existing applications via the common communication bus, etc.

In addition the architecture can help manage volatility as it enables connectivity with a large number of external entities, which may themselves be sources of change. This is feasible because the BPE is scalable due to the existence of the B2B engine, which enables BPE to handle all equivalent relationships with a single business process definition. More specifically, the use of the B2B engine provides for the decoupling of the business process definition from the communication and business protocol details.

6 CONCLUSIONS

Contemporary architectures for enterprise systems should enable and not hinder the management of information flow dynamics. In this paper, we proposed an architecture that caters for the above by offering the necessary flexibility for the management and control of node density and volatility and enabling automation and high integration needed for accommodating variances in velocity and viscosity. Beyond conceptualisation we also outlined a number of implementation technologies for the key parts of the architecture. We would like, as a final note to this paper, draw attention to the organizational competencies required to manage the use and evolution of such architectures once in place. High agility requires a revisiting of the ways enterprises develop and handle their capabilities to organise and manage agile system infrastructures. ‘Organizational Emergence’ – “a theory of social organization that does not assume that stable structures underpin organizations” (Truex et al., 1999) (p. 117) can aid in this respect. Emergence theory emphasizes a continuous redevelopment perspective demanding the creation of a systems development environment that is optimised for high rather than low adaptation. This can be interpreted as an environment where maximum independence and flexibility is allowed at the user and business unit level, but with the necessary culture, policies and controls in place so as to avoid the introduction of conflict that could jeopardise the integrity and stability of the organization as an entity. According to Truex et al. (1999), the closer to ‘emergent’ the development environment gets, the more freedom it gives to each and every end user or node for participating in an active reality reconstruction. As requirements conflicts rise – as they undoubtedly will – increased negotiation and other service activities are prescribed and provided to support ongoing business processes. To achieve the above an extended number of organizational capabilities is clearly required that will define the form of the interface between internal and external nodes, i.e. the stakeholders of the infrastructure. These capabilities can be technical, economic, social, cultural, or a mixture of them all. Isolating and paying attention to one level or to one aspect of this interface, will be at the expense of the others, and ultimately will have negative consequences on the flexibility of any contemporary enterprise systems architecture. Further research is urgently needed towards this direction.

ACKNOWLEGDEMENTS

We would like to thank the partners of EURESCOM P1106 and in particular Derrick Evans of British Telecom whose input gave form to a number of ideas presented herein. We are also grateful to Nikos Korres of the University of Athens for his valuable assistance in the preparation of this paper.

REFERENCES


