22 DYNAMIC CONFIGURATION OF COLLABORATION IN NETWORKED ORGANISATIONS

Brian Shields and Owen Molloy
Department of Information Technology, National University of Ireland, Galway, IRELAND.
brian.shields@geminga.it.nuigalway.ie, ***owen.molloy@nuigalway.ie***

Levels of collaboration between networked organisations are at an all time high. There is a need for some control in the connections made between organisations and the information they pass to one another. Controlling the collaboration between two or more organisations involves providing an information structure to which messages between all communicating organisations must adhere to when passing information and specifying a protocol or protocols that must be used by the organisations to pass these messages. Any system employed to undertake the communication management between organisations must be dynamically configurable; therefore no organisation is "offline" while being configured.

1. INTRODUCTION

There is a close coupling between Enterprise Application Integration (EAI) and information based collaboration. Most modern EAI solutions use messaging passing in their implementation. Message passing between two organisations is essentially information based collaboration. However it is how the messages are passed and what the messages contain that determines the systems level of integration.
Most modern EAI and information integration solutions use some form of information broker. They use standardized transport protocols such as SOAP, standardized information representation such as XML, and provide some synchronous and asynchronous messaging solution.

2. EXISTING COMMUNICATION TECHNIQUES

As was mentioned earlier, there are strong links between EAI and inter-organisational collaboration. Until recently, EDI, or Electronic Data Interchange, had been the de-facto standard for the exchange of business documents since the 1970's. Due to the explosion of eBusiness and the need for real time delivery of

documents, a replacement technology was needed. The best performance offered by EDI was near time delivery.

According to Lublinsky (Lublinsky 2001), "Data-level Enterprise Application Integration (EAI) assumes bypassing the application logic and extracting or loading the data into the database through its native interface". This is an approach that is frowned upon by many EAI developers. There are numerous obstacles to database integration; for example, data representation and undocumented storage algorithms. This paper will explain the more modern and topical integration and collaboration techniques.

2.1 XML

XML, despite its name, is not a markup language. It is a set of rules for building a markup language (Ray 2001). XML complements inter-organisational communication perfectly. This is mainly due to its huge acceptance across all industries. An XML file must be coupled a schema for validation purposes. "XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for describing the structure, content and semantics of XML Documents"(W3C 2001).

2.2 Web Services

"A Web Service is a piece of business logic, located somewhere on the internet, that is accessible through standard-based Internet protocols such as HTTP or SMTP" (Chappell 2002). A Web Service has some special characteristics:

- It is XML based.
- It is loosely coupled.
- It is course grained; it is an interface to business logic, the logic itself is abstracted.
- It has the ability to be synchronous or asynchronous.
- It supports Remote Procedure Calls (RPCs).
- It also supports document exchange.

The three basic components of a Web Service are, Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery and Integration.

"SOAP is a lightweight protocol for the exchange of information in a decentralized, distributed environment"(Bequet 2001). SOAP uses XML to encode its payload.

"Web Service Description Language (WSDL) is an XML-based technology standard that defines Web Service interfaces, data and message types, interaction patterns and protocol mappings" (Sharma 2002). There are six XML elements in the WSDL file; these six elements describe the name and location of the service, the data types used between the client and the server and the messages passed between the client and the server.

"The Universal Description, Discovery and Integration (UDDI) project provides a standardized method for publishing and discovering information about Web Services"(Chappell 2002). There are three types of information that an organisation can register in a UDDI register; the organisations contact information, information

describing the Web Service and technical information that describe the behaviors and supported functions of a hosted Web Service.

2.3 Messaging

There are two distinct types of messaging, synchronous and asynchronous. Synchronous messaging occurs when one application makes an RPC to another application, and must wait for a response. Asynchronous messaging, on the other hand, is the messaging involved in Message Oriented Middleware.
Messaging Oriented Middleware (MOM) insures asynchronous and reliable messaging between applications. MOM technology is important in Inter-organisational communication. A group of networked organisations can use the information or message broker approach. An organisation may pass all messages to this broker, the broker will route the message to the appropriate recipient, and therefore remove the need for each organisation to develop their own complex message sending and receiving handlers.
According to Eric Roch of TIBCO, modern MOM technologies must possess the following features (Roch 2002):

- The use of a publish and subscribe approach, therefore removing application knowledge from the MOM.
- Reliability and serialization of messages.
- The use of subject based addressees, therefore abstracting all network details.
- Real time delivery using multicast.
- Support for multiple communication models.

3. INTELLIGENT COLLABORATION

Intelligent systems are increasing in popularity as solutions to enterprise software challenges. This section of the paper presents the most common intelligent features in EAI and information integration solutions.

3.1 Business Rules and RuleML

Expert systems provide the ability to separate business logic from application logic. Using business rules, application specific data in ERP systems can remain static in a dynamic environment environment. A business rule is a statement that describes how a business is organized, how a business decision is made, or how a business process works (Blaze Software 1999). In other words, a business rule is a set of conditions that govern an event so that it occurs in a way that is acceptable to the business. Most business rules associate a condition to an action and naturally take the form of an IF-THEN statement.

RuleML is a markup language, creates in XML, used for representing rules in a structure that minimizes ambiguity. The RuleML Initiative started in August 2000 during the Pacific Rim International Conference on Artificial Intelligence (PRICAI2000). A RuleML file, which contains a set of rules, is constructed according to an XML Schema, agreed upon by the RuleML Initiative (Boley 2001).
An inference engine is then used to implement some decision logic within a process flow, as specified by a business rule.

3.2 Agents and Agent based systems

Describing the meaning of the term "agent" or "agent-based systems" is an activity which can ignite heated debates in the software world. There is no industry agreed definition of what an agent is, but most researchers would find themselves in agreement with Wooldridge and Jennings (Wooldridge 1995); "An agent is a computer system situated in some environment that is capable of autonomous action in this environment in order to meet its design objectives". There has been a huge increase in the application of agents in software. Agents are becoming the preferred choice in applications where the environment is not completely static.

4. PROPOSED SOLUTION

The current solution to the information integration problem uses an information broker. This solution can have business logic hard-coded into the information broker as is depicted in Figure 1 below.

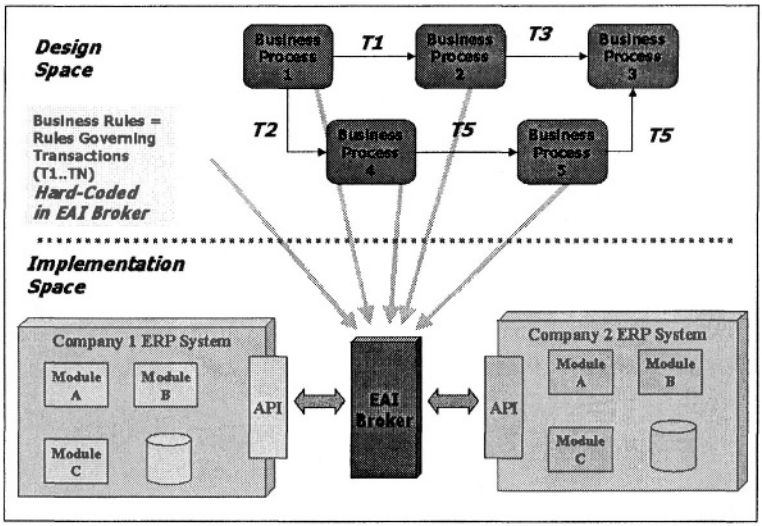

Figure 1 - Information Broker Solution

This paper proposes a new solution called XESS, an XML based Expert System Shell, involving the afore mentioned technologies. Business rules are created by a Business Rules Composer and passed, via Web Services to a Business Rules Engine of the XESS agent. Each node in the networked organisations contains an XESS agent. This agent contains a business rules engine which will validate business rules and then execute the associate business processes, a repository of precompiled rule sets, XML Schemas and XSL files. A high level design of the system is shown in Figure 2.

Each business rule is written to correspond to a particular XML Schema, therefore a particular type of XML document. The business rule can include attributes and elements from the XML Schema file in the condition and action statements. These attributes and elements will be replaced by the actual values from

an XML file when an appropriate one enters the system and triggers the validation of a rule set.

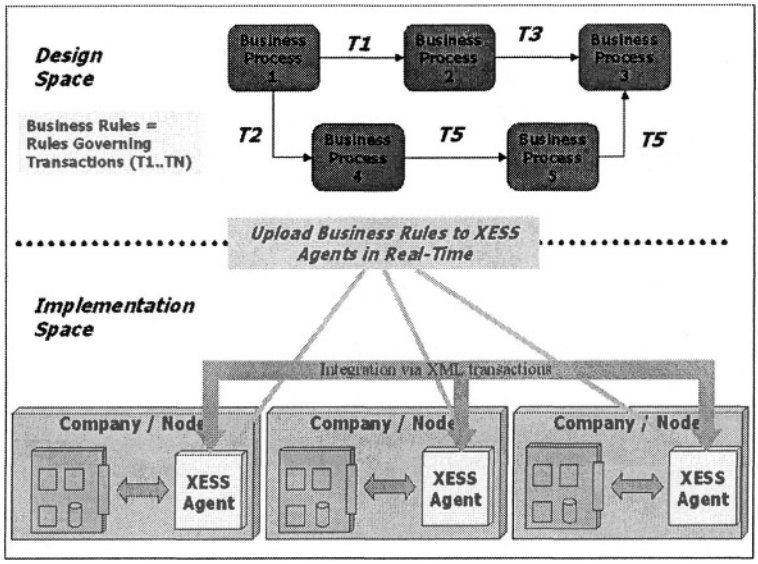

Figure 2 - (Virtual) Information Implementation using XESS and Information Broker (Shields 2003)

The architecture of the XESS system contains three main components, the Business Rules Composer, the Agent Administrator and the Business Rules Engine Agent. The architecture of the XESS Solution is shown in Figure 3.

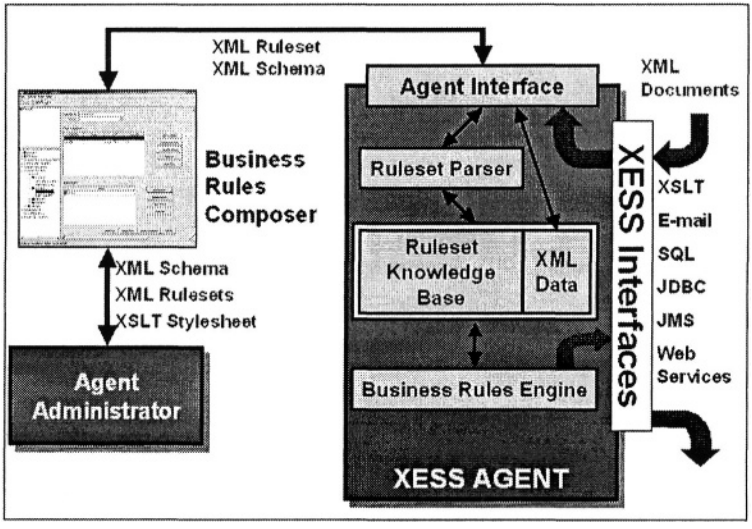

Figure 3 - XESS Architecture

The Business Rules Composer creates RuleML documents containing the business rules and loads it onto a remote Business Rules Engine Agent. These Business Rules

are created according to a previously uploaded XML Schema. The upload of rulesets is achieved using Web Services. The Agent Administrator uses J2EE containers to hold all necessary information about each agent. Information such as Web Service endpoint addresses, remote method signatures, locally stored XML Schemes and XSLT files.

This paper will primarily deal with the Business Rules Engine Agent.

4.1 Engine

The Engine is the principal element of each agent. The engine contains the rule compiler and the inference engine. The rules are inputted from the Business Rules Composer in RuleML format. The rules are compiled into Java classes, and are stored until an XML document, the fact base, is received. The rules are then tested and the valid rules prepared for firing.

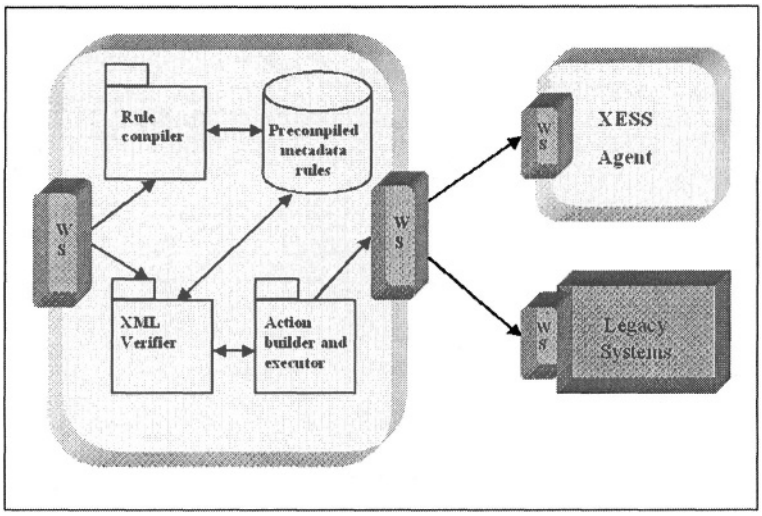

Figure 4 - XESS Engine Architecture

Figure 4 is a architecture diagram on the Business Rules Engine; it contains three business components and a data store.

4.1.1 Rule Compiler

The Rule Compiler is responsible for transforming the RuleML file into a RuleSet object, containing one or more Rule objects, each of which contain one or more Condition objects and one or more Action objects. A Rule object contains a name, a priority, a hypothesis, a list of Condition objects, a list of Action objects and a list of rule dependencies.

The hypothesis is the validity of the Rule, that is if the Rule is true or not. A Rule can only be true if all the conditions are true; the hypothesis is then set to true. If any of the conditions are false, the hypothesis, and therefore the Rule, is false. If the Rule has not been completely evaluated the hypothesis remains null. The list of Rule dependencies is the list of Rules that must be fully evaluated before this Rule can be evaluated. This concept is explained in the next section of this paper.

When the RuleML file arrives at the Engine, it is verified against the RuleML Schema, which is stored locally, in order to ensure the Rules are correctly constructed. There is also a check conducted on the Rules to ensure no Rule cycles exist. A Rule cycle occurs when the Condition of one Rule contains the hypothesis of another Rule as a parameter, and this Rule then refers to the former Rule. A type of deadlock occurs. If the RuleML file contains a rule cycle, the file is dropped and a notice sent to the user who has built the RuleSet. If the checks are completed satisfactorily, the file is then iterated through to parse all the contained Rules. A RuleML file contains at least one rule but usually multiple rules. For each Rule in the RuleML file, its Conditions and Actions are parsed and added to the Rule, the Rule is then added to the RuleSet. The RuleSet is then serialized and stored.

4.1.2 XML Verifier

When an XML file is inputted to the system, it is first validated against its XML Schema. If its XML Schema does not exist on the agent, the XML file cannot be processed. Each RuleSet is constructed according to a particular XML Schema, therefore if there is no XML Schema on the agent to verify the XML structure, there is no RuleSet to execute over the XML data. When the XML file has been verified as to being legally constructed according to its XML Schema, a list of Constraint objects are created. The Constraint class is a Java business class that contains one element or attribute from the XML file. It holds the name of the element or attribute, its XPath in the XML file, its value and the type associated with its value. Speed of execution of the RuleSet is the reason for this compilation. As was explained earlier, most business rules will contain a variable value that must be parsed from the XML file in order to execute the business process, instead of parsing the XML file each time the engine comes across an XML variable in a business rule, the engine can now perform a lookup on the collection of Constraints.

Rules are arranged in the RuleSet in order of increasing number of dependant Rules associated with them. This has been seen to improve the speed of validation of the business rules.

For a Rule to be evaluated, each of the Conditions in the Rule must be evaluated. The Condition will either refer to another Rule, which is the simplest form of Condition, or it will contain XML variables. The variable is located in the collection of Constraints and is populated into the Condition. The Condition is then evaluated as a simple IF statement using an operator specified by the user who created the business rule. If all the Conditions are true, the Rule is true, and therefore the Action of the Rule is added to an Action queue, according to a priority, again specified by the creator of the business rules. Once all the Rules are evaluated the Action queue is passed to the Business Process Execution component, which will fire each action in turn.

4.1.3 Action Execution

There are six basic actions involved in the current version of our proposed solution. They are:

- Email Action. An email is constructed according to the elements specified in the RuleML file. This email can have the XML file as an attachment.
- Forward Action. This action forwards the XML file, unedited, to another agent in the network.
- Save Action. This saves the XML file to a specified location in the network.

- Web Service Action. This calls any Web Service. This Action is used to access Web Service-wrapped legacy system information.
- XSLT Email Action. This transforms the XML file using an uploaded XSL file and saves it to a specified location.
- XSLT Save Action. This transforms the XML file using an uploaded XSL file and emails it to a specified address.

All of the Actions are Java business classes, each one extending the abstract Action class, therefore a level of abstraction can be used in the firing of the ordered Actions in the Action queue.

4.2 Transport Protocols

To provide a credible business solution, it was necessary to include a number of transport protocols for input and output of data from the engine. The protocols accounted for to date are SOAP, JMS, SMTP and FTP. These are the most common methods of information passing in today's inter-organisational communication. Figure 5 is a diagram of the interaction between the XESS Agent and its clients using these transport protocols.

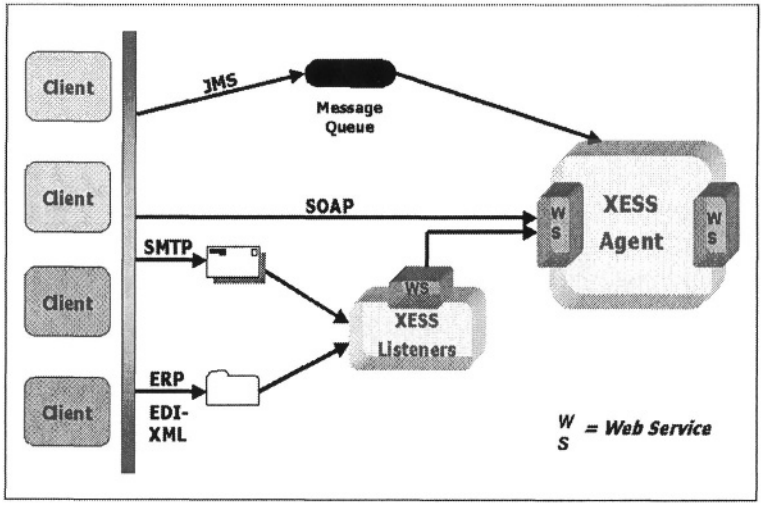

Figure 5 - XESS Transport Protocols

The XESS Agent creates and starts two threads of execution to act as information listeners. One thread listens to a specified email address inbox for the receipt of SMTP messages. It processes this email, populates an appropriate Web Service call to the Business Rules Engine and invokes it. The second thread listens for FTP updates to a specified folder on the network. As with the first thread, it populates a Web Service call to the Business Rules Engine and invokes it.

5. TESTING

A number of practical scenarios were devised to test XESS. One of these involved using XESS in a virtual multi-organisational supply chain. The supply chain consisted of three virtual organisations, all running ERPLite, a lightweight ERP software package. An XESS agent was deployed on two of the nodes in the supply chain and BizTalk server was installed on the third. Figure 6 is a model of the virtual supply chain.

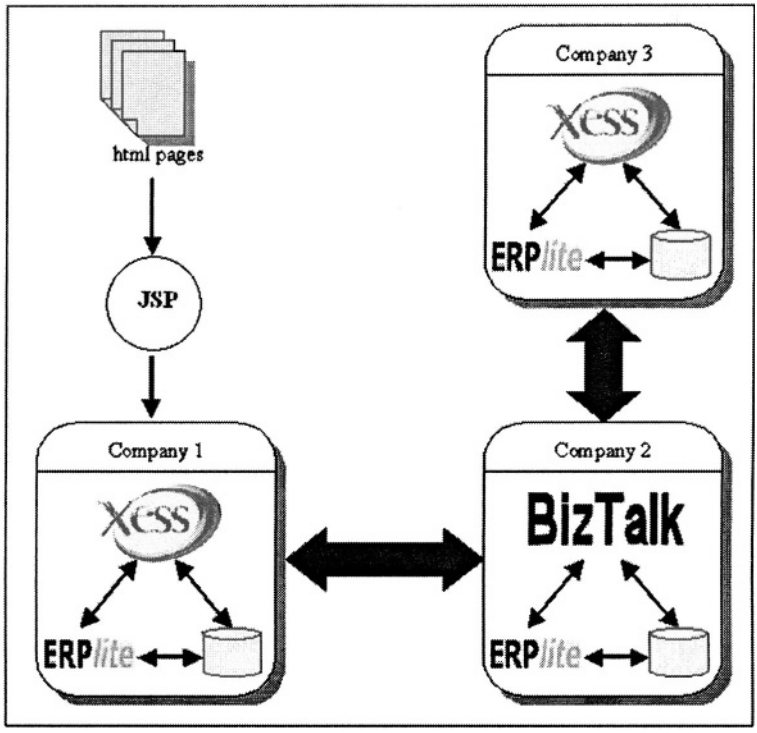

Figure 6 – XESS Solution for Supply Chain Management

The customer fills up an online order form. A purchase order is created in XML from this data and sent to Company 1. The XESS agent residing on company 1 intercepts this order, evaluates the data according to a precompiled rule set, and executes the appropriate actions; which may include database updates, legacy web service calls or forwarding the document to further nodes on the supply chain. From this scenario we concluded that an XESS agent could be deployed with equal success as a intelligent document routing system or a more complex expert system. Collaboration between nodes in a supply chain, such as the example in Figure 6, is necessary to provide optimization. Rule sets can be built and uploaded to multiple agents that force them to interact with each other. The dynamic, run-time upload of business rules to XESS agents would be integral in a live supply chain where hundreds of purchase orders are received per day.

6. FUTURE WORK

The XESS solution lacks any security model for its document transfer, which is potentially dangerous particularly in an environment such as that described in the previous section. Purchase orders often contain sensitive data. With respect to security, one of the greatest strengths of XML, its relatively comprehensive nature, has become one of its biggest weaknesses. A security model for XESS must address three main security issues; authentication, encryption and access control.

7. CONCLUSION

This paper has described the technologies currently available to developers of application or information integration solutions. We then proposed a solution for this domain using these technologies; namely, XML, Web Services, messaging, inference engines, business rules and agent theory. We then documented a number of performance tests that were carried out on the proposed solution.

XESS is a research initiative sponsored by Enterprise Ireland's Research Innovation Fund. Enterprise Ireland is a government organisation charged with assisting the development of Irish Enterprise.

8. REFERENCES

1. Bequet, H. (2001). Java SOAP, Wrox Press Ltd.
2. Blaze Software, X. S. (1999). "Blaze Advisor Technical White Paper."
3. Boley, H., Said Tabet and Gerd WagnerXML Schema (2001). Design Rationale of RuleML: A Markup Language for Semantic Web Rules. International Semantic Web Working Symposium, Stanford University, CA.
4. Chappell, D. A., Tyler JewellXML Schema (2002). Java Web Services. O'Reilly and Associates, Inc.
5. Lublinsky, B. (2001). Achieving the Ultimate EAI Implementation. Part 3: Data-Level Integration, EAIJournal.
6. Ray, E. T. (2001). Learning XML, O'Reilly & Associates, Inc.
7. Roch, E. (2002). Application Integration
8. Business Case and Technology Trends, eai.ebizq.net.
9. Sharma, M. a. D. (2002). Defining Web Services using WSDL, WebJives.
10. Shields, B. (2003). An Agent Based Approach to Enterprise Application Integration. The International Conference on Internet Computing, Las Vegas, NV, USA, CSREA Press.
11. W3C (2001). XML Schema, W3C XML Schema Working Group.
12. Wooldridge, M., N.R. JenningsXML Schema (1995). "Intelligent Agents: Theory and Practice." The Knowledge Engineering Review **10**(2).