TABLE OF CONTENTS

TABLE OF FIGURES

SUMMARY

NB. The meanings of abbreviations and acronyms used in this thesis are provided in a table following this summary.

The primary contributions to the area of electronic business integration, propounded by this thesis, are (in no particular order):

- *A novel examination of global Business-to-Business (B2B) interoperability in terms of a "multiplicity paradox" and of a "global electronic market-space" from a Complex Systems Science perspective.*
- *A framework for an, integrated, global electronic market-space,* which is based on a hierarchical, incremental, minimalist-business-pattern approach. A Web Services-SOA forms the basis of application-to-application integration within the framework. The framework is founded in a comprehensive study of existing technologies, standards and models for secure interoperability and the SOA paradigm. The Complex Systems Science concepts of "predictable structure" and "structural complexity" are used consistently throughout the progressive formulation of the framework.
- *A model for a global message handler* (including a standards-based message-format) which obviates the common problems implicit in standard SOAP-RPC. It is formulated around the "standardized, common, abstract application interface" critical success factor, deduced from examining existing models. The model can be used in any collaboration context.
- *An open standards-based security model* for the global message handler.

Conceptually, the framework comprises the following:

- An interoperable standardized message format: a *standardized SOAP-envelope* with *standardized attachments* (8-bit binary MIME-serialized XOP packages).
- An interoperable standardized message-delivery infrastructure encompassing an *RPC-invoked message-handler* - a Web service, operating in synchronous and/or asynchronous mode, which relays attachments to service endpoints.

- A business information processing infrastructure comprised of: a *standardized generic minimalist-business-pattern* (simple buying/selling), comprising *global pre-specifications for business processes* (for example, placing an order), *standardized specific atomic business activities* (e.g. completing an order-form), a *standardized document-set* (including, e.g. an order-form) based on *standardized metadata* (common nomenclature and common semantics used in XSD's, e.g. the order-form), the *standardized corresponding choreography for atomic activities* (e.g. acknowledgement of receipt of order-form) and service endpoints (based on *standardized programming interfaces and virtual methods* with customized implementations).

ABBREVIATIONS AND ACRONYMS USED

The following abbreviations and acronyms occur in this document. Explanations are supplied only in the text where it is doubtful whether a term is in everyday use.

AES	Advanced Encryption Standard
AH	Authenticating Header (an IPSec protocol)
ANSI	American National Standards Institute
ASC X12	ANSI's Accredited Standards Committee X12 EDI Standard
ATM	Asynchronous Transfer Mode; high-bandwidth packet- switching technology.
ASP	Application Service Provider OR Active Server Pages (a Microsoft technology)
B2B	Business-to-Business E-Commerce
B2C	Business-to-Consumer E-Commerce
BPEL4WS	Business Process Extension Language for Web services
CA	Certification Authority (for digital certificates)
CGI	Common Gateway Interface
CISC	Complex Instruction Set Computing
CORBA	Common Object Request Broker Architecture
CUG	Closed User Group
cXML	Commerce XML
DES	Data Encryption Standard
DHTML	Dynamic Hypertext Markup Language

DISCO	Discovery protocol for locating Web services
DTD	XML Document Type Definition
ebXML	Electronic Business XML
EDI	Electronic Data Interchange
ESP	Encapsulating Security Payload (an IPSEC protocol)
FIPS	Federal Information Processing Standard
FTP	File Transfer Protocol
HMAC	Hashed Message Authentication Code
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange, a combination of ISAKMP and OAKLEY; also known as IKMP
IKMP	Internet Key Management Protocol
IOIS	Inter-organizational Information System
IOS	Inter-organizational System
IP	Internet Protocol, a connectionless network layer protocol.
IPSec	Secure Internet Protocol, provides security services between pairs of nodes
ISO	International Standards Organisation
ISP	Internet Service Provider
MIME	Multi-purpose Internet Mail Extension
MTOM	Message Transmission Optimisation Mechanism (a SOAP services standard)
PGP	Pretty Good Privacy, an e-mail privacy utility that uses public key encryption
PKCS	Public Key Cryptography Standard from RSA Security Laboratories

PKI	Public Key Infrastructure (for digital certificates)
PKIX	Public Key Infrastructure X.509 (IETF)
PPP	Point-to-point protocol, a layer 2 WAN protocol
PPTP	Point-to-point Tunnelling Protocol, a Microsoft VPN encapsulating Layer 3
RISC	Reduced Instruction Set Computing
SAML	Security Assertion Markup Language
SET	Secure Electronic Transaction, an e-commerce protocol
SHA-1	Secure Hash Algorithm – 1
SME	Small to Medium Sized Enterprises
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol, an XML-based messaging protocol
SOCKS	A session layer proxy security protocol.
SSL	Secure Sockets Layer, a protocol for incorporating encryption into e-commerce transactions, developed by Netscape
SQL	Structured Query Language
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/ Internet Protocol
TLS	The IETF’s Transport Layer Protocol (also called SSL 3.1).
UDDI	Universal Discovery Description and Integration standard for Web services
UDP	User Datagram Protocol
UML	Universal Modelling Language
UN/CEFACT	United Nations’ Centre for Trade

	Facilitation and Electronic Business
URI	Uniform Resource Identifier
VPN	Virtual Private Network
W3C	World-Wide Web Consortium
WAN	Wide Area Network
WSDL	Web Services Description Language, an XML grammar for describing Web services
WS-*	Refers to all Web services standards
WS-S	Web services security standard
XACML	XML Access Control Markup Language
xCBL	XML Common Business Library
XKMS	XML Key Management System
XML	Extensible Mark-up Language
XML DSIG	XML Digital Signature
XOP	XML-binary Optimisation Packaging
XSD	XML Schema Document

Chapter 1

The Nature and Context of the Problem

Introduction

In this chapter the nature and context of the problem to be addressed is introduced. It also introduces the approach to be used in this thesis.

The Nature of the Problem

Even though business concepts can be described in (language-equivalent) terms which have common connotations across the globe, the inherent diversity among business systems inhibits global standards-based collaboration to a large extent. Business systems may differ in terms of, inter alia: products and services, broader goals and objectives, business models, business processes, customer-relationship management, and ambient factors such as culture. Even within a particular enterprise, business-collaboration can be limited due to a lack of *collaboration standards*. Collaboration standards may conceptually be divided into *common business standards* (such as, business processes, common vocabulary and semantics, and common documents) and *common technical standards* (such as, common protocols, and common computer-based mechanisms for data-exchange). The former should, ideally, be pre-defined by experts within the business domain; they will only be considered here in respect of generic requirements and implications for a common technical infrastructure.

The raison d'être for wide-ranging *collaboration* across great divides does not need much elaboration. In modern terms, this intuitively implies electronic collaboration. Greater *electronic collaboration* promotes greater *integration* within a defined collaboration-space. Greater integration implies *communication and sharing* of, for instance, resources, methods and ideas. In terms of electronic collaboration, integration effectively refers to the extent to which *interacting participants are linked by way of information technology* (essentially, computer software and hardware, although supplementary means of integration are not precluded).

A state of electronic integration (a state of communication-and-sharing) is preceded by a state of *interoperability* (a state of technical-readiness-for-communication-and-sharing). Interoperability implies providing a means/mechanism by which disparate systems can interact seamlessly (as an integral whole). Given the dynamic nature of information technology, interoperability can, more accurately, be considered to be a matter-of-degree, with “optimal interoperability” being an ideal that is constantly being strived for.

In order to ensure that the ensuing integration “system” has a *predictable structure,* with predictable states (of interoperability/integration), the pursuit of devising the structure of the system must be controlled (/managed); this of necessity suggests a high order of *pre-specification*. It is reasonable to speculate that the lower the *structural complexity* of the system, the more manageable the system. Conversely, increasing structural complexity – in terms of a large number of state-variables and “laws” (statements) governing the behaviour of the system - will cause it to tend towards becoming non-deterministic (unpredictable).

It is contended in this thesis that allowing the proliferation of discrete pockets of dyadic integration (1:1 or 1: n mutual-arrangements) would ultimately lead to non-determinism (chaos), when viewed from a global-integration perspective. As a counter-measure, the factors of disparateness - state-variables and “laws” - could either be abstracted into *a common, standardized interface specification, or be translated/ interpreted by a common, standardized “gateway” interface*. In the latter case, the complexity of the interface would depend on the total number of permutations of possible state-variables (and “laws”); this is, obviously, the less-predictable option. In the former case, the standardized interfaces eliminate the need for reconciling differences in interacting state-variables (and “laws”). For example, different transport and network protocols, or different security protocols could be replaced by pre-specifying a common interface (with a common protocol-profile) for all participants.

Pre-specifying a common, standardized interoperability-interface specification will, therefore, have the effect of both reducing complexity of the (global) system and promoting (global) interoperability. This standardized specification must simultaneously permit disparity beyond the shared interface and yet enable the disparate systems to integrate, by pre-specifying the interface components (such as a standardized, interoperable messaging protocol). Figure 1 below illustrates the scenario depicted here.

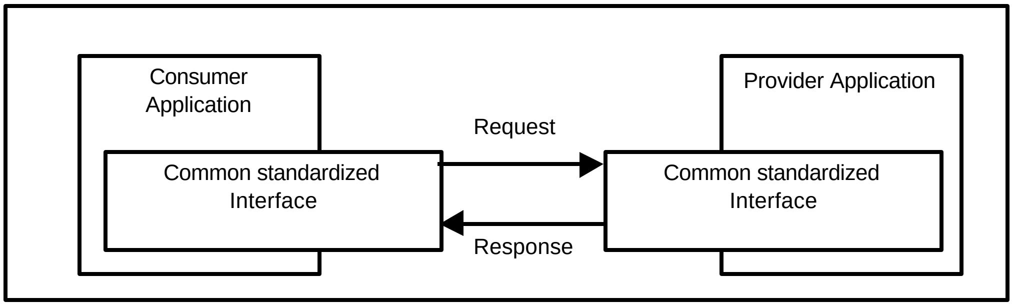


Figure 1. Disparate applications require a pre-specified common interface for

Further, to be universally-applicable, the common, standardized interface specification must be provided by (or at least be endorsed by) a global standards body; it must be based on open (freely-available, non-proprietary) standards; and the system must be freely-available and accessible to all potential participants.

The Context of the Problem

This thesis examines the existing paradigms, technologies and standards for electronic integration within a global context, referred to throughout as a “Global Electronic Market-Space” (GEM). The “Global Electronic Market-Space” concept is difficult to define unequivocally due to its essentially hypothetical nature. It must be distinguished from related contemporary phenomena such as Grid Services and Service-Oriented Architectures, yet is inextricably intertwined with them. “Market-space” refers to an abstract, virtual domain for business-to-business (B2B) collaboration. It encompasses a wide variety of business models, with concomitant business flows, business transaction types, atomic business activities, business documents, document-exchange patterns, and so forth.

The GEM-context can be loosely described as a virtual market-space within which all business systems can potentially participate in a seamless fashion, irrespective of type, platform, size or location, in a free, non-affiliated manner. Superimposed on this multifaceted scenario are various implementations, and configurations, for numerous contextual issues, such as security. For instance, how and where does one implement and configure directory services for participants in a multi-party collaboration? This is one of many problems which will be explored later.

A Simple Analogy

The nature and context of the problem can be clarified by means of a simple analogy (adapted from Mak & Ramaprasad, 2001): How would individuals in different parts of the world work together as a group in a non-computerized context? In attempting to answer this question, one could – through inductive conjecture - arrive at the following typical minimum prerequisites:

Working together as a single group presupposes a means of *discovery of suitable potential participants.* All participants must either *share a common language*, or have translators for each language used. The *mode(s) of communication* and the *message format(s) (for example, document-structures) must be pre-specified*. They must be able to contact one another using *a pre-specified common (secure) protocol-set for interacting* (essentially, sending and receiving messages). They must *agree on goals and tasks* and *designate roles* for each participant. Each participant must *complete the assigned tasks in the time, sequence and manner commonly-agreed upon*. “End-point” processing can be individualised but must subscribe to a *common service-level agreement*.

Assume that the group refers to a particular business segment (for example, an industry-group such as the automotive industry). Now, consider the situation where various groups (perhaps, various industry-groups) interact with each other (in a cross-industry fashion). How does one establish uniformity in the cross-industry matrix for pre-existing groups in which collaboration mechanisms have already been defined? Now, add the independent participants that have no specific affiliation to any group. This mimics the quintessential pre-GEM-context. The requirement for pre-specification (standardized interoperability states) conducive to global electronic integration depicts the nature of the problem to be addressed.

A More-Formal Information Systems Perspective

From a formal systems perspective, inter-business collaborations would ordinarily be referred to in terms of “workflows”, “business processes”, “collaborations”, “collaboration choreography”, and so forth. The following are some of the interoperability aspects of the collaboration that would require consideration:

- *Inter-organizational pre-specification* of requirements (goals and objectives), strategies (including degree of systems integration, timeframes, and so forth), workflow modelling, business-process orchestration/choreography modelling, common nomenclature (common semantics, common vocabulary), and roles.
- *Infrastructural/architectural pre-specification* of environments/platforms, interoperability technologies and standards, security model, system configurations and common protocols (network, transport, application, and security).
- *Developmental/implementation pre-specification* of coding paradigm, error-handling, templates, libraries, configuration and performance tuning.
- *Support pre-specification* of administration procedures, user management, scalability and availability.

Typically, industry participants would have representatives who would meet to compile the pre-specifications, as was the case with RosettaNet (RosettaNet, 2002). In a cross-industry GEM-context, pre-specification would seem an arduous task. However, as will be discussed in the Methodology section, a formal systems analysis-and-design approach is eschewed in this thesis.

Initial Contentions Raised in the GEM-Context

The following points, inter alia, will be asserted in relation to the hypothetical GEM-context, in this thesis.

Pre-specification should be handled by a group of experts, under the auspices of a global standards body, such as the World-Wide Web Consortium. A specialized registry/repository should be utilized for discovering participants and to disseminate specifications, templates and even core (programmatic) components for facilitating participation in the GEM.

The participating application systems should not have to be homogeneous. The ramifications of heterogeneity implicit in participating systems will be explored further in subsequent sections and chapters.

It will be argued that interoperability is achievable by pre-specifying common standards at the conceptual/pre-implementation level and abstracting configuration variables to *a common, platform-independent application interface*. For participants that have invested in legacy systems, interoperability within the common "market-space" is attainable through overlaying existing systems with such a globally-uniform interface for accessing the market-space. The specified interface will reduce the degree of "required commonness" in individual participating systems. However, a certain degree of obligatory commonness to be imposed on collaboration in this GEM, by a global authoritative standards body (such as the World-Wide Web Consortium) is inevitable.

Electronic collaborations comprise innumerable possible permutations of business processes, atomic activities and business documents. If the group (of people) in the analogy were to expand to include other groups (other industries, for instance), the number of permutations would increase exponentially. Yet, it will be argued, automated business process integration on a global scale, that is, between all participating systems, can be achieved with appropriate standardisation. By this is meant that every participating provider application will, potentially, need to be able to process data contained in messages exchanged with any other participating consumer system. For this, *a comprehensive, standardised business collaboration system* is required.

More specifically, it will be argued in later chapters that a pre-specified protocol-profile for the messaging, transport and network layers, together with a common nomenclature (based on a common semantics and a common vocabulary) for business processes and business documents, a common security model and a common application interface, are essential to ensure that participating systems in the GEM are interoperable.

A major contention in this thesis is that the prevailing approach to multilateral B2B-collaboration needs to be altered dramatically if a Global Electronic Market-Space (GEM) is ever to be realized. This concept is introduced in Chapter 2 in terms of a “Multiplicity Paradox”.

The Problem Statement

Essentially, the problem defining this study may be stated as follows:
Beyond the usual collaborations on the Internet, there exists a need for a global electronic market-space in which businesses of all types, sizes and location can participate. Can a mechanism for interoperability be designed that will facilitate secure and seamless collaboration at this global level, allowing disparate systems to form an integrated whole?

Methodology

It will be shown in Chapter 2 that B2B-collaboration is a complex “system”. However, a systems approach would be rather pretentious for the following reasons: the dynamic nature implicit in the existing B2B standards; the fact that some of the standards being proposed have no authoritative status; and the fact that the analysis-and-design would be non-user-centric (contrary to modern software-development life-cycle methodologies). Although a concurrently-developed rudimentary prototype (Appendix A) of the model proposed in Chapter 5 does exist, this is merely being used as proof-of-concept for some of the technical assertions made in the thesis.

The model is therefore essentially conceptual; it will focus on what should-not-be as much as what should-be aimed for. Given the complex, multifaceted and global nature of the model, a Complex Systems Science perspective will be adopted, where considered appropriate. Nevertheless, with each chapter, the requirements-specifications will be deduced, in a progressive fashion, from an analysis of existing models, technologies and standards, inherent problems/shortcomings and known critical success factors.

It will be argued that, from a GEM-perspective, the prevailing "system" is tending towards non-determinism. An attempt will be made to analyse each aspect of the system in order to converge, piece-meal, on the "*attractor*" region (the state space around which states of the system fluctuate) for that aspect (Goertzel, 1994:1-23). Goertzel (Ibid) posits that the structure of the dynamics of a complex system consists of the patterns in its "*strange attractor*" (when the behaviour which results from the set of states fluctuates in a chaotic manner). The "multiplicity paradox" discussed in chapter 2, it will be argued, represents a "strange attractor" in the B2B-collaboration system and will be shown to have a marked effect on the (predictable) structure of the dynamics of the pre-GEM "system". *Patterns* (typically processes; representations as "something simpler") will be identified – and indicated as such - at intervals throughout the thesis. These paraphrase the deduced requirements-specifications alluded to earlier as well as technologies, standards and paradigms which might be used to serve the common end (GEM-interoperability). Complex Systems Science terms used in the next paragraph are based on Goertzel (1994).

Initially, each pattern in a chapter might appear as a member of a "*fuzzy set*" (membership is not "either/or" but gradual). Chapter 5 will represent the *gestalt* (the patterns that appear when all the patterns from previous chapters are considered as a *whole*, but not in each individually) of the desired "patterns" derived at the end of each chapter. If the structure of the thesis, St(x), is the set of all the patterns in chapter x, then the *emergence* $Em(x_0,.., x_n) = St(x_0,.., x_n)- St(x_0) - St(x_1)…. - St(x_n)$ is the gestalt (Ibid:19-20). Patterns will be related to known requirements criteria for each aspect of the system; the latter criteria will also be used as critical success factors (CSF's) in the efficacy-analysis of the proposed model. Patterns will be examined in terms of whether their utilisation/occurrence in the GEM would contribute to structural complexity.

One key element of non-deterministic behaviour is *sensitive dependence on initial conditions* ("a positive Liapunov exponent"); it is hopeless to attempt to predict the exact value of the future state of a system which displays sensitive dependence

However, if such a system displays enough regularity of overall structure, one can roughly predict future patterns (Ibid: 21). To this end, current and previous paradigms are the focus of Chapters 2 and 3. It will be argued that the proposed GEM can be *structurally predictable*, and will simultaneously be able to evolve.

Brief Chapter-Analysis

Chapter 2 asserts that current applications of B2B-collaboration standards are antithetical to the establishment of a GEM. A phenomenological analysis of B2B-collaboration will be attempted and the key concepts - integration and interoperability - will be defined. Given that contrived collaborations occurring along industry-specific lines (for example, Covisint (www.Covisint.org) in the motor-vehicle industry and RosettaNet in the IT-focussed industries) are well-established, one may question if there is sufficient reason to contemplate a global trading arena in which anyone/everyone can participate. The motivation for a Global Electronic Market-Space (GEM) will thus be examined.

Chapter 2 also describes the motivation for the proposed model discussed in Chapter 5. It argues primarily that current B2B-collaboration technologies and standards, while attempting to promote interoperability between participants in collaborative relationships, in fact act counter-productively in terms of global interoperability. Innumerable dyadic configuration-permutations are being created on an ad hoc basis. This is mainly due to the “flexibility” (neutrality) that is accommodated in terms of protocol profiles, business process flows, business documents, and so forth. This situation is referred to as the “Multiplicity Paradox” in this thesis.

In Chapter 3, an attempt is made to answer the following question: If the basis for resolving the interoperability issue is to be found in interface technologies, can existing interface technologies be used to address the “Multiplicity Paradox” issue discussed in Chapter 2? Interface technologies and Services-Oriented Architectures (SOA’s) are elaborated. Chapter 3 also examines the feasibility of the ubiquitous Web service application interface as a possible means to arriving at global interoperability. The evolution of remote procedure call (RPC) models in SOA environments is explored. The changes are briefly discussed in terms of sensitive dependence (predictable structure) and structural complexity.

Chapter 4 extends the ideas developed in Chapters 1, 2 and 3. In this chapter the pros and cons inherent in existing B2B-collaboration models, technologies and standards in relation to a GEM are analyzed. Typical examples include: RosettaNet (www.RosettaNet.org), Microsoft's BizTalk (www.BizTalk.org), and ebXML (www.ebXML.org). The integration objectives implicit in these models are examined for possible critical success factors to be used in the context of a GEM. Reasons are provided for why these models are not suitable for creating the envisaged GEM context.

Chapter 5 represents a synthesis of the patterns explored in previous chapters. A framework (the gestalt) is proposed for promoting interoperability in a global market-space context.

Chapter 6 evaluates the model using the Critical Success Factor Analysis method. It is also the conclusion chapter.

Chapter 2

GEM: The Nature of the Beast

Introduction

In Chapter 1, the concept of a global electronic market-space (GEM) was introduced. A GEM represents an “ideal” virtual space in which businesses of all types and sizes, from all over the world, can interact as an integral whole.

In this chapter it is argued that although sufficient motivation exists for the realisation of a GEM infrastructure, current B2B-collaborations based on protocol-neutral technologies and standards propagate unwieldy permutations of possible trading partner configurations. This represents the “strange attractor” (in Complex Systems Science terms) introduced in Chapter 1. It will be argued that, ultimately, this is antithetical to the creation of a GEM; despite underlying determinism, the structural complexity of B2B-collaboration – from a GEM perspective – has introduced randomness into the predictable-structure of a (conceptual) GEM which equates to chaos.

Throughout the chapters leading up to the proposed framework, it will be attempted to identify patterns within contributory aspects of the envisaged-GEM. In many cases, the number of patterns (in “fuzzy sets”), and hence the complexity, will be reduced – in order that the emergence (gestalt) of all individual aspects will be a structured, manageable, integrated whole.

In this chapter the phenomenon of B2B-collaboration will also be analyzed and the “strange attractor” will be introduced.

B2B-Collaboration Defined

One can view “collaboration” as describing the behaviour-patterns between two or more entities in a relationship; the collaboration can manifest in a varying number of occurrences, each of varying duration and of varying quality. In terms of businesses collaborating with each other, one can assume the behaviour to be based on complementary business patterns (such as purchasing-selling), which can occur once or many times, which can occur in a single exchange (of data) or require many exchanges, and which can be successful or not.

Types of exchanges can include: the shop-front (where a seller simply offers goods at a fixed price), iterated bargaining (with buyer and seller taking turns to exchange proposed agreements); the English auction, the Dutch auction and the Contract Net; the continuous double auction and the call auction (Trastour et al, 2003). Although these are typically B2C models, they can be found in B2B scenarios as well.

It is thus obvious that variations in business processes and related business documents are numerous. Further, systems involved in B2B-collaborations my be comprised of innumerable permutations of hardware platforms, operating systems, programming languages, transport protocols, messaging protocols, security protocols, and so forth. This emphasizes the complexity of B2B-collaboration and the seemingly-insurmountable task of *achieving integration (linking of systems) on a global market-space level*. Even though one can argue that integration is a matter-of-degree and does not require total uniformity, a threshold degree-of-commonness is required for integration between each two participants. One may suppose that the greater the depth of integration (linking at various virtual-machine layers), the greater the prerequisite degree-of-commonness. This situation is exacerbated by various configuration-items being dependent on others (with the “structural distance” approaching 1); this increases the structural complexity of the system.

This view is corroborated to some extent by Fastwater (2003). He defines B2B-collaboration as consisting of “interoperation” and “integration” between both internal and external enterprise applications. He states that B2B-collaboration is generally complicated by issues such as scalability, volatility (dynamism), autonomy, heterogeneity, and legacy systems. Typically, differences in data-representations between collaborating systems, differences in interfacing-mechanisms between front-end and back-end systems, and differences inherited from proprietary or legacy data sources are further issues that complicate B2B-collaborations (Medjahed at al, 2003).

These are factors that contribute to the multiplicity paradox (and ensuing “chaos”) discussed later.

B2B-integration can, therefore, be defined in terms of the degree-of-connectedness, which in turn is a function of commonness, between Web-based collaborating systems. One can infer that commonness, therefore, determines the degree of interoperability between collaborating systems. “Integration” and “interoperability” are highly significant terms within the problem-context and will be explored further in the next section.

Key Concepts in B2B-Collaboration: Integration and Interoperability

In physical terms, electronic integration refers to the degree to which organizations are linked by their software and hardware. The efficacy of electronic integration is of supreme importance if the global market-space vision is ever to be realized.

One can argue that the *state* immediately preceding integration describes interoperability; it is the state of readiness-to-integrate. *A state of interoperability therefore precedes a state of integration.* When two entities are *able to inter-operate*, they are ready to integrate. Obviously, the attainment of a final “state of integration” is relative. Interoperability may be considered an *attractor* in the dynamics of the GEM. Various fluctuating pre-interoperability states may be identified.

Pattern

GEM-integration presupposes a state of *global* interoperability. Irrespective of the means by which this can be achieved, global intervention (as in, for instance, *global pre-specification*) by a global authority, such as the World-Wide Web Consortium (W3C) is essential.

The International Standards Organisation Technical Committee (as cited in Rayman, 1999) defines *interoperability* as:

“The ability of systems to provide services to and accept services from other systems and to use the services exchanged to enable them to operate effectively together.”

Interoperability thus involves the means (or a set of means) for exchanging services which may originate in disparate systems, but are capable of operating effectively together (inter-operating). Systems do not have to be homogeneous; it is not necessary for the applications to possess the same design or internal architecture. It is merely necessary *to apply some means for ensuring interoperability among them*.

This is the crux of this thesis: finding the (set of) means to ensure appropriate interoperability between all collaborators in a global multilateral B2B-integration scenario. Interoperability is thus the *means* to an *end;* the end is integration.

The critical question is therefore: precisely what are the means (for example, authoritative intervention, structure, standards and technologies) required/available for uniform interoperability to be attained in a global heterogeneous context (the GEM)?

Williams, Whalley and Li (2000) consider the concept of an “*interface*” to be central to interoperability, but emphasize that the technical specifications of interfaces need to be *open,* that is, “readily and non-discriminatorily available to all vendors, service providers, and users, and if such specifications are revised only with timely notice and public process” (Eurobit-ITIC-JEIDA Chapter, January 1995, cited in Williams et al, 2000). Interface paradigms enable interoperability and are central to solving the integration problem.

Pattern

It, therefore, seems essential that applications supplying services (providers) and the applications requesting the services (consumers) implement a set of specifications for *a common, standardized application-interface,* which enables the accessibility of services. In the global market-space context, these specifications must be universally usable and universally accepted as standard; for example, a W3C standard.

Figure 2.1 below illustrates the integration between two disparate application systems in a GEM-context. The common application-interface provides interoperability and hence allows integration at the application level. The state of interoperability – in terms of the entire GEM "system" - is, however, determined by a number of (interoperability) state variables, as noted earlier. An obvious variable to consider is security. Security, in itself, is a multifaceted determinant of interoperability. Security services such as privacy (confidentiality), authentication, access control (authorisation), integrity and non-repudiation, can be implemented in a number of diverse ways.

Pattern

Many variables, such as a common application interface and security, contribute to the state of interoperability.

This will be explored further in subsequent sections.

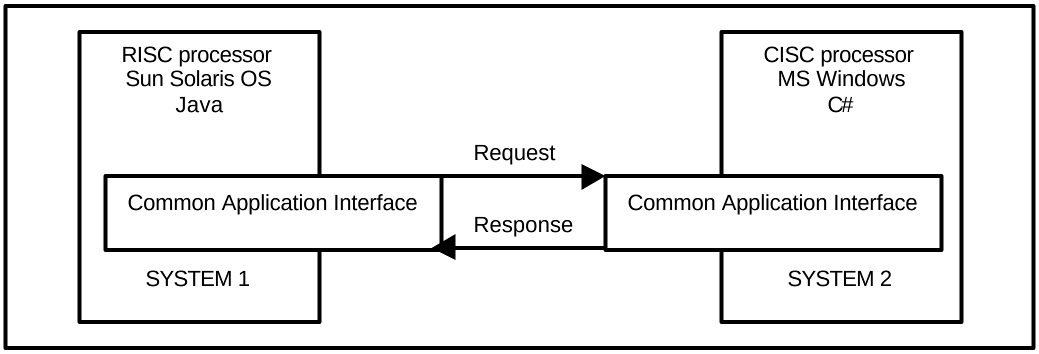


Figure 2.1 Disparate platforms require a pre-specified common application-interface for integration

In the next section, the "essence" of inter-business collaboration is examined.

An Analysis of Inter-Business Collaboration

This section analyzes inter-business collaboration in its raw, native, essential ("ontic") form, bereft of ancillary characteristics (such as business models, technologies, and so forth) which "add colour" to its existence. In its "*a priori*" state, there can be no reference to "B2B-collaboration" which presupposes the use of Internet technology; hence the reference to "inter-business collaboration" in this section.

This is necessary, in view of the discussion in Chapter 1 on *sensitive dependence on initial conditions* as a key element of chaotic behaviour. One needs to assess if the envisaged-GEM, which emanates from simple inter-business collaboration, is *structurally predictable*. Unlike classical Mechanics in Physics (for example), for which the underlying (immutable) determinism can be exposed through formalisms, inter-business-collaboration is a human-contrived phenomenon; as such, a deterministic structure can be imposed upon it, at least to some extent. However, before contemplating such structure, the essence of inter-business-collaboration must first be clarified.

Choudury (1997) contends that the business strategy which motivates inter-organisational information systems (equated to inter-business collaboration in this discussion) is either *competitive*, for example, by interacting with an organization in order to woo it away from a rival company; or *cooperative*, in which case it may be either a *strategic alliance* among a few selected organizations, or a *public good* which is open to all organizations. This obviously has implications for how existing inter-business-collaboration configurations are selected (in prior trading partner agreements). These strategies are observable as three distinct inter-business-collaboration implementation types (as derived from the work of Choudury, 1997 and Kumar and Dissel, 1996). These are described as follows.

Multilateral inter-business collaboration: This type is depicted by an m:n relationship between buyers and sellers. See Figure 2.2(a) below. It is characterized by a *pooled dependency*, where participants share and use common resources for example, common databases and common applications, such as is used in airline reservation systems. This type appears to be the obvious "base" type for extrapolation into a common-good, GEM. There appears to be an insignificant sensitive-dependence on initial conditions in this type; the means for implementation and the nature of this relationship-type have evolved quite dramatically. A typical example is RosettaNet, which will be elaborated in Chapter 4. From a Complex Systems Science perspective (of complex systems) this enhances the structural predictability of a GEM.

Electronic Dyads: These are 1:n relationships between collaborators. See Figure 2.2(b) below. Electronic Digital Interchange (EDI) links are a common contemporary example. A *reciprocal dependency* exists between organizations in a bilateral fashion. Typically, modern organizations may work towards designing, developing and delivering a common product. They usually represent joint ventures, which may be long-term or short-term and may be competitive or cooperative (in a strategic alliance sense).

Electronic Monopolies: This is a special case of an electronic dyad - a bilateral relationship, but only one link is established between a buyer and a seller (1:1), as in a single-supplier supply chain. See Figure 2.2(c) below. This is a *sequential dependency*, and represents strategic necessity rather than strategic advantage.

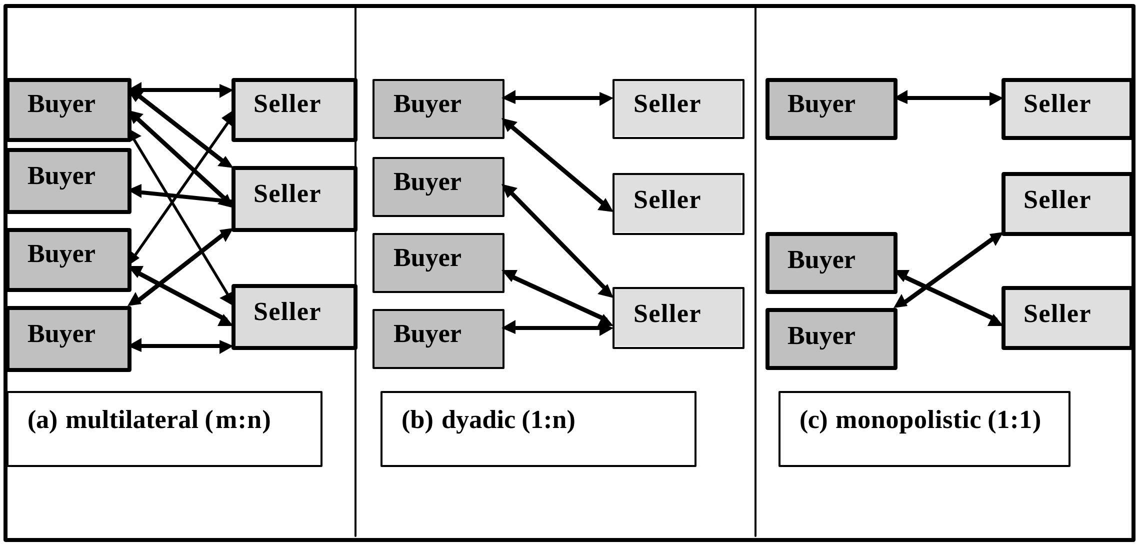


Figure 2.2: Types of inter-business-interactions (based on Choudury, 1997)

Pattern

A multilateral (m:n) inter-business collaboration, minus the common resources, can be considered a “base type” from which to extend the GEM-context. Further, a GEM should be able to co-exist with other (dyadic) collaboration-types.

The nature of inter-business collaboration has led to a diversity of integration paradigms, typically manifested in B2B-collaboration models such as Microsoft’s BizTalk (Li, 2002; www.BizTalk.org), RosettaNet (www.RosettaNet.org) and electronic business XML (www.ebXML.org). (These models will be discussed in Chapter 4).

Having analyzed inter-business collaboration – and the key concepts of integration and interoperability - as well as examining the various “native” forms in which B2B-collaboration can be manifested, one needs to verify that there is authentic incentive for a GEM, before proceeding. Thus, before exploring interoperability in the GEM-context, the next section attempts to first justify (at least in part) the need for a GEM.

How real is the incentive for a global market-space?

This section argues that contemplation of a global market-space infrastructure is valid. The following factors, based on Kumar and Dissel (1996), have been posited as motivation for the implementation of inter-organisational information systems. One can infer that they are valid for a GEM as well, since a GEM is merely a multilateral inter-organisational system of global proportions.

Globalization: This refers to the trend to expand trading world-wide (the “global village” concept). The Internet is the primary facilitator of this trend, and this is the primary motivation for a GEM. The newer tools for the discovery of trading-partners together with new collaboration platforms, for example, ebXML (elaborated at length in Chapter 4) are intended to buttress this drive. A GEM has the intention of, ultimately, enabling businesses of all types, location and size to participate “seamlessly” in a globally-integrated B2B-network.

Environmental turbulence: Business process re-engineering is driven by ever-evolving technologies; companies both want to be competitive and to be seen to be using cutting-edge technologies. The emergence of XML, SOAP and Web services (to be elaborated in Chapter 3) has enhanced the possibilities for integrating highly disparate systems more than any previous technologies. An increasing number of companies are gravitating towards these technologies. Forrester Research (cited in Knorr, 2003) indicates that 85% of their Fortune 500 respondents had intended to deploy Web services in 2003.

Service-oriented architectures (SOA's) and Grid Computing (harnessing distributed computing/ processing resources) have received much attention in the popular press recently. More precisely, there is a paradigm-shift towards services-based Application Service Providers (ASP's) and Computing Service Providers (CSP's) ("The Next Big Thing", Economist, 2004, for example). The new OGSI (Open Grid Services Infrastructure) heralds a new era for Web services-based SOA's (Czajkowski et al, 2004). Grid Services involve the provision of services through the combined efforts of Grid partners across the Internet, and is generating a great deal of interest on the Information Technology front. Again, the degree of collaboration expounded by these trends would increasingly rely on a standardized global market infrastructure (for m:n collaboration).

Resource-pooling: Business partners can share resources in order to lower costs; code-reuse pools, as in utilizing common Web services and related UDDI (Universal Description and Discovery and Integration) Registries (http://uddi.microsoft.com) would provide avenues for resource-pooling. Grid Computing will probably epitomize this strategy, especially if the envisaged pay-per-use implementation comes to pass (Czajkowski et al, 2004). In the light of global interoperability (as envisaged in a GEM), accessibility of common resources will become more plausible.

Risk-sharing: Business processes and resources can also be shared in order to reduce risk; the developing of standards is typically an interest-group effort with participants originating globally, thereby reducing the risk of failure. Again, global interoperability implies that collaborative endeavours will be enhanced.

Reducing supply-chain uncertainty: Product availability and fulfilment procedures have become more effective with online communication. Global m:n relationships based on pre-specified standards would ensure more competitive services in this regard; a greater number of suppliers can be used (with seamless integration of systems).

The most significant incentive for a GEM is *the integration of heterogeneous systems across the globe* by means of a globally uniform mechanism. Further motivation for a GEM can be found in the potential benefits promised by a GEM, such as the standardisation of business patterns, standardized business semantics, a standardized set of business documents (based on a standardized metadata nomenclature), a standardized messaging format and message-delivery system, a standardized message handler, and a standardized security model. Discussing these benefits would, however, be pre-empting the model to be proposed in Chapter 5.

It may therefore be presumed that sufficient impetus towards a GEM-context (even as an adjunct to the existence of other forms of B2B relationships) certainly does exist.

The next section examines the main *obstacle* to (and the main motivation for) achieving GEM-integration.

The Multiplicity Paradox

As mentioned earlier (section 2.2.1), a great number of permutation-possibilities exist for each individual computer-system (hardware and software) that can participate in B2B-collaboration. Configuration-items for each participating member's system include the processing platform (CISC versus RISC processors, for example), which often determines the operating system platform; the network protocol(s) (for example, IP, IPX, and so forth), the transport protocol (for example, TCP) and the application protocol (for example, HTTP, FTP or SMTP). In some cases, the configuration goes beyond simply selecting the protocol(s), as with, for instance, the security protocol(s). The protocol chosen could be IPSec, or SSL/TSL, or PGP, and so forth. IPSec, for example, could be configured for the use of digital certificates, choice of encryption protocol, tunnel mode or transport mode, and so forth. There is obviously implicit multiplicity in the possible configurations of individual participating systems. All configurable parameters listed above will be termed *configuration-items*, for simplicity. These are simply *interoperability state-variables* discussed earlier. They may also be referred to as *properties* of an entity.

The following argument is adapted from Nagel (1974: 277-335). Suppose S represents the set of computer-systems collaborating in a GEM (isolated from the influence of other systems). Assume further that members of S have certain properties (configuration-items) falling into a definite class K. Members of S have properties other than those specified in K, but these are ignored, provided that there is common agreement that K is adequately specified. Let us assume that the values of the properties in K possessed by members of S at any given time define the *state* of S for that time. Relative to the stated class of properties of K, the states of S can be made deterministic for any initial time (t_0) or time interval (t_1-t_0).

Pattern

Assume that the state of S describes GEM-integration. Thus, relative to the set of specified configuration-items (interoperability state-variables), K, the behaviour of the system described by S (GEM-integration) can be predicted at any instant in time. By identifying the interoperability state-variables for K, one should therefore be able to make S deterministic.

However, this is merely a *theory* of imposing determinism on a system of known interoperability state-variables. Simply identifying these variables is not sufficient to achieve integration.

Assume that a set of general statements, L, has been established such that, given the state of S at some initial time, a unique state of S for any other time can be deduced with the help of L. L may be considered a deterministic set of *laws* for S relative to K. (A law might be, for instance, that Advanced Encryption System (AES) be used for all symmetric encryption). *If the number of variables (configuration-items) needed to specify the state of S is very large, it will not be practically feasible to describe the state.* In fact, it is unlikely that L can be established in this case. Therefore, the total set of predicates designating the properties in K must be definable in terms of a relatively small number of mutually-independent predicates belonging to the set. The set of laws L, therefore, constitute a deterministic set of laws for S relative to K, if given the sate of S at any initial time, the laws L logically determine the unique state of S for any other time (Ibid).

At a practical level, in order to integrate each two systems, existing B2B-collaboration models adopt one of two mechanisms. The disparate configuration-items are reconciled – either in an "out-of-band" antecedent collaboration-partner agreement or algorithmically in real-time – to produce a single mutual “interface” (interoperability state). If entire interacting systems were considered, computing the Cartesian product of state-variables for each binary (1:1) inter-business relationship, the number of ordered pairs of corresponding configuration-items could be quite considerable. The number of variables in K, and hence, L, could become unmanageably large (in terms of having to reconcile the differences, as mentioned). In an m:n GEM context, the number of permutation-possibilities would certainly ensure that K and L are unmanageably large, thereby increasing the *structural complexity* of the ensuing system; it is quite likely that L could not be established in this scenario. Therefore, the system cannot have a predictable structure. *Structural predictability* in an environment (such as a GEM) implies that, knowing patterns in one part of the environment allows one to predict patterns in other parts of the environment, at each time, as well as over time (based on Goertzel, 1994:21). Increasing structural complexity thus (intuitively) seems antithetical to structural predictability.

Recall from Chapter 1 that a *pattern* (according to Goertzel, 1994) is a “representation as something simpler” and typically refers to a process. (In the context of a GEM, a typical pattern would be the selection of a messaging protocol; it would be perceived as a pattern in the integration-system). Further, the *structure* of an entity may be defined as the set of all patterns in that entity. The total amount of structure in an entity determines the *structural complexity* of the entity. Patterns too can constitute state-variables (in K) and are also governed by laws (in L); hence, patterns and properties are considered as being equivalents for argument purposes.

<u>Pattern</u>

Thus, a particular law in L could be *pre-specified* to determine the selection of the messaging protocol (for instance), thereby obviating the inclusion of other messaging protocol variables, *thus reducing the size of K.* Other laws governing other messaging protocols would also become superfluous, thus *reducing the size of L.* The smaller the size of L, the more predictable the structure of the GEM

The *intensity* of a process in an entity is a measure of the degree to which it simplifies the entity; it is the ratio of the complexity of the process to the complexity of the entity (Goertzel, 1994:15). The range of possible configuration settings for IPSec (for instance) in each entity (individual participating system) is quite diverse. Each configuration (e.g. setting tunnelling mode) might be of low intensity. Other patterns, such as the algorithm for polling in the Listener, might of higher intensity. A rule of thumb is: *The greater the number of patterns and the greater the variation in intensity between patterns, the greater the "fuzziness" in the emergence.*

In general, the *emergence* between two entities A and B may be defined as the set of all processes that are patterns in the combination of A and B, but not in either A or B individually (Ibid:20). In a dyadic relationship, each entity (participating system) will comprise a "*fuzzy set*" of processes of varying intensity. Thus, the emergence between the two entities is of necessity a "fuzzy set", with "fuzziness" increasing as the variation in intensity between patterns increases. *The emergence (gestalt) in the GEM should thus be "optimized" in terms of reduced "fuzziness" by predefining individual interoperability state-variables for all participants.*

Pattern

It seems reasonable therefore, that one should "backward-engineer" the emergence to reduce fuzziness in the emergence. Laws for the GEM should be formulated to reduce supernumerary configurations and configuration-settings, e.g. "The messaging protocol MUST be SOAP and only the standardized SOAP-envelope MUST be used for messaging". (The SOAP-envelope would have pre-specified configuration-settings).

Unfortunately, current B2B-collaboration models present an ostensibly "flexible" mode of collaboration by (for instance) being protocol-neutral, allowing a range of protocols, with individual configuration-nuances. Examples of this are the ebXML's Collaboration Protocol Profile (CPP) and RosettaNet's Implementation Framework (RNIF) which may be pre-specified by each collaboration partner.

This certainly promotes flexibility within the context of bivalent relationships, but allowing multiplicity in protocol-choice is a limiting factor for integration in a GEM context. The relatively smaller numbers of participants can negotiate common terms of agreement for bivalent relationships, but each participant might have to configure a different protocol-profile for each trading partner it intends to interact with. As the number of trading-partners increases, so the number of possible combinations of protocol-profiles increases exponentially. The emergence (from the m:n relationship) thus comprises many “fuzzy sets”. Thus, interoperability in a GEM-context would, paradoxically, be stultified, since isolated pockets of discrete relationships would proliferate on an ad-hoc basis. (As mentioned earlier, K becomes very large and L becomes difficult to establish). From a GEM perspective, there is apparent randomness emergent from underlying determinism. This is the definition for (an intuitive concept of) chaos (Ibid:21). *Dyadic-interoperability* (the state achieved by providing the means to enable mutual collaboration in 1:1 or 1:n relationships, and thereby allowing integration of the participating systems) therefore, propagates a *multiplicity-paradox.* In attempting to eliminate differences, the “flexibility” of choice allowed actually engenders pockets of diversity.

An *attractor* – introduced earlier - is a region of the space of possible system states (a subset of S) with the property that: 1) states “sufficiently close” to those in the attractor eventually lead to states within the attractor 2) states within the attractor lead immediately to other states within the attractor. Recall the notion of interoperability being an attractor, in this sense. Now consider the notion of dyadic-interoperability as being a “strange attractor” in the envisaged-GEM. A “*strange attractor*” is defined as a “complex region” in which the system fluctuates between states (delineated by this region) in a chaotic manner. However, to a large extent, this behaviour imposes a rough predictability on the system; it steers it in a particular direction (Ibid:21-23). Thus, *the dynamics of the structure of the envisaged-GEM are determined by the patterns in its strange attractor (dyadic-interoperability).* Despite the chaotic, random movement within the strange attractor, there are discernible patterns. In fact, dyadic-interoperability is a highly-patterned strange-attractor, similar to the Freeman olfactory cortex example quoted in Goertzel (Ibid:22-23), in which an initially chaotic fluctuation of a smell-perception progressively settles down to a smaller set of “recognition” states, and eventually selects one pattern.

It is contended here that current models for B2B-collaboration have proliferated more support for dyadic- and monopolistic-interoperability than for GEM-interoperability.

ebXML (discussed in Chapter 3) is described as a suite of specifications for providing interoperability in a "global marketplace". However, it makes provision for a Collaboration Partner Profile (CPP) which fosters discrete collaboration pockets and, hence, would increase structural complexity in a GEM. Trading-partners may specify, inter alia, security, messaging and transport protocols in a CPP. Potential trading-partners then negotiate a Collaboration Protocol Agreement (CPA), which represents a reconciliation of their individual CPP's. Even company-specific business process specifications may be submitted to the ebXML Registry/Repository, adding to the patterns in each entity's "fuzzy set".

It is further contended here that, in terms of establishing a GEM, this approach can be considered *laissez faire*. What is required is an authoritatively-prescribed set of specifications (L, as described above) for m:n relationships in a GEM, which need not be mutually-exclusive with non-GEM arrangements. In a GEM infrastructure, these multiplicities would have to be resolved to a global set (K) (see Figure 2.3). The actual contents of K and L - a few possible "candidates" are indicated in the figure - will be explored in subsequent chapters.

Pattern

A global initiative, in which a set of open standards (state-variables in K) is unequivocally specified as a single standard for global interoperability, must be undertaken by an authoritative body such as the W3C. *Wherever possible, specific values and behaviour for the variables in K (configuration-items and patterns), such as specific protocols, must be unequivocally specified (as laws in L).* A common interoperability interface should allow access from any system; equivalent to any system with a browser being able to download and interact with HTML-based Web pages.

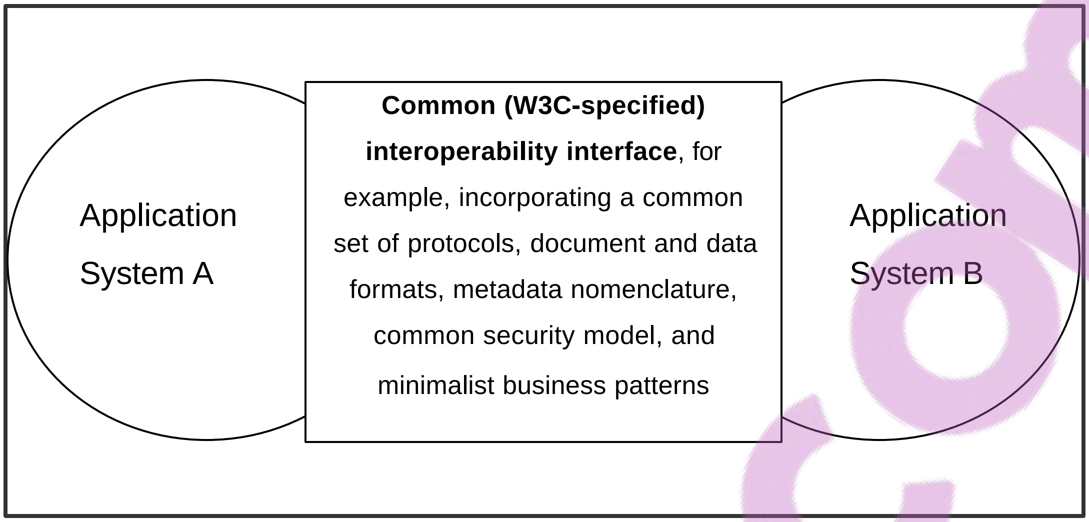

Application System A
Common (W3C-specified) interoperability interface, for example, incorporating a common set of protocols, document and data formats, metadata nomenclature, common security model, and minimalist business patterns
Application System B

Figure 2.3. Lowering GEM complexity with a standard common interface

Conclusion

In this chapter it is argued that powerful impetus exists for the realisation of a global multilateral electronic market infrastructure. Crucial in this regard is the problem of interoperability: making it possible for disparate systems to interact fluently. An obvious panacea would be to use a pre-specified set of protocols comprised of selected globally-accessible open standards. However, in the status quo, B2B-collaboration implementations comprise protocol-neutral technologies and standards that propagate unwieldy permutations of possible trading partner configurations, which, ultimately, are antithetical to the creation of a global electronic market-space.

Allowing diversity in the choice of protocols is a limiting factor in cross-integration. It facilitates the relatively smaller numbers of participants to negotiate common terms of agreement for bivalent relationships, but each participant might have to configure a different protocol profile for each trading partner it intends to interact with. Thus, interoperability in the wider global multilateral market-space context would be stultified, since isolated pockets of discrete relationships are proliferating on an ad-hoc basis. It has been argued here that the "multiplicity paradox" has spawned non-determinism, from the GEM-perspective. Unless a global initiative in which a set of open standards is unequivocally specified as a single standard for global interoperability, a GEM will remain a pipedream.

The successful emergence of the hypothetical-GEM, undoubtedly, resides in interface-technologies. One needs to investigate the application paradigms, standards and technologies available for implementing the required standardized common interface. This will be the focus of the next chapter.

Chapter 3

Arriving at a Standardized GEM Interface

Introduction

Services-oriented architectures (SOA's) are intended to provide application services on demand from a variety of client platforms. They also *provide common application interfaces that mediate interoperability between the requestor and the provider service*. Therefore, the significance of SOA's in B2B-collaboration is examined in the next section.

Earlier SOA mechanisms will be reviewed, briefly, to provide a basis for comparison with newer mechanisms. Thereafter, the use of XML, SOAP and Web Services as supporting open standards for interoperability is explored. It will be verified whether the advent of these standards has significantly enhanced the prospects of making the GEM structurally predictable.

Service-Oriented Architectures

The concept of Service-Oriented Architecture (SOA) has existed for many years. As such, it has a low sensitive dependence on the original state, having evolved with time primarily in relation to the technologies used to implement it.

The concept of SOA, in terms of its objectives and fundamental mechanism of operation, will first be examined. Existing SOA technologies will then be briefly explored to identify pros and cons which have necessitated the evolution of SOA. These technologies include Remote Method Invocation (RMI), Distributed Component Object Model (DCOM) and Common Object Request Broker Architecture (CORBA). All of these provide remote services by way of a stub/skeleton-based architecture. Web Services – as the successor to these technologies - will subsequently be examined to ascertain why it has become the SOA technology of choice. The current related standards for Web services will also be described.

The Essence of SOA

There are many definitions that are used for SOA's. In essence, it is, ideally, a software architecture that exposes a service that *can be discovered*, is *platform-independent* and has a *self-describing interface* (Mysore, 2003). Further, SOA's emphasize *reusable components* - through *loosely-coupled*, *standards-based interfaces* - and application functionality (Yang, 2004).

Pattern

SOA's therefore require minimum configuration by individual participants, in line with the requirements for a GEM.

However, simply making many services available does not constitute an SOA. Discoverability on a network, for example, is an important criterion (Bloomberg, 2004). An underpinning philosophy is that services that are published need to be meaningful and easy-to-use (Cape Clear, 2004). SOA services should also support security, monitoring, interoperability and (open) standards (Yang, 2004).

An SOA encompasses three basic roles: Service Broker, Service Provider and the Service Consumer/Requestor. Further, there are 4 basic steps that occur in the binding of a service to a client requesting it (Systinet, 2001).

The Service Provider first makes the service available and publishes the service contract. The service contract describes the services interface.

1. The Service Provider then registers the service with a Service Broker, which provides "directions" to the location of the service and the service contract.
2. When a Service Consumer requires a particular service, it queries the Service Broker for information on available compatible services.
3. The Service Consumer binds the client (application) to the service using the service contract and the endpoint reference of the service.

Figure 3.1 illustrates this procedure.

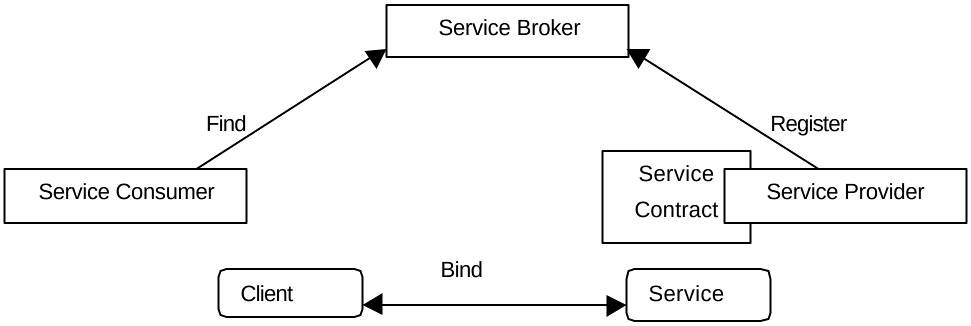


Figure 3.1. SOA: Steps in the binding of a service (Systinet, 2001)

The general short-comings (not specifically in relation to a GEM) in existing SOA-technologies are considered in the next section.

Short-Comings in Existing SOA-Technologies

Earlier SOA approaches had inherent problems, which are widely documented, for example, proprietary technologies, tightly-coupled communications and fine-grained interfaces.

<u>Pattern</u>

Proprietary technologies (as opposed to vendor-independent, open technologies) restrict broader integration through vendor-specific conceptual frameworks, protocols, interfaces, available libraries, and run-time platforms (based on Bloomberg, 2003). Fine-grained interfaces imply greater "fuzziness" and increased structural complexity. This, together with tight-coupling, increases the reliance on maintaining state, which is contrary to the nature of the Internet.

One needs to examine the "sensitive dependence" of current SOA paradigms on the original paradigms. Earlier SOA's (RMI, DCOM and CORBA) will therefore be considered in the next section.

Stub/Skeleton-Based Architectures

The stub/skeleton architecture is intended to allow access to a remote service as if it were a local one. Two modules are needed to build a stub/skeleton architecture: a stub on the client side and a skeleton on the server side. The skeleton exposes the available methods or services, via the stub, to the client. The stub “is aware” of the services available on the server and acts as a translator between the client and the skeleton. The skeleton marshals communication between the stub and the server (Potts and Kopack, 2003). The interface communication layer is provided transparently by the corresponding SOA infrastructure. This process is illustrated in Figure 3.2.

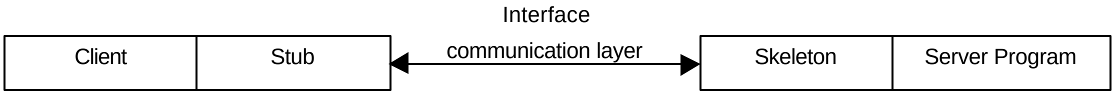


Figure 3.2. Stub/skeleton architecture (Potts and Kopack, 2003)

In a GEM-context, the remote procedure call (RPC) will typically be made across business-application boundaries. This implies the full gamut of configuration-items. Is there a stub/skeleton SOA available that adequately addresses this challenge? CORBA, RMI and DCOM will be examined vis-à-vis stub/skeleton-based architectures in the next section.

(a) CORBA

Figure 3.3 below illustrates the fundamental architecture of CORBA. Unless specifically indicated, this summary is derived from the works of the Object Management Group (2004), Keahey (2004), (Potts and Kopack, 2003), Raj (1998) and Mahmoud (2003).

CORBA is considered in the context of the envisaged-GEM because it is commonly favoured by Java programmers developing on open-source platforms. The interoperability framework upon which it is based also has implicit value in this regard.

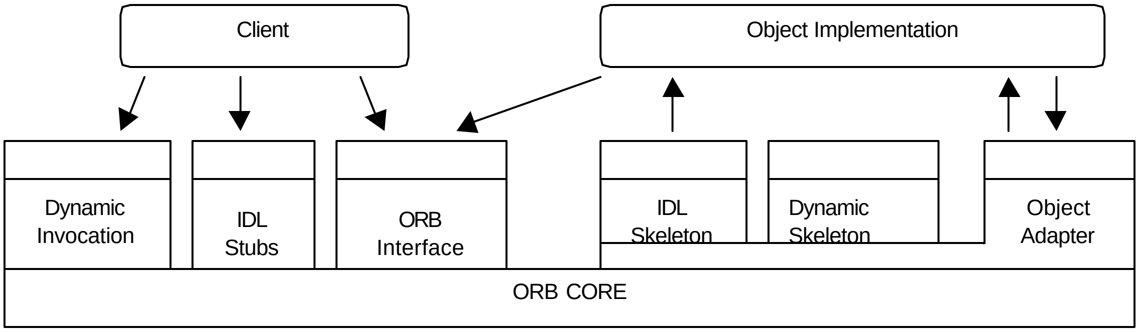


Figure 3.3. CORBA Architecture (Keahey, 2004)

The Object Request Broker (ORB) is the central component of CORBA; it allows each CORBA object to interact with another CORBA object, locally or remotely. The ORB Core provides the communication interface. It selects the object-implementation corresponding to the request, prepares it to receive the request and transfers the data between the client-requestor and the object.

A client uses an object-reference to invoke methods on the actual object. The Interface Definition Language (IDL) is compiled into client-stubs and object-skeletons, which serve as proxies for the client and server, respectively. The client uses the object-reference which identifies the ORB and the instance to invoke. The invocation passes to the ORB Core through the IDL stubs or the Dynamic Invocation Interface. The ORB Core then sends the request through the IDL-skeleton or the Dynamic-skeleton to the Object-Implementation. The Dynamic Invocation Interface is used when the client requests an object and its details are not known. The Object Adapter is responsible for communication between the ORB Core and the Object Implementation. It is the means by which the Object Implementation accesses services provided by the ORB, viz. object-reference invocation, security, and activation and deactivation of the object implementation.

One of the two main key features of CORBA is that it was designed to work with different languages; it uses an Interface Definition Language (IDL) to do this. The second is that it allows many vendors to provide ORB’s. Again, while attempting to provide flexibility, this has merely resulted in increased structural complexity in the resulting “fuzzy sets”. Notwithstanding the fact that CORBA allows different languages, developers need to learn IDL to describe interfaces and objects concerned. There is also a *discernible time-lag resulting from converting a language to IDL* and then into another language at run-time. Further, by allowing *multiple vendors to produce ORB's*, some ORB variants are not interoperable while others only support certain languages. The Internet Inter ORB Protocol (IIOP) used in CORBA 2.0 and higher has addressed the interoperability problem to some extent, but the overall structural complexity of CORBA has increased. CORBA also has two main security disadvantages, namely (a) that it requires certain ports to be open so that the ORB's can communicate and transfer data; and (b) that the data transferred is not encrypted and is sent in clear text. Although these problems have also been addressed, industry has already sought other solutions.

Pattern

The lessons to be learnt from CORBA include the following:

A *universal "language"* (with standardized data format, standardized data types, standardized syntax, standardized character-encoding format, and standardized processing for communicating data) between the consumer and the providers is required. *Translation and marshalling should be obviated*, if possible. The specification for the equivalent of the ORB's must be centrally produced (be vendor-neutral).

(b) RMI

RMI is very similar to CORBA in that it also uses a stub/skeleton architecture, needs an open port for communication and also performs client-server calls (Potts and Kopack, 2003). RMI is illustrated in Figure 3.4 below.

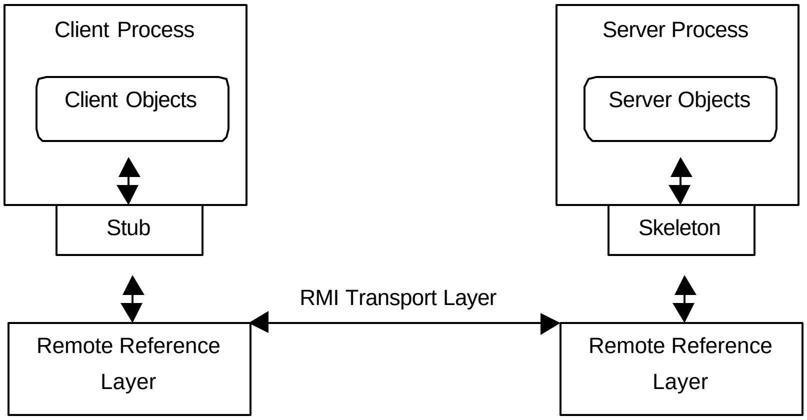


Figure 3.4. RMI Architecture (Flanagan et. al., 1999)

Essentially, RMI operates as follows (Flanagan et al, 1999): The client requests a remote object through the stub, thereby initiating a remote method invocation (RMI) call. The stub represents the remote object being requested and keeps an internal reference to it. The stub then forwards the request through the remote reference layer to the server's remote reference layer. The latter marshals the request to the server, via the skeleton. (Marshalling involves packaging and sending interface parameters across process boundaries). The skeleton passes the request to the server object which performs the method on the object. The skeleton then marshals the response to the server's remote reference layer. The server sends the result back through the RMI Transport Layer to the client's remote reference layer and then to the stub. The stub then marshals the result to the client that made the request

RMI is different from CORBA in that it is written in Java and does not require an IDL to translate languages. However, the RMI architecture can communicate over IIOP. This means that it can communicate directly with CORBA. Nevertheless, *the language-dependence is considered a major disadvantage* in the diverse Internet-environment. RMI allows the developer to include encryption and decryption methods when marshalling and un-marshalling objects. Unfortunately, the problem of *security vulnerability due to open ports* persists in RMI (Potts and Kopack, 2003).

(c) DCOM

DCOM differs from RMI in that the end-points need not be written in a specific language. It is, however, considered a Microsoft-dependant mechanism (Potts and Kopack, 2003). DCOM also performs remote calls. It is based on COM (Component Object Model) which defines how clients communicate with application components. With COM, a COM-interface receives a call or request from a client application, which it forwards to the component. DCOM simply extends the wire protocol that connects the client and component and replaces the local inter-process communication with a network protocol (Microsoft Corporation, 1996). The difference can be seen in Figure 3.5.

While Microsoft claims that DCOM is open-licensed to other software companies (Microsoft Corporation, 1996), it is still *widely considered to work only on Microsoft platforms* (Potts and Kopack, 2003). However, if a platform supports COM services, DCOM can be used on that platform (Raj, 1998). However, a major disadvantage of COM is the infamous *"DLL hell"* problem. This is primarily manifested in the problem of maintaining different versions of a COM component. Except for support for legacy software, the .NET Framework from Microsoft has made COM and DCOM redundant, in favour of .NET-remoting and Web services.

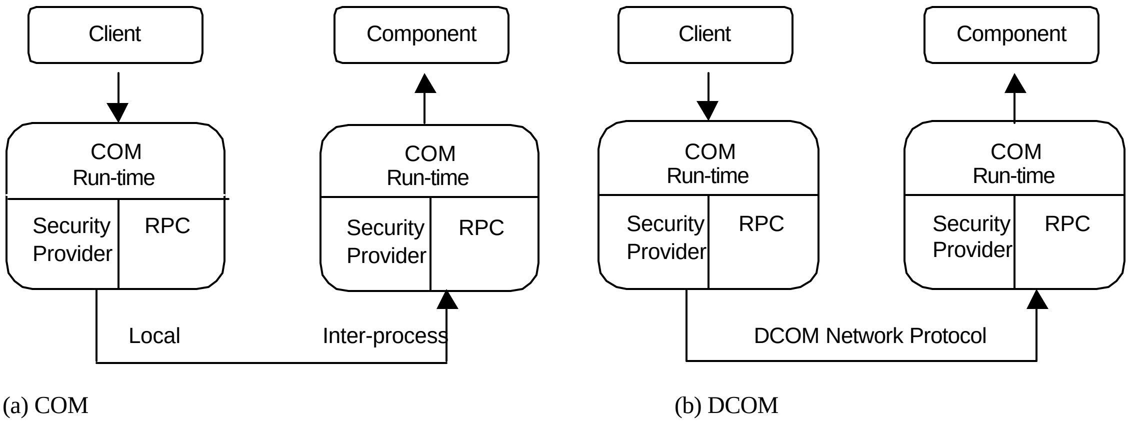


Figure 3.5. COM and DCOM Architectures (Microsoft Corporation, 1 996)

Pattern

The following benefits, attributed to the SOA models above to a greater or lesser degree, represent important characteristics to be sought in an SOA for a GEM: *code-reuse, independence-of-location, independence-of-language, connection-management and scalability.*

In attempting to address the problem of a common application interface, which is accessible from disparate platforms (of any kind), and utilizes the open-systems nature of Web-based technologies, SOA has evolved to produce what is potentially the most interoperable solution to date. The technology is broadly referred to as Web services, but has specific defining characteristics. The Web services "family" of technologies is discussed in subsequent sections.

Current W3C Standards in Web Services Interfaces

Only those standards which are considered pertinent up to this juncture will be considered here. Other standards will be incorporated in the progression towards the gestalt.

XML

At this point it can be pre-emptively declared that XML is favoured as a base technology in the B2B-collaboration context; this is common knowledge. It is the foundation upon which Web services have come into being. Its suitability in a GEM-context will be explored in this section.

The XML specification is an open and freely-available document from the World-Wide Web Consortium (W3C). The design goals for XML, as excerpted directly from the W3C Recommendation document (W3C, 2004[2]), are shown in italics below:

1. *XML shall be straightforwardly usable over the Internet*. To create an XML-document, all one needs is a plain-text editor. To view XML one simply needs a browser. No other special software or hardware is required.
2. *XML shall support a wide variety of applications*. XML can be used in practically all applications. It is commonly used for persisting data and for transferring data.
3. *XML shall be compatible with SGML*. XML is a subset of SGML and can be used in the same environment as SGML.
4. *It shall be easy to write programs which process XML-documents.* With languages like XPath, XLink, and Xpointer (Arciniegas, 2001), one can embed commands to be executed by the XML-processor inside the XML-document. The syntax and command set of these languages are relatively simple. Software development IDE's like Visual Studio.NET provide many tools for developing applications which can process XML.
5. *The number of optional features in XML is to be kept to the absolute minimum, ideally zero*. This is indeed the case. XML is extended by way of namespaces and Infosets (discussed later).

6. *XML-documents should be human-legible and reasonably clear*. XML-document structure resembles structs in programming languages. Even for the uninitiated, it is a relatively simple task to grasp the structure of data in an XML-document.
7. *The XML design should be prepared quickly*.
8. *The design of XML shall be formal and concise.* This makes XML an ideal medium for formal specifications.
9. *XML-documents shall be easy to create.* Apart from being able to create XML-documents in a text editor, IDE's like Visual Studio.NET can generate XML-documents (including XML Schema Documents) from data in a relational database.
10. *Terseness in XML markup is of minimal importance.*

These specifications for XML make it, arguably, the most-promising *document interface* for supporting interoperability in a GEM-context.

XML is a markup language designed to define the structure of data. It uses a set of tags to describe elements of data, which can be either simple or complex types. An unlimited set of these tags can be defined by the author of an XML-document to describe data and data structures. XML is a widely-adopted standard due to its simplicity and platform-independence. It separates style and content, thus enabling data to be integrated from various sources. As a data-description language, it is understandable by both systems and humans; being text-based, it is more readable.
(Refsnes, H. Refsnes, S. and Refsnes, J. E, 2002).

XML allows parties to exchange structured data, as stored and used in databases, via the Internet; relational data can be converted to XML and vice versa. It supports and utilizes Unicode character encoding, which enables the exchange and display of various languages throughout the world. As it is a principal technology in prominent information technology companies, such as Microsoft and IBM, its (relative) longevity as a standard is guaranteed. This is further underlined by the fact that the Web services specifications (and the SOAP standard) are XML-based standards; these are highly-significant standards in current IT developments.

Pattern

XML is fast becoming ubiquitous, even in non-Internet applications, as the lingua franca for data exchange. It is the obvious choice for describing data and transferring data in a GEM context. Applications can exchange structured data, in chunks, between heterogeneous systems. Each system merely has to be able to parse the XML-document and process the data as it deems fit.

The following is an example of an XML-document:

```xml
<?xml version="1.0"?>
<purchaseOrder xmlns="http://tempuri.org/po.xsd" orderDate="1999-10-20">
   <shipTo country="RSA">
      <name>Daleen Mathee</name>
      <street>123 Maple Street</street>
      <suburb>Constantia</suburb>
      <city>Cape Town</city>
      <code>8000</code>
   </shipTo>
   <billTo country="Nigera">
      <name>Adewale Ogunke</name>
      <street>8 Paneg Avenue</street>
      <suburb>Yurubah</suburb>
      <city>Lagos</city>
      <code>dc3452</code>
   </billTo>
   <comment>No news of the oil-strike yet?</comment>
   <items>
      <item partNum="872-AA">
         <productName>Diamond-bit</productName>
         <quantity>1</quantity>
         <ZARPrice>148.95</ZARPrice>
         <comment>Confirm this price</comment>
      </item>
      <item partNum="926-AA">
         <productName>Drill Cooling-fan</productName>
```

```
            <quantity>1</quantity>
            <ZARPrice>39.98</ZARPrice>
            <shipDate>2005-03-21</shipDate>
        </item>
    </items>
</purchaseOrder>
```

Figure 3.6. A sample XML-document

XML-documents can be transformed into different formats for visualisation, notably .pdf (Adobe Acrobat reader format), .ps (postscript printer format), and XHTML (for presentation in Web browsers). Therefore, even if a system receiving an XML-document cannot directly process the data contained in the document, it can display the data in some meaningful manner (as a Web page, or as a .pdf document, for instance). A simple order-form could be generated as an XML-document, which could be transformed using XML Stylesheet Language Transformation (XSLT) – a language for transforming XML-documents – and sent to the receiver. The receiver's browser (which contains an XML parser) will then process the XSLT instructions for presenting the data visually.

As mentioned earlier, other related technologies and standards are available for the parsing and programmatic manipulation of XML, including: XPath (XML Path Language), XPointer and XLink. Further, the Document Object Model Application Programmer Interface (DOM API), as used in DHTML (Dynamic HTML), allows an XML-document to be manipulated as an object tree. Another available API is the SAX/SAX2 (Simple API for XML) event-oriented API, originally developed for Java programmers.

Pattern

In its simplest form, the GEM can consist of Service Providers/Sellers providing Consumers/Buyers with forms to complete. The forms would reflect essential business process activities such as completing an order-form. The data entered on the (Web) forms (for example, shopping carts) would be used to generate XML-documents, which could be processed (or simply visualized) by the Provider's system and/or the Supplier's system. Even simpler, XML-forms could be transferred as e-mail attachments.

The use of XML in its fundamental form thus already reduces the number of elements in set K, described earlier. No special configuration-items are required in either entity A or entity B (apart from browsers/e-mail clients and plain-text editors).

Note, however, that although XML represents the chosen "common language" discussed in Chapter 1, if every trading party were to develop its own set of XML tags and data structures, this would exacerbate the randomness implicit in the "system".

XML-data elements and data structure (*metadata*) can be specified in another XML-document, which is either a Document Type Definition (DTD) or XML Schema Definition (XSD); XSD's have, to a large extent, made DTD's redundant. These are documents that are used for describing and validating the content and structure (the metadata) of XML-data (W3C, 2004[2]). Standards bodies, such as the W3C, use XML Schema Definitions (XSD's) to represent standard specifications (that is, XML "Infosets", containing metadata for tags and data structures, related semantics explanations for how to use the specification) (XML Information Set, 2004).

Pattern

Standards for business patterns, business processes, business documents and business coordination, in a GEM, should be pre-specified, as standardized XML Infosets, by a higher authority, such as the W3C. These can be used by participants to create systems with uniform, predictable behaviour and characteristics.

The SOAP specifications and Web Services Description Language (WSDL) specifications comprise such Infosets; as are the WS-* (Web Services standards) specifications. (These specifications will be elaborated later).

Existing B2B-collaboration models (for example, RosettaNet, BizTalk and ebXML) also use XML-documents that can be validated using pre-specified XSD's. However, even when this is done in the context of n:m relationships, it is done on an ad hoc basis, which retards the progress towards a GEM. This will be elaborated in the next Chapter.

Pattern

The metadata and terminology for all the standardized XSD's and other standardized XML-Infoset specifications, in a GEM, need to be standardized. To this end, a *common GEM-nomenclature* - taxonomy and naming system, based on a common semantics and a common vocabulary - is required.

XML may therefore be used as a business document, for specifying the structure of a business document and for specifying various standards, for example, how to physically transfer documents between Consumer and Provider (as in the SOAP specification). Examples of industry standards that utilize XML include: RosettaNet, Open Applications Group Information Standard (OAGIS), Commerce XML (cXML), XML Common Business Library (xCBL) and electronic business XML (ebXML) (Ojala, 2001).

As pointed out, an XML-document can be used as a common interface between two applications; as a means for transferring structured data. *Would it not be more expedient to send an XML-document containing data directly by HTTP, SMTP or even TCP to a particular endpoint (participant application) for immediate processing as in the RPC mechanisms described earlier?* XML has been used to implement such a messaging mechanism, namely, the Simple Object Access Protocol (SOAP). This is a crucial technology in current B2B-collaboration strategies and will be discussed next.

Simple Object Access Protocol (SOAP)

Unless otherwise stated, this section is based on the W3C SOAP 1.2 Specification document, (SOAP, 2003[1]).

SOAP is a lightweight, extensible, XML-based protocol for information exchange in a decentralized, distributed environment. It is used primarily to send data from one application to another and is therefore a messaging protocol. SOAP simultaneously defines a framework for message-structure and message-processing. It also defines a set of encoding rules for serializing data and a convention for making remote procedure calls; that is, it enables one application to use the functionality published by other remote applications, via RPC. SOAP is based on an extensibility model which provides the foundation for a wide range of extensible protocols.

SOAP has been designed to be simple and extensible. Although it defines a message-processing model it does not define any application semantics, such as a programming model or implementation-specific semantics. Although SOAP may be used as a foundation for building complex systems, most features from traditional message systems and distributed object systems are not part of the core SOAP-specification. In terms of Complex Systems theory, SOAP in itself, therefore, does not bring large "fuzzy sets" into the emergence.

A SOAP-message is based on XML and contains the following parts:

- The *Envelope* is the top-level container representing the message.
- The *Header* is a generic container for adding *features* – an important concept to be clarified later - to a SOAP-message in a decentralized manner. SOAP defines attributes to indicate the intended recipient of a particular feature, and whether understanding is optional or mandatory.
- The *Body* is a container for mandatory information intended for the ultimate message receiver. SOAP defines one element within the body for reporting errors.

As Figure 3.8 below illustrates, a SOAP-message may have multiple application-defined blocks within the Header and Body containers.

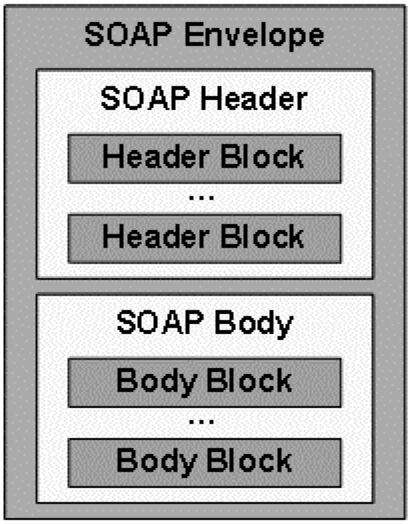


Figure 3.7. SOAP-message structure

The basic XML structure of a SOAP-message is as follows:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xml.soap.org/soap/envelope/ >
        <SOAP-ENV:Header>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
        </SOAP-ENV:Body>
</SOAP-ENV: Envelope>
```

Figure 3.8. Basic SOAP-message XML structure

It is clear that the SOAP-message is simply an XML-document. The first line is a reference to the namespace upon which the Infoset is based; it is merely an identifier. The root element is specified as <Envelope> </Envelope> but is named to indicate the namespace origin; hence, <SOAP-ENV:Envelope></SOAP-ENV:Envelope>.

This is the compulsory extensible envelope for encapsulating data. The SOAP-envelope describes the contents of a message and how to process the message. The SOAP-envelope can comprise multiple separate header and body blocks. Processing information is contained in the header and the payload (data) is contained in the body section.

Pattern

Additional headers (referred to as “modules” or “features”) can be added to reference other XML Infosets, such as existing Web Services standards, for example, WS-S (Web Services Security standard). Web services can therefore utilize the header section for adding standards-based features such as addressing, routing, and security.

SOAP with Attachments (SWA) (SOAP, 2002), now subsumed by Message Transmission Optimization Mechanism (W3C, 2004[1]), enables SOAP-messages to include binary attachments. SOAP is the foundation upon which other related protocols may be built to provide the complete set of services required by a secure and reliable messaging GEM. For example, WS-Security and WS-License may be used to ensure the integrity and/or confidentiality of SOAP-messages. Protocols like these take advantage of the modular packaging of the SOAP-message structure wherein multiple SOAP-based protocols can logically co-exist in the same message. This extensibility allows SOAP to be used in a large variety of systems ranging from small devices to global Web services.

Each SOAP-message is, by default, a one-way transmission. SOAP is not limited to being a single-message request/response mechanism even though its roots are in remote procedure calls (RPC’s). Web services (discussed in a subsequent section) often combine SOAP-messages to implement request/response patterns, but the SOAP specification does not include a multiple message-exchange pattern.

SOAP can be used in combination with a variety of protocols that are able to transport the SOAP-envelope (W3C, 2002, 2002[1]); the primary binding protocols used are HTTP (W3C, 1997) and SMTP (W3C, 2001). If an application is using HTTP-binding, it can be designed to relate a SOAP-message sent in the body of an HTTP-request message to a SOAP-message returned in the HTTP-response. Similarly, HTTP identifies the server endpoint via a URI, the Request-URI, which can also serve as the identification of a SOAP node at the server; the endpoint is inherited by SOAP from HTTP.

The actual SOAP specifications - the latest is version 1.2 (W3C, 2003[1,2,3]) – are fairly lengthy, as are the related Web services specifications and standards. More information can be found on the W3C.org website. However, the basic information provided here is required to clarify the assertions upon which the model proposed in Chapter 5 is based.

Pattern

At this juncture, it is relatively clear that the lingua franca for global collaboration has to be XML/SOAP. However, XML/SOAP per se cannot solve the interoperability-integration problem; it cannot, for instance, handle synchronous multiple message exchange patterns. Although one may infer at this point that if a global SOAP-envelope standard (with specific added standardized modules/features and managed by the W3C) were to be created, specifically for GEM-use, it would go a long way towards promoting global interoperability. It should be based on the envisaged standardized nomenclature.

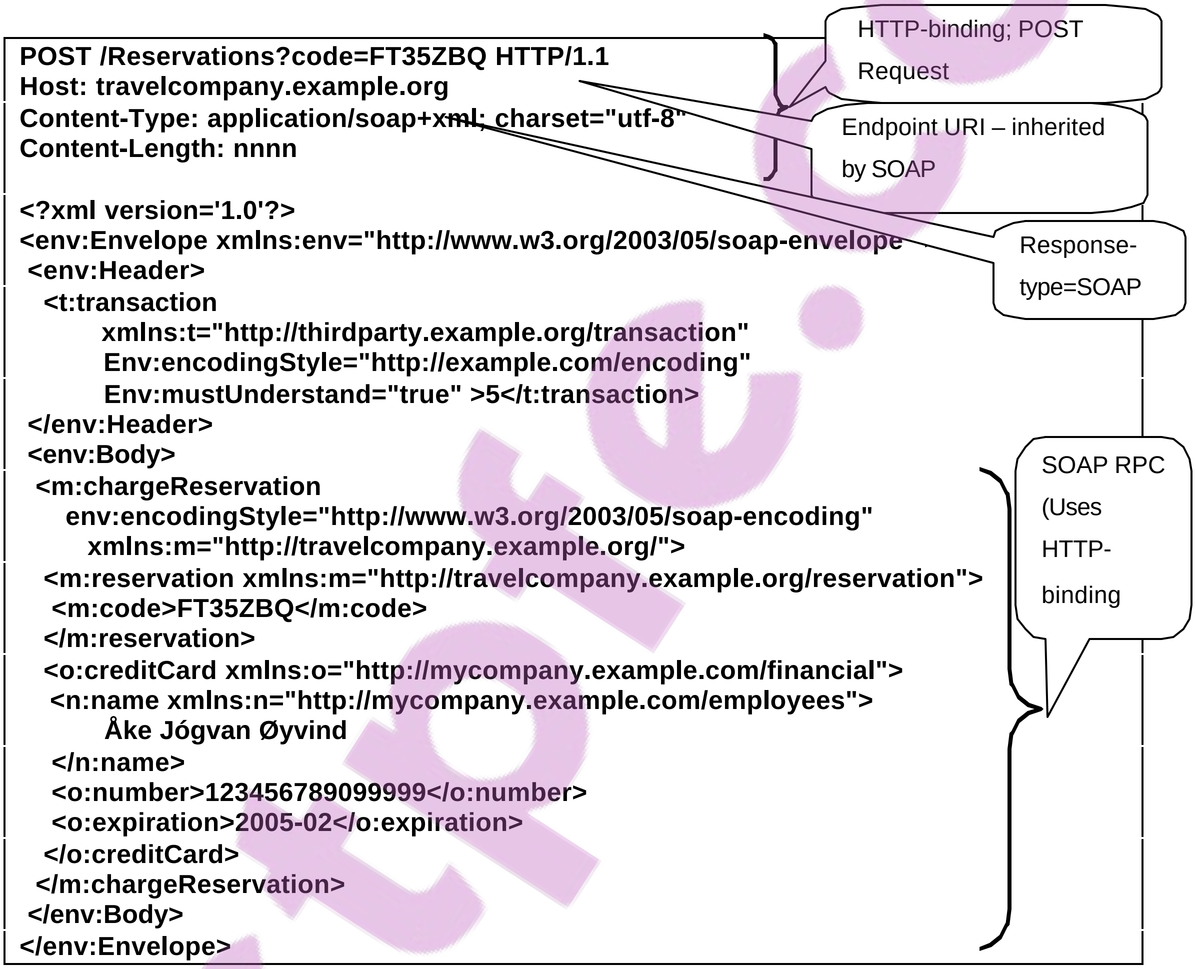
Precisely where and how should SOAP be used? Subsequent chapters will attempt to answer this question, but it is important to realize at this point that even using SOAP does not automatically preclude reverting to a multiplicity-paradox scenario. This is illustrated in the next paragraph.

SOAP 1.1 does not specify how SOAP-messages should be routed. SOAP 1.2 caters for routing through extension, using the WS-Routing specification (a Web services-standard feature). SOAP-messages can be routed between numerous actors or roles in a message path and it is important to ensure that sensitive data is not intentionally or inadvertently compromised or disclosed by one of these intermediaries along the path to the final recipient. There is thus a requirement for security to be persisted in a SOAP-message from message producer to final, intended recipient (O ‘Neill et al, 2003). (WS-Routing has been deprecated by WS-Addressing). The implementation (and configuration) possibilities for ensuring this potentially contribute to the multiplicity paradox alluded to earlier, as a number of possible security-implementation possibilities exist.

Pattern

In the GEM context, pre-specified standards (for example, security and addressing) for SOAP-messaging are crucial. This would be part of the standardized SOAP-envelope specification.

The following figure illustrates an example of a SOAP-message with an HTTP-binding.



```
POST /Reservations?code=FT35ZBQ HTTP/1.1
Host: travelcompany.example.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0'?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope
 <env:Header>
   <t:transaction
          xmlns:t="http://thirdparty.example.org/transaction"
          Env:encodingStyle="http://example.com/encoding"
          Env:mustUnderstand="true" >5</t:transaction>
 </env:Header>
 <env:Body>
  <m:chargeReservation
     env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
        xmlns:m="http://travelcompany.example.org/">
   <m:reservation xmlns:m="http://travelcompany.example.org/reservation">
    <m:code>FT35ZBQ</m:code>
   </m:reservation>
   <o:creditCard xmlns:o="http://mycompany.example.com/financial">
    <n:name xmlns:n="http://mycompany.example.com/employees">
          Åke Jógvan Øyvind
    </n:name>
    <o:number>123456789099999</o:number>
    <o:expiration>2005-02</o:expiration>
   </o:creditCard>
  </m:chargeReservation>
 </env:Body>
</env:Envelope>
```

HTTP-binding; POST Request
Endpoint URI – inherited by SOAP
Response-type=SOAP
SOAP RPC (Uses HTTP-binding

Figure 3.9. Basic (incomplete) SOAP example (W3C, 2003[1])

The listing below shows a sample SOAP-message (that is an alert message to a Web service). Notice the references to different namespaces, which are XSD's, in which the metadata for the tags used, are defined. The first namespace refers to the SOAP envelope specification; all tags "inherited" from this namespace are prefixed by "env:". Similarly, the tags prefixed by "n:" denote elements from another schema specification. This part of the document can be validated against that schema document programmatically (or by using an XML-validator application). The request contains a text message (in the Body) and is marked to indicate that the message is not useful after the given time (in the Header). If an endpoint is indicated in the HTTP header, SOAP makes an RPC to the “<m:alert> </m:alert>” endpoint (Web services method), passing it the text message as a parameter. No security is specified in this example.

```
<?xml version="1.0" encoding="UTF-8" ?>
<env:Envelope xmlns:env="http://www.w3.org/2001/09/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
      <n:priority>1</n:priority>
      <n:expires>2005-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Fetch D from school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

Figure 3.10. A single-message SOAP-request

As emphasized earlier, the concept of an “*open interface*” is central to solving the global marketplace interoperability problem.

Pattern

As SOAP is the best candidate for messaging (transferring XML-data for RPC processing) in a GEM, an open-standards application interface is required to accommodate and extend SOAP functionality. This functionality must operate within a Services-Oriented Architecture, in a superior manner to SOA models discussed earlier.

Do Web services represent this application interface? XML Web services have been specified as a standardized means for exchanging and processing SOAP messages in a platform-independent manner. This is the subject of the next section.

XML Web Services

The significance of Web Services to a GEM

A Web service, in simple terms, is an application providing a service (Wall and Lader, 2002: 220-221), for example, a calculator application that calculates mortgage bond repayments, accessible via standard Web protocols. It, therefore, resides on a Web Server (such as Internet Information Server or Apache), which might be on an intranet, an extranet or the Internet.

In its simplest form, a Web service may be accessed from any platform-type as follows: The uniform resource identifier (URI) of the Web service is entered on a Web browser. The browser automatically generates a local client through which the service can be activated. Input is entered via the client and the output(s) of the service is (are) returned to the browser client. Messaging between the client and the Web service is by way of SOAP. The Web service is invoked by SOAP-RPC from within the message; no stub-skeleton architecture is required. A more comprehensive (thick) client for handling the output (and automated entering of input) can be used in place of the browser. In this case, a client proxy (stub) is required, which is typically generated by development tools such as Visual Studio.NET. The message is simply a more complex HTTP-request than the standard requests from Active Server Pages (ASP), Common Gateway Interface (CGI) and Java Server Pages (JSP); or even HTML-form requests (indicating Action and Method attributes, for HTTP). SOAP-requests use the same HTTP POST method and the Action attribute is emulated by a SOAPAction attribute in SOAP 1.2. In Microsoft’s .NET Framework, a request-response message exchange pattern (MEP) is transparently invoked when a method is denoted as being a Web (service) method, by preceding the method with a standard Framework attribute, “[WebMethod]”, in the C#-language. (If the Microsoft.Web.Services2 libraries are used, then the Web service class can be created as an HTTP Handler and can be denoted by the “[SOAPService]” attribute; the methods can be denoted by the [SOAPMethod] attribute). The application language, the Web server, the operating system, and so forth, are of little or no consequence in the GEM context. The client must merely be able to generate a SOAP-request.

The platform-independence of Web services is illustrated in the diagram below:

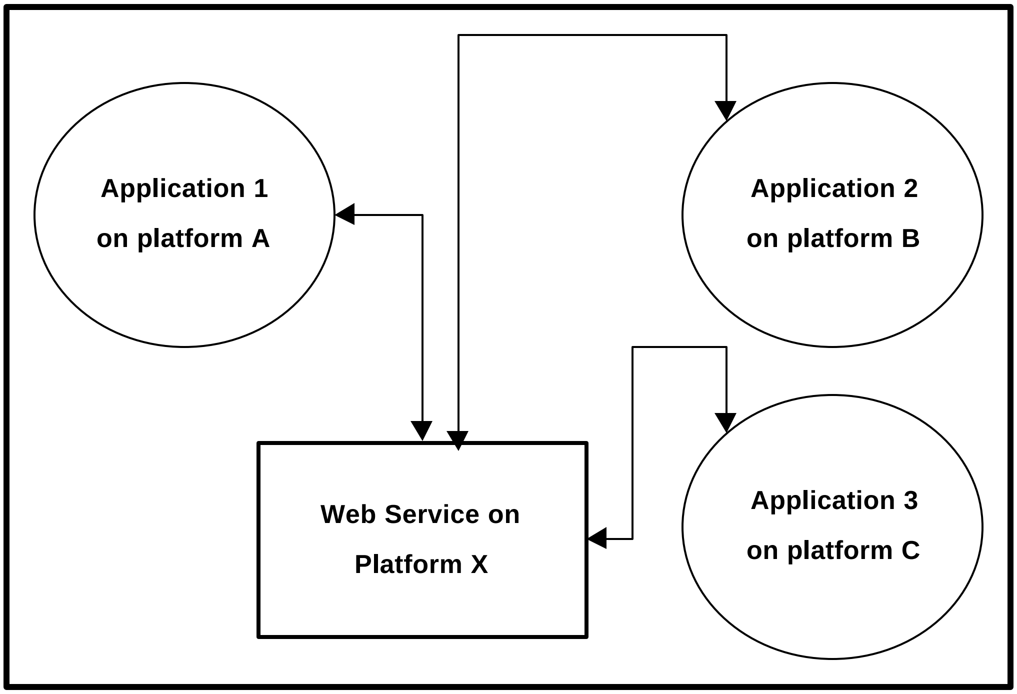


Figure 3.11. Web services: platform-independence and global accessibility

A Web Service, like any other n-tier application, is typically comprised of logical layers as illustrated in Figure 3.12 below. The Data Layer stores the actual data that is used by the Web Service. The Data Access layer is used to present a logical view of the physical data to the Business Layer. The Data Access Layer is divided into two layers, namely the Business Logic Layer and the Business Façade Layer. The

Business Façade Layer maps a simple interface for the actual services (endpoint-methods) on the Business Logic Layer. The client application interacts with the Listener which is responsible for parsing the incoming SOAP-request and calling the appropriate method via the Business Façade. When the service responds, the Listener will then package the response into a SOAP-message and send it back to the client. (Kirtland 2001).

Pattern

The Listener needs to comply with the universal interface standard in terms of the GEM requirements. The Listener could be elaborated into a global message handler (GMH) implemented as a Web service or embedded in a larger Web service.

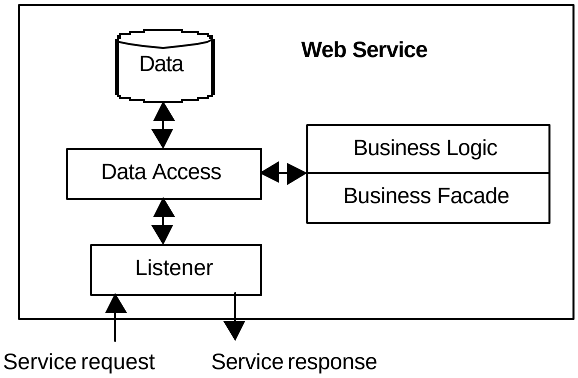


Figure 3.12. A Generic Web Service Architecture (Kirtland 200)

The *major benefits of Web services* as an application interface, in general, can be summarized as follows:

Web services enable Service-Oriented Architecture. Unlike earlier SOA models, Web services are platform-independent, vendor-independent, language-independent, do not require additional ports to be open (except for HTTP or SMTP), are not complicated by IDL-translation or marshalling, incorporate the copious benefits of XML and SOAP, and are easy to implement. By using the SOA model, business processes in an application are separated into services (independent, easily-distributed components); these processes can interact across disparate application and machine boundaries, either within an organization or can be made available over the Internet for use by other organizations.

As Web services utilize XML/SOAP-based messaging, they promote interoperability on both data and system levels; XML is (UNICODE) text-based and can be processed by any computing platform. Web services allow a high degree of abstraction between the implementation and consumption of a service. Therefore, both the provider and consumer systems essentially require only knowing the inputs, outputs and location of the Web service. Web services are loosely-coupled. Thus, each service exists independently of the other services that make up the application, which allows individual modifications; this facilitates development and deployment.

Web Services infrastructure

The W3C's Web Service Architecture Group defines a Web service (more comprehensively) as follows: “A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered by XML objects and supports direct collaborations with other software applications using XML-based messages via Internet-based protocols” (Jenz, 2002). This extends the simplistic definition provided in the earlier section somewhat; a Web service represents more than just an application interface. It represents the most interoperable SOA to date. In the context of a GEM, this is a vital development. To elaborate, Web services use a standard infrastructure that also provides for the discovery and publishing of services over the Internet. In fact, this infrastructure provides for Web service directories, a Web services discovery mechanism, a Web services description mechanism and open-standard wire formats (Brown, 2002).

The Universal Description, Discovery and Integration (UDDI) standard has been designed for implementing directories (registry and query facilities) for Web Services. Additionally, the Web Services Description Language (WSDL) standard supports a Web service description and discovery mechanism. *Web services directories* (UDDI registries) provide a central location where Web services can be published and discovered, using WSDL. Currently Web services may be published ad lib. The *Web* *service discovery* mechanism (WSDL and UDDI) ensures that one or more related documents that describe and contain links to a particular Web service can be located.

WSDL defines an XML grammar for describing Web services in a structured way. The existence of Web services and their location is learned through UDDI. The *service description* is an XML-document based on the WSDL Infoset that defines the format of messages the Web service understands. The service description serves as an agreement/contract that defines the behaviour of a Web service and provides instructions for how potential clients can interact with it. The behaviour of a Web service is determined by messaging patterns that the service defines and supports. These patterns theoretically dictate what the service consumer can expect to happen when a properly-formatted message is submitted to the Web service.

The *service wire formats* provide universal communication through SOAP over HTTP, SMTP, or even TCP. The SOAP-specification specifies all the rules for locating Web services, integrating them into applications and communicating between them. It does not define any application or transport semantics and this allows it to be used in a variety of systems (Ibid).

Web services SOA (WS-SOA) complements object-oriented approaches but it differs in one aspect, namely binding. This refers to the functions that are provided and how they are delivered. Earlier SOA technologies implemented binding using references to the components or objects. A WS-SOA has a mechanism for dynamic discovery of services that provides the interfaces to objects or components (Ferguson et. al., 2003).

The WS-SOA thus has three main components: transport, discovery and description. Transport involves the protocols and formats for communication with a service. Description refers to the WSDL used to describe a service and the information necessary for binding to it. Discovery is the mechanism to allow the service and its description to be found and is implemented using UDDI. The binding can be static or dynamic. Static binding occurs at run time or compile time and the developer specifies the binding with a specific Web service. Dynamic binding is where the developer can use part of the WSDL at compile time so that it can work with a certain service type; at runtime the client dynamically compiles the WSDL and implements the binding.

Even the discovery process can be implemented as a dynamic process (Systinet, 2002). Since UDDI is itself a Web service, an application can accomplish the following at runtime: query the registry, dynamically discover a service, locate its access point, retrieve its WSDL information, and dynamically bind to it. A discovery mechanism is needed if the client does not know where the Web service is. UDDI is a discovery protocol that defines an Application Programming Interface (API) to interact with a centralized Web services repository and an XML Schema definition of supporting messages and data structures. Web services information can be published on a site registry/repository that implements the UDDI specification.

UDDI API commands include the following: To find a service one invokes *find()*, which is a SOAP-request message; this is sent to one of the UDDI sites' endpoints. The SOAP-response message contains the result. The *save()* operation allows one to register or update service information and *delete()* removes the information. User authentication and encryption is required to publish, edit or delete service information. UDDI allows lookup and categorization features. The disadvantage is that it increases complexity and the UDDI site has control of the service; alternatively, developers can implement their own UDDI site (Skonnard, 2002)

DISCO is a Microsoft technology for discovering and publishing Web services. It makes it possible to discover the capabilities of a service using documentation. Using this method to publish a Web service, one creates an XML-document saved as a .disco file, which contains links that describe the Web service(s). To discover services statically, two tools are available; a command-line client and the *Add Web Reference* in Visual Studio .NET. For dynamic discovery, one needs to create a .vsdisco file; this will have the same appearance as the .disco files. Services can be discovered on a given server. A disadvantage of DISCO is that it does not sort information on Web services like UDDI. However, it is simple and effective and allows the developer control over the discovery of the service (Ibid).

In order to optimise the functionality of specific Web services, one can include SOAP features in accordance with existing standards for Web services (known as WS-* standards). IBM, Microsoft and other organizations have compiled a group of Web services specifications (features). These specifications are available from: www.specifications.ws, among many other sites.

Related Specifications

SOAP defines a framework for message structure, but with the exception of SOAP faults, does not define messages-types. (WSDL may be used to describe the message-types a Web service can send and receive). The information below was obtained mainly from www.specifications.org.

The WS-* standards messaging specifications, which have implications for the framework proposed in Chapter 5, are: SOAP version1.2, WS-Addressing, Message Transmission Message Optimisation/ MTOM (Attachments) and WS-Eventing. However, by August 2004, only support for WS-Addressing was available in the .NET Framework through the Web Services Enhancements 2.0 (SP2) for Microsoft .NET (available from the Microsoft website for download). However, an object library for supporting serialized attachments was also available at the time.

As a note, en passant, the following are important changes regarding the WS-* specifications. WS-Routing and WS-Referral have been superseded by WS-Addressing. The Direct Internet Message Encapsulation (DIME) specification, the SOAP Messages with Attachments (SwA) and the Proposed Infoset Addendum to SOAP Messages with Attachments (PASwA) have all been superseded by the SOAP Message Transmission Optimization Mechanism (MTOM) specification. MTOM may be used to optimize transmission of binary data underlying a SOAP-message.

MTOM

MTOM is considered in some detail here as it is significant in the framework proposed in Chapter 5.

MTOM (W3C, 2004[1]) describes a mechanism for optimizing the transmission and/or wire format of a SOAP-message by selectively re-encoding portions of the message while still presenting an XML Infoset to the SOAP application. It describes an inclusion mechanism that operates in a binding-independent way, as well as providing a specific binding for HTTP. The working draft was published on 8 June 2004. MTOM relies on XML-binary Optimized Packaging (XOP) (W3C, 2005). The diagram below (from the W3C specification document) explains the optimisation process. An XOP package is created by placing a serialization of an XML Infoset in an extensible packaging format such as a MIME Multipart/Related package. Then, selected portions of the data that are base64Binary-encoded are extracted and re-encoded (that is, the data is decoded from base64) and placed into the XOP package. The locations of those selected portions are marked in the XML with a special element (xop:Include) that links to the packaged data using URI's. Only element content can be optimized; attributes, non-base64-compatible character data, and data not in the canonical representation of the base64Binary data-type cannot be successfully optimized by XOP. Base64-encoding is necessary for signing with XML Digital Signatures, as well. A comprehensive listing of MIME Content-Types can be found at: http://www.utoronto.ca/webdocs/HTMLdocs/Book/Book-3ed/appb/mimetype.html.

The MIME Multipart/Related content-type provides a common mechanism for representing objects that are aggregates of related MIME body parts. The Multipart/Related content-type addresses the MIME representation of compound objects. The object is categorized by a "type" parameter. Additional parameters are provided to indicate a specific starting body part or root and auxiliary information which may be required when unpacking or processing the object, e.g. 'boundary=MIME_boundary' used in XOP.

The following definitions are obtained (verbatim) from the XOP specification:

- *Original XML Infoset* - An XML Infoset to be optimized.
- *Optimized Content* - Content which has been removed from the XML Infoset.
- *XOP Infoset* - The Original Infoset with any Optimized Content removed and replaced by xop:Include element information items.

- *XOP Document* - A serialization of the XOP Infoset using any W3C recommendation-level version of XML.

- *XOP Package* - A package containing the XOP Document and any Optimized Content. As a whole, the XOP Package is an alternate serialization of the Original Infoset.

- *Reconstituted XML Infoset* - An XML Infoset that has been constructed from the parts of an XOP Package.

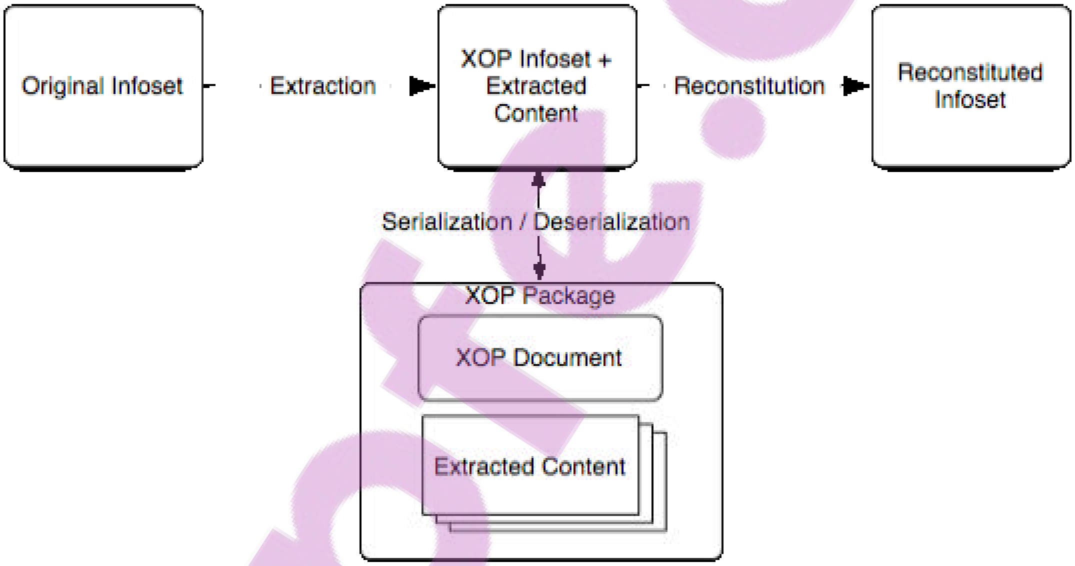


Figure 3.13 The Mechanism of XOP as used in MTOM

Unfortunately, no current implementation support is available for MTOM yet. It is reported on the Microsoft website that Web Services Extension (WSE) 3.0 for the .NET Framework will, however, provide such support. Nevertheless, XML/SOAP encoded as Base64Binary can be used in place of the MIME package.

The code listing below illustrates an example of how MTOM is intended to function.

```
<!A MIME-serialized multi-part XML Document containing a .png image : Before
Optimisation>
<soap:Envelope
   xmlns:soap='http://www.w3.org/2003/05/soap-envelope'
```

```
    xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>
  <soap:Body>
    <m:data xmlns:m='http://GEM.org/InsAssessor'>
      <m:photo
          xmlmime:contentType='image/png'>/aWKKapGGyQ=</m:photo>
     </m:data>
  </soap:Body>
</soap:Envelope>
```

```
<!After Optimisation>
MIME-Version: 1.0
Content-Type: Multipart/Related;boundary=MIME_boundary;
    type="application/xop+xml";
    start="<AssessorReport.xml@InsuranceCo.org>";
    startinfo="application/soap+xml; action=\"ProcessData\""
Content-Description: A SOAP message with picture of damage

--MIME_boundary
Content-Type: application/xop+xml;
    charset=UTF-8;
    type="application/soap+xml; action=\"ProcessData\""
Content-Transfer-Encoding: 8bit
Content-ID: < AssessorReport.xml@InsuranceCo.org >

<soap:Envelope
    xmlns:soap='http://www.w3.org/2003/05/soap-envelope'
    xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>
  <soap:Body>
    <m:data xmlns:m='http://InsuranceCo.org/InsAssessor'>
      <m:photo
  xmlmime:contentType='image/png'>
<xop:Include
    xmlns:xop='http://www.w3.org/2004/08/xop/include'
    href='cid: http://InsuranceCo.org/InsAssessor/damage.png'/></m:photo>
     </m:data>
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/png
Content-Transfer-Encoding: binary
Content-ID: <http://InsuranceCo.org/InsAssessor/damage.png>

// binary octets for png

--MIME_boundary
```

The example – a typical context of an insurance assessor sending a report of the damage to a vehicle, is used - illustrates a multipart XML document (Content-Type=application/xop+xml) containing a .png image, from which the image can be serialized as a MIME-part (Content-Type = “image/png”). The image is serialized as binary octets and an <xop:include> element with an href attribute that points to the location of the serialized file, is added. The original XML Infoset is also serialized as binary octets within the XOP package.

Other WS- Specifications*

WS-Addressing provides transport-neutral mechanisms to address Web services and messages. Specifically, this specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission in a transport-neutral manner through networks that include processing nodes such as endpoint managers, firewalls, and gateways. WS-Addressing was published as a public specification in March 2004 (and is supported in WSE 2 SP3).

WS-Eventing describes how to construct an event-oriented message exchange pattern using WS-Addressing concepts, allowing Web services to act as event sources for subscribers. It defines the operations required to manage subscriptions to event sources, as well as how the actual event messages are constructed.

These specifications are of significance in relation to the envisaged global SOAP-envelope standard, as will be discussed in Chapter 5. Of extreme importance to the success of using Web services in a GEM is the issue of security. This will be discussed in the next section.

Security Challenges in Utilizing Web Services

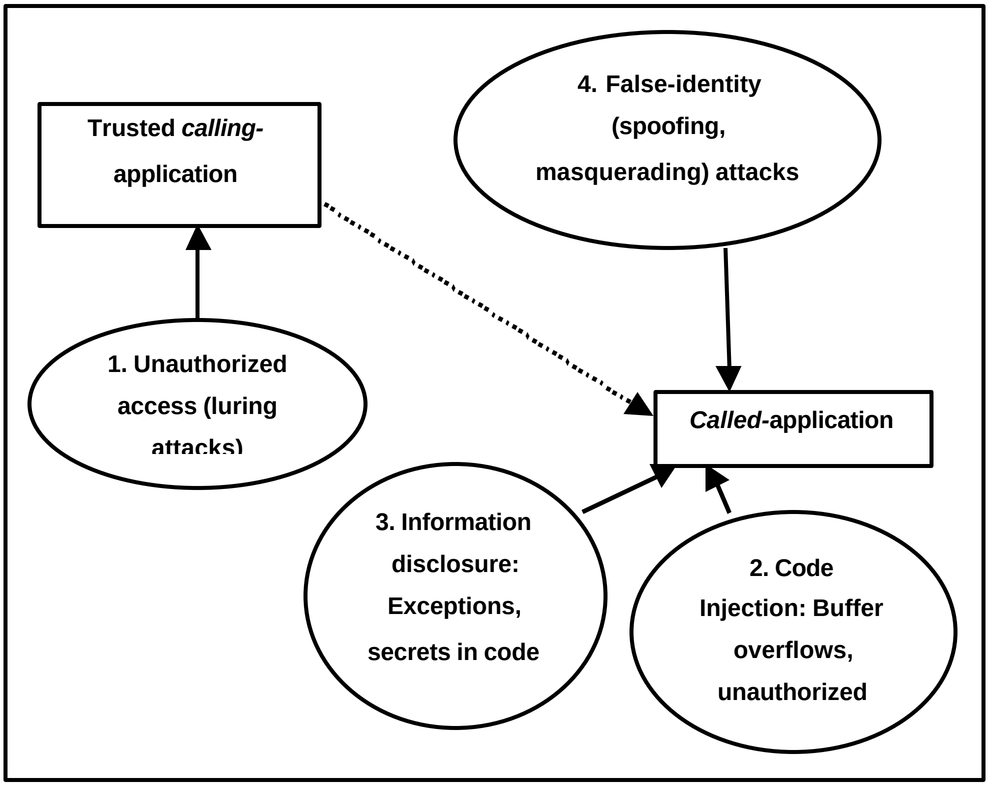
SOAP-based Services, in general, present many security challenges, which include the following:

1. The identity and role of the original service-requestor principal needs to be known by the called SOAP-based Service. Unauthorized access may occur as a result of the called-application being lured into divulging privileged information or executing privileged operations by a third-entity principal manipulating a trusted calling-application (*luring attacks*). Security information (identity, role, and related permissions) is required to be persistent for multi-hop SOAP-messages; it

can be demanded down to the application-component level at each hop. An attacker may also capture and copy a message containing valid security information and attempt a *replay attack* (impersonating a valid client).

2. Applications requiring input from the user are susceptible to *code-injection and parameter-manipulation attacks*, particularly if the input is not controlled and checked for validity (using patterns/masks/regular expressions, and checking data types, ranges and specific values, including SQL-commands). The code injected in lieu of required input can typically be used to cause buffer-overflows, to access privileged data and to execute privileged operations.
3. The runtime within which the SOAP-based Service executes can disclose privileged information as part of an error message. An attacker could, therefore, deliberately force errors in order to *extract information from returned error messages*. Error messages could also reveal secrets, such as passwords.
4. The identity and/or role of a valid principal can be mimicked (in *spoofing and masquerading attacks*) in order to gain access to the SOAP-based Service resource.

The GEM SOAP-based services would be susceptible to these threats, among others. Therefore, the currently available countermeasures for ensuring sufficient security of the SOAP-based services need to be explored.



Trusted *calling*-application
4. False-identity (spoofing, masquerading) attacks
1. Unauthorized access (luring attacks)
Called-application
3. Information disclosure: Exceptions, secrets in code
2. Code Injection: Buffer overflows, unauthorized

Figure 3.14. Examples of threats to typical SOAP-based Service applications

The network-transport pathway between the calling-application (requestor/consumer) and the called-application (provider) also needs to be secured. The following sections examine security open-standards and technologies which can be used in a SOAP-Web Services context.

Web Service Countermeasure Standards and Technologies

The standards and technologies discussed below are intended to provide a brief overview of what is currently available; the discussions are by no means comprehensive. Where no specific reference is provided, information has been obtained from Erl (2004).

Security Assertion Markup Language (SAML)

(Rosencrance, (2002).

When an enterprise environment consists of applications that independently control user access lists, it becomes difficult to administer user credentials. Adding Web services can contribute to the problem. As new integration channels are added, more user-accounts are required. Single sign-on technologies help address this problem.

Popular technologies for single sign-on include the Security Assertion Markup Language (SAML), the .NET Passport and Sun’s Liberty Alliance specification. SAML provides mechanisms for both authentication and authorization processes. Both request and response message formats are defined to facilitate the transmission of necessary credentials within a Web service activity. Microsoft’s .NET Passport is a competing technology, but relies on proprietary protocols for handling authentication. It also introduces a centralized management system for user credentials, which differs from SAML’s decentralized approach. However, interoperability options between the technologies do exist.

SAML is an XML-based framework that enables Web services to readily exchange information relating to authentication and authorization. This information takes the form of trusted statements, called *security assertions*, about end users, Web services, or any other entity that can be assigned a digital identity. There are three major types of SAML assertions.

Authentication assertions, which are issued by an authentication service, declare that the identity of a user or a Web service has been authenticated; the principal may thus have access to protected resources.

Attribute assertions, which are generated by an attribute service, verify that a user or Web service possesses certain attributes. These can be static attributes (for example, job role or company affiliation) or dynamic attributes (such as a consumer’s bank account balance or a reseller’s quarterly sales volume to date). Attribute information can be used in assigning Web service access privileges.

SAML also comprises an authorization service, which resolves authentication assertions, attributes assertions and authorization policies, and generates *authorization assertions* that define which resources a user/service is allowed to access. SAML shields applications from the complexity of the underlying authentication and authorization systems. SAML is already being used for authorization in higher education institutions and on the public Internet.

Thus the main benefit of SAML is as a single-sign-on mechanism; authentication and authorization are persisted during navigation from one procedure/resource to the next.

Extensible Access Control Markup Language (XACML)

(Kay, 2003)

The XACML specification consists of two related vocabularies: one for access control and one that defines a vocabulary for request and response exchanges. Both may be used to create fine-grained security policies.

Griffin (2004) states that XACML can be used to define a general policy language to protect resources, as well as an access decision language. This is considered particularly useful for access control in federated authentication environments.

Olavsrud (2003) states that, according to Sun Microsystems, XACML has a number of advantages over other access control policy languages, including:

- It can replace dozens of application-specific languages;
- XACML is flexible enough to accommodate most access control policy needs and extensible enough so that new requirements can be supported;
- One XACML policy can cover many resources; this helps avoid inconsistent policies on different resources; and
- XACML allows one policy to refer to another; this is important for large organizations, for instance, a site-specific policy may refer to a company-wide policy and a country-specific policy.

A major detractor, mentioned by Griffin (2004), is the fact that XACML can increase system overhead; in a given example - a Web page which aggregates many resources – an XACML request is required for each resource.

In a Web services-SOA, one presumes that access to data on the Provider-side can only be gained by way of the Web service and related components; the data would be protected by various standard means (for example, role-based authentication using security tokens in a Kerberos-model and/or IPSec between the application server and the database server). However, access to the Web service needs to be controlled.

Pattern

The Listener module of the Web service has to be able to authenticate the sender-principal and ascertain the permission-set before the business-data contained in the message is processed. To this end, *standard roles* need to be identified, together with a concomitant *standard policy for corresponding access-control* for each (standardized) service.

XACML provides a possible means for persisting authorization from the SOAP-Envelope to the service(s).

XML Key Management Specification (XKMS)

XKMS enables Web services to register and manage cryptographic keys used for digital signatures and encryption via XKMS clients. XKMS server components are implemented by PKI providers. Thus applications are able to interface with key-related services, such as registration and revocation. Even though XKMS is compatible with a number of public key infrastructure (PKI) technologies, it does not require any of them. The XKMS consists of two complementary standards: the XML Key Registration Service and the XML Key Information Service specifications. Together, they allow for the integration of a number of security technologies, including digital signatures, certificates, and revocation status checking. For instance, XKMS can utilize XML-Digital Signatures to protect the integrity of XML-document content (XML Trust Centre, 2004).

In theory, it is therefore possible for a Web service, upon receiving a digital certificate, to dynamically check the certificate revocation list of a PKI provider. It can also make a request for its public key to be distributed to a particular client (after the PKI service has validated the client’s credentials).

Pattern

However, in the context of a GEM, the number of possible PKI service providers and the variety of possible services would make this a non-standard function (as it could lead to a multiplicity-chaos).

XML-Encryption

(Siddiqui, 2002).

Privacy in business transactions is an extremely sensitive issue. The XML-Encryption specification (Infoset) provides a standard means for encrypting both binary and textual data.

The XML-Encryption standard provides standardized metadata for use in XML-documents bearing encrypted data. Standard encryption algorithms (such as Advanced Encryption Standard/Rijndael (NIST, 2001)) can be used to encrypt parts of an XML-document payload or the entire payload. The data contained within standardized tags are processed on the recipient-end in a manner dictated by the XML-Encryption standard. The standard defines a means of communicating information essential for recipients to decrypt the contents of received messages.

Pattern

Since SOAP-documents are XML-documents, XML-Encryption can be used to selectively encrypt a SOAP payload, thereby providing *message-confidentiality*. In a GEM-context, a standard would have to be arrived at for the encryption algorithm. In the generic buying/selling business-pattern (discussed later), the entire payload could be encrypted using a GEM-standardized mechanism, to avoid introducing structural complexity into the “system”.

.

XML-Digital Signatures

XML-Digital signatures are digital signatures designed for use in XML-based transactions. XML-Digital Signatures can be used to authenticate the sender (although the vulnerability to man-in-the-middle attacks still persists, as it does for standard digital signatures). They also provide a means of assuring message integrity, as well as support for standard non-repudiation.

As with the XML-Encryption standard, XML-Digital Signature also supports binary and textual data. A useful feature of XML-Digital signatures is that they enable one to sign selective parts of a message. Also, different sections of an XML-document can be signed by different signatures (Simon, Madsen, and Adams, 2001**).**

Pattern

XML-Digital Signatures (XML DSIG) is also defined by a standardized XML-Infoset. As such, it represents a logical choice for providing support for *integrity and non-repudiation* in the context of a GEM.

Transport Layer Security (TLS)/ Secure Sockets Layer (SSL)

(Canavan, 2001).

The TLS/SSL protocol suite encrypts HTTP communications between Web servers and Web browsers; it provides secure "tunnelling" over the Internet at the transport layer. TLS/SSL makes use of standards-based encryption and authentication, and provides secure access to data and applications over the Web. No specific software is required; TLS/SSL is simply enabled on Web servers, and browsers are designed to provide TLS/SSL support when required. TLS/SSL uses both public and private key encryption, which can include the use of digital certificates. The main disadvantages of TLS/SSL are that it affects network throughput and it does not protect against traffic analysis. Nevertheless, if TLS is applied in a standard manner, it provides a relatively simple means for ensuring *transport confidentiality.* Again, standard encryption algorithms need to be pre-specified in a GEM-context. However, in a GEM-context, the use of TLS/SSL is made redundant by WS-SecureConversation (discussed later).

Secure IP (IPSec)

(Cisco Systems, 2002).

IPsec is a set of open standard protocols developed by the Internet Engineering Task Force (IETF) to support secure exchange of packets at the IP (network) layer. IPsec has been deployed widely to implement Virtual Private Networks (VPN's). Tunnelling can be achieved by using either the Authentication Header (AH) protocol or the Encapsulating Security Payload (ESP) protocol. Two encryption modes exist. In tunnel mode, both the headers and payload are encrypted. This is used in end-to-end network-layer connection. In transport mode, headers remain unencrypted, but payloads are encrypted. This is used when intermediaries exist between endpoints. (In tunnel mode, the inner IP header is also encrypted.) However, with transport mode, the risk of traffic analysis is possible IPSec may also be configured to use digital certificates.

The WS-Security Framework

The WS-Security (Web Services-Security) Specification (WS-S, 2004) provides a standard set of SOAP extensions that can be used when building secure Web services. It is designed to be used in conjunction with other Web service extensions (features) and higher-level application-specific protocols to accommodate a wide variety of security models and encryption technologies.

WS-Security defines additional SOAP elements that can be included within the SOAP-message header to provide persistent integrity and confidentiality in message exchanges between Web services applications (Apshankar, 2002). The specification also defines and describes how security tokens may be used for authentication and authorization in SOAP-messages. A security token can be defined as a representation of a claim about an entity (principal). A claim, in turn, is a statement (assertion) about an entity that may originate from the entity itself or from a trusted claim-issuing authority (O'Neill et al, 2003).

As discussed earlier, WS-Security facilitates integration and interoperability between dyadic security implementations by supporting a wide variety of security technologies. WS-Security is considered to offer a level of abstraction beyond the various models that may be implemented at various business entities (O'Neill et al, 2003). WS-Security also provides for end-to-end security at the messaging level by defining how SOAP-message integrity and confidentiality can be preserved en route between intermediaries to the intended Web services.

WS-Security is one of a number of other specifications outlined in a "security roadmap" for Web services created by IBM and Microsoft. These specifications include WS-Policy, WS-Trust, WS-Privacy, WS-SecureConversation, WS-Federation and WS-Authorization. Together with WS-Security, the specifications comprise a pyramid model, with some specifications being based on those that precede them and all the specifications being inter-related to form a comprehensive Web service security infrastructure model (Apshankar, 2002). This is illustrated in the diagram below.

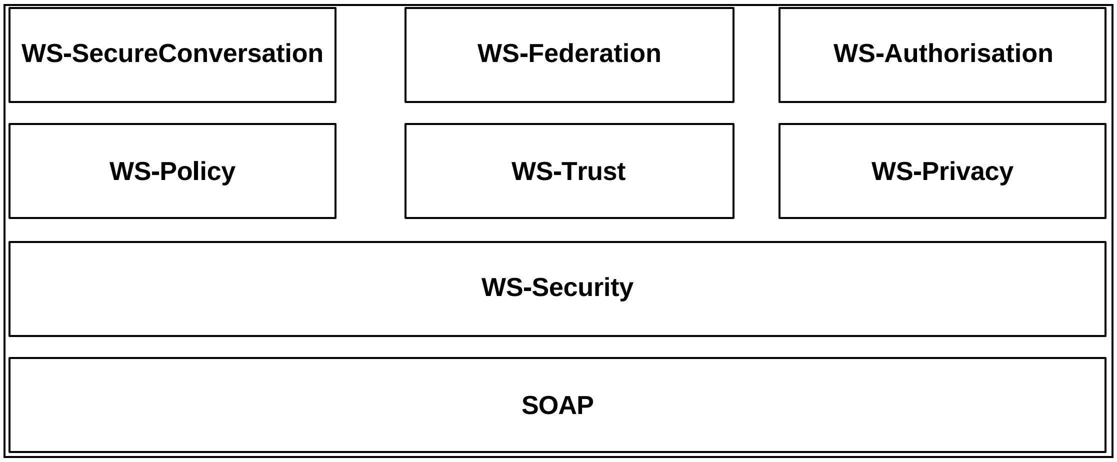


Figure 3.15. IBM-Microsoft Web Services Security Framework (O'Neill et al, 2003)

Each of the additional specifications will now briefly be described (based on O'Neill et al, 2003).

To secure a message end-to-end, WS-Security implements security measures in SOAP-header blocks. Although the WS-Security framework complements the specifications described in this section, it is supported further by a series of supplementary standards, listed in section (g) below.

WS-Policy, WS-Trust and WS-Privacy constitute the *Policy Layer;* the latter provides a number of building blocks for the creation of trust and privacy policies. WS-SecureConversation, WS-Federation and WS-Authorisation, together known as the *Federation Layer*, utilize these policies to unify disparate trust domains.

(a) WS-Policy

Though not limited to providing security-related policies, this standard is a key part of the WS-Security framework. Existing corporate security polices can be expressed through policy assertions that can then be applied to groups of services.

Typically, Web service-Providers can therefore use WS-Policy to describe the security policy for (successfully invoking) their Web services. The policy would specify, for instance, the types of encryption, encryption algorithms, digital signature algorithms, as well as the supported security tokens, required in a SOAP-request-message. The means for binding these security requirements to a SOAP-request is also specified.

Pattern

Therefore, in a GEM, *a single (extensible), pre-specified security policy for the standardized Web service interface (the Listener) that handles the receiving of SOAP-requests can be described by WS-Policy*. All client-participants would be required to subscribe to the same policy. WS-Policy would be included as a feature in the pre-specified SOAP-envelope.

(b) WS-Trust

WS-Trust can be used to create a “confederation” of trusts – with the aim of unifying discrete dyadic trust relationships – by validating the security tokens used in each dyadic relationship. The validation can involve third-party trust authorities.

The trust relationships can either be established directly or brokered. In the latter case, a trust proxy is used in order to request security tokens based on WS-Policy information. These tokens would be requested and obtained from some token issuer and then inserted into a SOAP-request by the trust proxy on behalf of the requestor. The tokens would be digitally signed and encrypted to ensure their integrity and confidentiality en-route to a Web service.

Pattern

In the context of a GEM, in order to avoid the multiplicity paradox, “m x n – crosses” involving a diversity of state variables, are to be avoided. Using WS-Trust might, therefore not be expedient in a GEM-context. Hence, rather than attempting to integrate entities – in which the complex structure of the emergence can be predicted – a single trust model can emanate from the standardized (GEM) policy specified with WS-Policy. A standardized identification-and-authentication claim-mechanism (for example, X509 certificates) needs to be specified.

(c) WS-Privacy

Individual organizations can use WS-Privacy to extend WS-Policy and WS-Trust with specifics regarding privacy policies for their Web services.

SOAP-requests for Web services would need to conform to type of security tokens specified in WS-Security. WS-Trust would evaluate the requests by comparing embedded privacy claims (within tokens) against privacy statements expressed within WS-Policy descriptions.

Pattern

Again, in the context of a GEM, using WS-Privacy would seem superfluous, considering that WS-Policy and WS-Security would be used to specify a single, standardized privacy policy and the standardized security controls to implement it. Individual, privacy policies would have to operate beyond the common GEM range of collaboration.

(d) WS-SecureConversation

WS-SecureConversation provides formal definitions for the creation and exchange of security contexts and associated session keys in a secure ‘session’ between a Web service-requestor and a Web service. It specifies a process of dual authentication using SOAP-messages. This ‘session’ includes the derivation of session keys. WS-SecureConversation persists security credentials for all SOAP-messages exchanged in a session. WS-SecureConversation is based on WS-Security and WS-Trust.

Pattern

In a GEM-context, this is mechanism is obviously preferred to TLS/SSL, since it can form an integral part of the standardized SOAP-envelope, rather than using an additional mechanism.

.

(e) WS-Federation

The WS-Federation specification defines how trust relationships are managed in a heterogeneous security environment. Using WS-Federation, a requestor authenticated to one Web service that consumes a particular type of authentication-token can automatically be authenticated to another Web service that may consume a different authentication-token, assuming that the two services have a trusted relationship. WS-Federation is based on WS-Security, WS-Policy, WS-Trust and WS-SecureConversation.

Pattern

The argument for combining dyadic trust domains has been presented in the WS-Trust section. However, if the number of authentication-token types allowed is reasonably small, it would be expedient to allow organizations to retain the type they have invested in. In this case, a single WS-Federation-defined standard for a GEM would probably be acceptable.

(f) WS-Authorization

WS-Authorisation enables the defining and managing of access-control policies for Web services. Various specific authorisation mechanisms, including Access Control Lists (ACLs) and Role Based Access Control (RBAC) are accommodated. The manner in which claims are represented within security tokens can be defined.

Pattern

Like WS-Privacy, this provides a means to extend the details provided in WS-Policy and WS-Security. Presumably, as the envisaged-GEM evolves, so the use of this specification would become more significant.

It is argued here that unless a single specification for a security model for the GEM is formalized, the range of variation in using the Web Services Security framework will simply result in fostering the multiplicity paradox discussed earlier. One also needs to be wary of contributing to the structural complexity of the emergence.

(g) Supplementary specifications for inclusion in SOAP-headers

The following is a list of currently available specifications (from www.specifications.ws) which can be used to define and extend SOAP functionality. This is provided here merely for the sake of illustrating the range of specifications; many – as will be argued in Chapter 5 – are more suited to dyadic collaborations than to a GEM-context.

Security Specifications

Web Services Security: SOAP Message Security

Web Services Security: UsernameToken Profile 1.0

Web Services Security: X.509 Certificate Token Profile

WS-Security

WS-SecureConversation

WS-Trust

WS-Federation

WS-Federation Active Requestor Profile

WS-Federation Passive Requestor Profile

Web Services Security Kerberos Binding

Reliable Messaging Specifications

WS-ReliableMessaging

Transaction Specifications

WS-Coordination

WS-AtomicTransaction

WS-BusinessActivity

Metadata Specifications

WSDL

UDDI

WS-Policy

WS-PolicyAssertions

WS-PolicyAttachment

WS-SecurityPolicy

WS-Discovery

WS-MetadataExchange

XML Specifications

XML

Namespaces in XML

XML Information Set

Business Process Specifications

BPEL4WS (Business Process Extensibility Language for Web Services)

Specification Profiles

Devices Profile

Web Services Specification Licensing

WS-Security – extended considerations

WS-Security allows the use of various security tokens with SOAP-messages. These may be binary tokens such as X.509 certificates (using the Web Services Security: X.509 Certificate Token Profile specification in the list above) and Kerberos tickets, or username/password combinations (using the Web Services Security: UsernameToken Profile 1.0 specification in the list above), or XML-based assertions (as in Security Assertion Markup Language, SAML). These tokens need to be secured in transit to their intended recipient. The integrity of security tokens and of the SOAP-message payload can be ensured through the use of XML Digital Signatures (by signing the tokens and/or the entire message payload or selected elements of the payload). Similarly, confidentiality can be implemented using XML Encryption to encrypt security tokens and/or the entire message payload or selected elements of the payload. In this way, only the intended message recipients are able to decrypt tokens and data for consumption and any unauthorised modifications to tokens and data can be detected by comparing hash values.

WS-Security, while defining a means for including one or more authentication and authorization tokens of varying formats within a SOAP-message, does not provide a complete Authentication and Authorisation mechanism in an application-to-application collaboration model. Similarly, while defining a means of providing integrity (using, for instance, XML Digital signature) and confidentiality (using, for instance XML Encryption) at the messaging level, it does not stipulate how PKI functions are to be co-ordinated or how non-repudiation and auditing services are to be implemented (Apshankar, 2002).

Web services consumed in a B2B-integration model ultimately reside on servers. It is vitally important that all servers participating in message exchanges are secured through the use of devices or software on which granular security policies are defined and all superfluous entry points are closed. These are just some of the security considerations, but it provides sufficient reason to conclude that WS-Security in isolation cannot be considered an all-encompassing Web services security model.

Conclusion

It is clear that Web services provide the most promising means of achieving global interoperability. Yet, within the context of a global multilateral electronic market-space, unless specific directives are forthcoming at the highest level, standards such as WS-Security[1] remain protocol-neutral and continue to support an opposing momentum. So, precisely how will Web services be utilized in promoting

[1] http://www.verisign.com/wss/wss.pdf (contributions from Microsoft, Verisign and IBM)

interoperability in a GEM context? The following scenario is envisaged (as an introduction to the model to be described in Chapter 5):

At this point, it seems fair to surmise that the applications of all participants in the GEM would comprise either Web Services, Web Services clients or both. Further, the service-provider application should expose a standardized Listener interface that handles SOAP-messages. A standardized SOAP-envelope for the GEM can be composed from elements and attributes from WS-Policy, WS-Security, WS-Addressing, MTOM, WS-SecureConversation, and so forth, to extend the functionality of SOAP and Web services in a uniform manner.

The open-standards nature of SOAP and Web services, as described earlier, implies that GEM-collaboration will be platform-independent. A *minimalist-business-pattern standard* (to be clarified in Chapters 4 and 5) will be used by all applications in order that only fundamental business processes occur according to pre-defined choreography implicit in the standard. This is to mitigate the increasing structural complexity implicit in the evolution of a GEM. Library-components for Web services encapsulating the minimalist-business-pattern can be made available (as the equivalent of ebXML's Core Components; to be elaborated later) to developers.

Pattern

To standardise the service, *a global XML nomenclature* (a taxonomy for defining metadata), based on a common vocabulary (mainly for business process terms, metadata and programming identifiers) and common semantics (unequivocal meanings of generic words used) would be necessary. Any particular pre-specified business process should occur in a *pre-specified business workflow*, specified using the common nomenclature and comprising business documents based on *pre-specified XSD's. Existing standard Infosets,* such as BPEL4WS, already possess a common nomenclature and processing guidelines, which can be incorporated into a GEM standard, with pre-set, uniform values for GEM-common data-items. Specific data-items, for example, the naming of an invoice-XML-document, also need to be considered in the pre-specified nomenclature. Certain specific values would be entered in the standardized SOAP-envelope incorporating elements and attributes from standardized features (Infosets), for addressing, security, and so forth.

Chapter 4 will focus on prevailing B2B-collaboration models. It will attempt to identify the shortcomings of these models and to suggest possible improvements, in the light of patterns observed thus far.

Chapter 4

An Analysis of Existing Business-to-Business-Collaboration models

Introduction

In this chapter, various existing B2B-collaboration models will be examined in the light of the issues raised in Chapter 2 (the nature of the envisaged GEM, the multiplicity paradox and the standardized, open interoperability interface) and critical success factors (CSF's).

As explained in the previous chapter, an organization trading with a large number of trading-partners (TP's) could be encumbered with a different set of business processes (together with the corresponding nuances in related documents) and a different functional services configuration for each TP. The converse situation could be construed where *a single, uniform standard* applies to all collaborations between all TP's. In the context of a global market-space, this would be represented by the "ideal" set of configuration-items (variables), denoted by set K and the corresponding set of "laws", L, described in Chapter 2.

Electronic Business eXtensible Markup Language (ebXML) was developed to provide such a standard (a specification suite). The goal of ebXML was to facilitate the creation of a single global electronic market. The ebXML specification suite attempts to provide the technical infrastructure for enabling the following general functions ("Enabling Electronic Business with ebXML", 2001; Keily, 2001):

Potential TP's (pTP's) need to *discover each other* and the products and services each has to offer. They need to *determine which shared business processes* to implement in the collaboration. Further, pTP's need to *arrive at a formal agreement* on the contractual terms regarding the chosen business processes and the exchange of information (the contact points: network/host addresses or URI's; communication:

messaging protocols and procedures; documents, and so forth). pTP’s would then be able to *exchange services and information* in an automated fashion, as pre-specified by the agreement.

ebXML is therefore based on the SOA paradigm. Although the interoperability mechanisms employed by ebXML can be criticized in terms of the multiplicity paradox, the goals and objectives of ebXML provide a solid foundation for a GEM framework. In subsequent sections, ebXML will be examined in greater detail, focussing on the B2B-collaboration issues raised in Chapter 2. For instance: Does ebXML cater for the nuances in B2B-collaboration as described in Chapter 2? Does ebXML satisfy the fundamental requirements for global interoperability?

ebXML

ebXML is a comprehensive specification suite, comprising a vast array of documentation. The following discussion is an attempt to summarize the salient points.

Essentially, ebXML was an eighteen-month international initiative started in November 1999, involving members of OASIS and UN/CEFACT, with the mandate to identify the technical basis upon which the global implementation of XML (eXtensible Markup Language) could be standardized. It is a specification suite for business collaboration, creating a standardized way for companies to carry out common business practices and processes. ebXML has the stated aim of creating a “global marketplace” where companies of any size may participate due to the benefits of reduced development costs, increased flexibility and resulting ease of use. It is intended to be non-proprietary, platform-independent and open-standards-based, using standards such as Hypertext Transport Protocol (HTTP), Transport Control Protocol/Internet Protocol (TCP/IP), Multi-purpose Internet Mail Extension (MIME), Simple Message Transport Protocol (SMTP), XML, and Simple Object Access Protocol (SOAP). Currently, ebXML enjoys the support of various standards organizations, including Accredited Standards Committee (ASC) X12, the XML/EDI group, the Data Interchange Standards Association, GENCOD-EAN France, XMLGlobal and PeopleSoft Incorporated (Irani , 2001;Taschek, 2002; Keily, 2002;

Naujok and Berwanger, 2001, Rawlins[1] 2001). Further, ebXML allows the investments made in acquiring EDI infrastructure to be preserved (Grangard et al., 2001).

The following section briefly elaborates on the actual ebXML Technical Architecture specifications.

ebXML Technical Architecture

("Enabling Electronic Business with ebXML", 2001).
ebXML is built on three core concepts:

1. *An infrastructure to ensure data communication interoperability.*

   This is facilitated in two ways. Firstly, a standardized message transport mechanism, containing a well-defined interface, packaging rules, with a predictable delivery and security model, is specified. Secondly, a standardized 'business service interface' to handle all incoming and outgoing messages is specified.

2. *A semantic framework ensuring commercial interoperability.*

   ebXML specifies a *metamodel* to define business process and information models; participants can submit their own business processes and information models. A set or re-usable business logic derived from *core components* reflecting common business processes and XML vocabularies was proposed, but remains incomplete (Rawlings, 2001[1]). ebXML specifies a process for defining actual message structures and definitions as they relate to the activities of the business process model.

3. *A mechanism catering for organisation discovery, business relationship establishment and business activity.*

   This is facilitated by means of a shared Registry/Repository and a formal process by which a Collaboration Protocol Agreement (CPA) is reached between to pTP's. The shared Repository allows organisations to register and discover the nature of each other’s business services via a partner information profile (called the Collaboration Partner Profile or CPP). The Repository is used for the storage of organisation profiles, business process models and other related technical message structures.

Unless otherwise specified, the material in the following section and subsections is derived from Specification documents available on the ebXML.org website – document filenames are prefixed by “eb” - and Rawlins[2] (2001).

The ebXML technical architecture, as depicted in the ebTA document, may be considered to comprise two architectures. The first is an architecture for the ebXML technical infrastructure, often referred to as a *product* architecture. The second is an architecture for performing systems analysis and development, often referred to as a *process* architecture.

Product Architecture Overview

The product or technical infrastructure is composed of the following major components:

- The Messaging Service
- The Registry
- TP Information
- Business Process Specification Schema (BPSS)

The Messaging Service provides a standardized way in which to exchange business messages between organisations. It provides a means to exchange a payload – which may or may not be an XML business document – and a means to route the payload to the appropriate internal application on the receiver’s end. Any application level protocol can be used, including common protocols such as HTTP, SMTP and FTP; it is therefore protocol-neutral.

The Registry essentially stores information about items that actually reside in a Repository. Items in the Repository are created, updated or deleted via requests made to the Registry. No particular implementation of the Registry/Repository database is specified in ebXML. However, there is a Registry Information Model, contained in the ebRIM document, specifying the minimum information model that the registry must support. It specifies the types of information stored about registry items; for example, XML schemas for business documents or TP agreements. The Registry Services specification, contained in the ebRS document, specifies how foreign applications should interact with the registry via registry services interfaces; again, no implementation details are provided.

The *TP Information* consists of Collaboration Protocol Profiles (CPP's) and Collaboration Protocol Agreements (CPA's). The CPP's (W3C XML Schemas or Document Type Definitions) specify the details of how an organisation conducts its business electronically. These details could include information on how to locate and contact an organisation, the type of network and transport protocols the organisation uses, network addresses, security implementations and references to business process specifications (information models for how the organisation conducts business).

The CPA specifies the details of a mutual agreement between two organisations to conduct business electronically between each other. It is formed by combining two CPP's of the involved organisations. It was intended that the CPA be used by a software application to configure the technical details for the actual conducting of business between two organizations. The CPA/CPP specification describes the general tasks and issues in creating a CPA from two CPP's, but does not specify an actual algorithm for this process.

It is contended here that this model must ultimately succumb to the multiplicity paradox (discussed in Chapter 3); this is considered a major short-coming of the ebXML model. For each pTP-pTP relationship the number of permutations for technical inclusions in a CPA could be quite considerable; for an n:m situation, the permutations would simply be unmanageable.

The *Business Process Specification Schema (BPSS)* is an XML Schema document, which is used to describe how an organisation conducts its actual business. Note that the BPSS deals with the actual *business* process – identifying roles, transactions, the overall business process, actual business document usage, document flow, security and legal aspects, business level acknowledgements and status - whereas the CPP/CPA deals with the *technical* aspects of how to conduct business electronically. This specification allows an organisation to communicate its business processes and practices to other organisations, thereby enabling business process integration/collaboration. The BPSS could be used by a software application to configure the business details in the collaboration.

Again, the criticism that can be raised here is that innumerable business processes can be defined. Although a common semantics model is provided by ebXML, the intrinsic potential for the multiplicity paradox to occur is very obvious. A potential "buyer" would have to retrieve the BP schemas for each potential "Seller" with which it plans to collaborate.

Rawlings (2001) states that the ebXML infrastructure is completely modular and only loosely related – allowing for the various infrastructure components to be used fairly independently. The components interact with each other as designed, but, in most cases, this is not required. For example, the messaging services can be used completely independently, although a message header might contain a reference to a CPA. Both the CPA and CPP are entirely optional in ebXML. The Business Process Specification (BPS) may specify how the various services offered by the messaging service, such as the request for a digital signature, are to be used in the conduct of a business process; however these are also not absolute requisites. The Registry may store any type of object - of XML type or any other type - not only the CPAs, CPPs and Business Process Specifications. However, all communications with the Registry must use the ebXML messaging service to do so.

Again, the capriciousness implicit in the ebXML model complicates contemplation on the contents of the sets, K and L, described in Chapters 2 and 3. State-variables and rules ("laws") are not fixed, diminishing the predictable structure of the envisaged GEM.

The infrastructure components described above will be elaborated in subsequent chapters, in order to point out specific strengths and weaknesses. The following subsection will provide an overview of the process architecture.

Process Architecture Overview

The process architecture, also described in the ebTA document, is based on the UN/CEFACT Modelling Methodology (UMM) and its corresponding Meta Model. The UMM prescribes how to perform business process and information modelling for electronic commerce, utilising UML. The Meta Model specifies all the items that must be produced from the analysis and their relationships. The BPSS, discussed previously in the product architecture overview, represents a subset of the information found in the Meta Model.

Ideally, the BPSS should be developed by applying the entire UMM process. However, the use of UMM is not obligatory and other tools and aids are provided for the creation of the BPSS. These include the following:

- Business Process Analysis Worksheet & Guidelines
- A Catalogue of Common Business Processes
- E-Commerce Patterns
- Core Components

In Figure 4.1 below, the use of the ebXML Registry/Repository (R/R) for centralized control is illustrated. Business process and information models are ultimately available as standardized XML Schema documents. Implementers may use the Schema documents to determine the specifications for each business process (BP). The diagram shows how the Schema are arrived at: via UNCEFACT Modelling Methodology (UMM) and Meta Models and/or using Business Process and Analysis Worksheets and Guidelines, a Catalogue of common Business Processes, Core Components and common Business Patterns. Organizations may use the prescribed process model for designing and registering their own Business Process Schema.

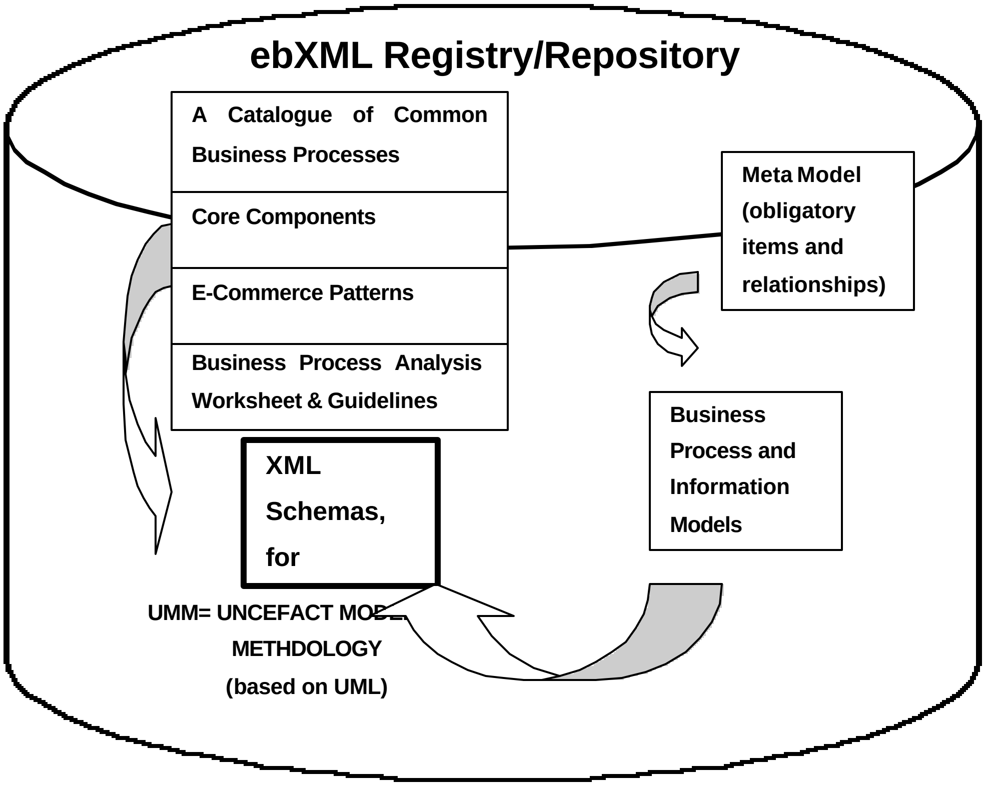

ebXML Registry/Repository
A Catalogue of Common Business Processes
Core Components
E-Commerce Patterns
Business Process Analysis Worksheet & Guidelines
XML Schemas, for
Meta Model (obligatory items and relationships)
Business Process and Information Models
UMM= UNCEFACT MODEL METHDOLOGY (based on UML)

Fig 4.1. The workings of the process model

The *Business Process Analysis Worksheets & Guidelines* are a set of worksheets and guidelines for using the worksheets, designed to assist the analyst in producing a BPSS XML-document, which describes a business process. The ebXML architects envisage that Business Process Editors software would, eventually, automatically generate the XML BPS.

A *Catalogue of Common Business Processes* provides an initial cross-listing and description of business processes, which may be common in numerous industries.

E-Commerce Patterns provide examples and descriptions of common business patterns. The only pattern made available by May 2001 was for plain contract formation.

The *Core Components* are similar to "common business objects". However, the specification (ebCC) makes provision for a *context* feature (optionally referring to company, industry type, or geography). A methodology for discovery, re-use, and extending core components (CC) into industry-specific domain components, is also included in the specification. A Naming Convention, based on ISO 11179, for core component names was also developed. Both an initial catalogue of core components and a Catalogue of Context Drivers for available context extensions (such as geographic region, industry, role and product) are also provided.

The analysis worksheets are based on the Catalogue of Common Business Processes and E-Commerce Patterns. They are used to develop a specification document that covers the essential data set specified by the UMM Meta model. Rawlings (2001[1]) suggests that the relationship between the BP work and the Core Components work is not very clear, especially since the BP document makes reference to context as being "beyond the scope" of the ebXML effort.

The various components of the process architecture may also be used independently. For instance, XML schemas for business documents could be developed from core and domain components, without any formal business process analysis having been performed.

Rawlings (Ibid) describes the BPSS as being the only strong link between the two architectures.

Figure 4.2. shows all the ebXML specifications (in both the process architecture and product architecture).

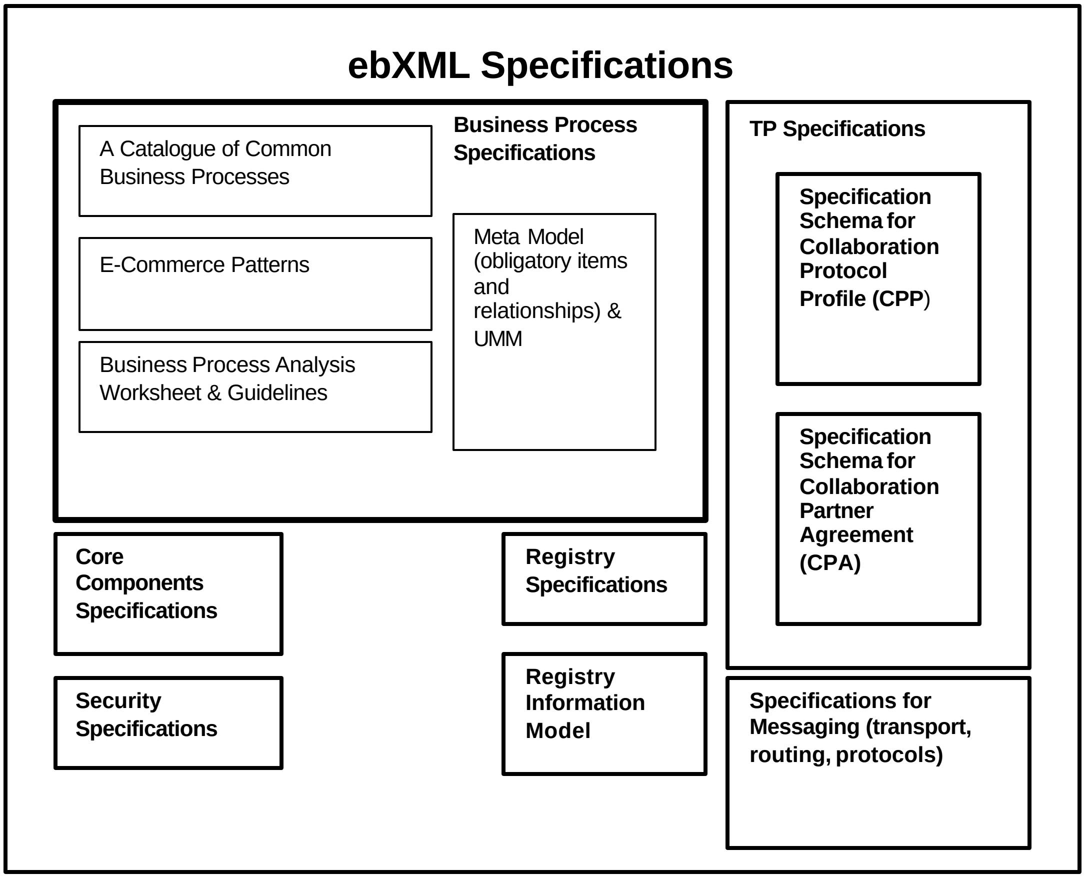

ebXML Specifications
Business Process Specifications
A Catalogue of Common Business Processes
E-Commerce Patterns
Business Process Analysis Worksheet & Guidelines
Meta Model (obligatory items and relationships) & UMM
TP Specifications
Specification Schema for Collaboration Protocol Profile (CPP)
Specification Schema for Collaboration Partner Agreement (CPA)
Core Components Specifications
Registry Specifications
Security Specifications
Registry Information Model
Specifications for Messaging (transport, routing, protocols)

Figure. 4.2. ebXML Specifications

In Figure 4.3., the envisaged use of ebXML is illustrated.

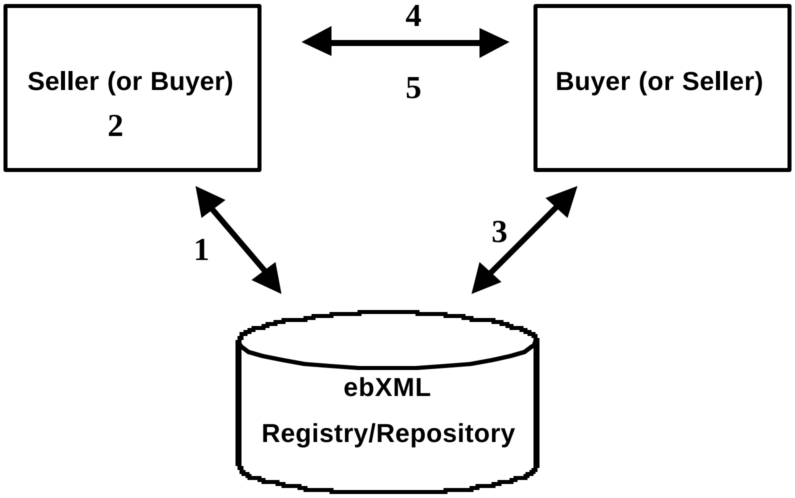


Figure. 4.3. How ebxml is intended to work

Step 1: A Seller (or Buyer), wishing to trade using ebXML, downloads the specifications for developing a compliant system (“shrink-wrapped” applications may also be available)

Step 2: The ebXML-compliant system is developed. The TP completes the Business Process Specification Schema (BPSS, an XML Schema) for the BP’s he/she requires, completes a Collaboration Protocol Profile (CPP) for each BP, and registers it on the R/R.

Step 3: A Buyer (or Seller) may use the Regsitry/Repository (R/R) to search for (discover) an appropriate potential TP. The CPP’s of that potential TP are downloaded.

Step 4: A Collaboration Partner Agreement (CPA) is negotiated with the prospective TP, based on the CPP’s of both parties. The CPA (an XML-format specification) includes elements for BP’s, messaging protocols, and security.

Step 5: Once mutual agreement has been reached, trading between the TP’s may ensue.

ebXML uses a Messaging Service for secure and reliable communication between TP’s. The Messaging Service uses SOAP and HTTP. It is clear that ebXML and Web services are complementary technologies and may be used in conjunction with each other (Patil & Newcomer, 2003). Still, ebXML’s suitability for its intended goals is debatable; this is discussed in the next section.

An Appraisal of ebXML vis-à-vis the GEM

In the ebXML Terms of Reference document, compiled by UN/CEFACT and OASIS, the various issues and objectives requiring attention en route to achieving the initiative’s primary goal of enabling interoperability are identified and discussed. Exactly how well the initiative has managed to address these points will be discussed in this section. Unless otherwise referenced, the material in this section is based on the authoritative assessment of the ebXML initiative found in Rawlings[3], 2001; Rawlings was the chairperson of the ebXML Requirements Specification team.

The various aspects constituting the “goal of interoperability” in ebXML, are listed as:

Common Business Processes, Common Semantics, Common Vocabulary, Common Character Encoding, Common Expression, Common Data Transfer Protocol, Common Network Layer, and Common Security Implementations. These aspects form a comprehensive set of critical success factors (CSF's) for evaluating the efficacy of ebXML as a GEM model. The same CSF's will be used to evaluate the model proposed in Chapter 4.

The Common Business Process aspect

The *Common Business Processes* aspect is depicted by "Both entities involved in the exchange of data MUST be engaged in executing the same transaction in the context of a business process" (Rawlings et al., 2001). However, Rawlings (Ibid) retrospectively admits that ebXML did not define specific business processes, but instead provided a methodology, with which to analyse and document business processes, and to represent the key attributes of these business processes in an XML-document, which is achieved utilising the BPSS.

The ebXML specifications did develop an initial high-level catalogue of business processes, but these contained few specifics. Rawlings believes the methodology, including the UN/CEFACT methodology on which this business process work is based, is still incomplete in many areas. He sums up the problem by stating that the bulk of the work focused on defining the overall business process context and that very little work was done in actually addressing the "same transaction" aspect.

As pointed out earlier, different business models encompass different business processes. Further, different industries have different business processes. Even within a particular industry, business processes differ. The whole concept of business process re-engineering (BPR) adds an extra perspective on what is already an extremely malleable and dynamic aspect of interoperability. Thus, allowing businesses to submit their own business process models – as in the ebXML model – will inevitably exacerbate the multiplicity problem.

This thesis contends that the basis upon which common business processes should be elaborated in a GEM context is to emphasize the “common” (to reduce the contents of sets K and L).

Pattern

All other forms of collaboration should initially be eschewed in favour of the more “common” business patterns. Focussing on the most elementary collaboration pattern, commerce typically involves the exchange of goods or services usually for some type of revenue-generating means. Therefore, the most generic business pattern will be the buying/selling of products and services (Rawling’s notion of “same transaction”) and hence the pattern would involve complementary Provider/Consumer views of:
Placing an order, fulfilment of the order, and payment for the order.

For every Provider-Consumer collaboration of this nature, a *single, standardized generic, minimalist business-process model* can be specified (preferably by an authority like the W3C). This would include: a standardized sequence/activity (choreography) model and a metadata model for all documents involved in the exchanges (XSD’s showing all the elements and attributes based on a standardized nomenclature). The Business Process Extension Language for Web services (BPEL4WS) would provide a useful platform for standardizing the business process nomenclature.

What would be the GEM implications for this minimalist model? All participants can create their own applications for buying/selling, using the standardized business pattern, the same XSD’s for validating documents and the same nomenclature for metadata in (XML) forms (orders, receipts, and so forth.). For instance, all applications can processes XML-element names (such as “provider_ID” or “message_ID”) as deemed fit by their systems; the same names will be uniformly used. The application developers merely need to take cognisance of the (minimal, pre-specified) business process event-sequence and metadata for documents that need to be processed or generated. Any communication beyond the generic model can initially be effected in a non-standardized fashion (for example, by e-mail); subsequent revisions of the standardized business pattern can incorporate increments to the essential pattern, in a hierarchical fashion, provided that the multiplicity paradox is avoided. This approach is discussed further in Chapter 5.

The Common Semantics, Common Vocabulary and Common Expression Aspects

The *Common Semantics* aspect is depicted by "Common meaning, as distinct from words, expression, or presentation" (Rawlings et al., 2001). Rawlings (2001[3]) states that ebXML developed an extremely limited initial catalogue of Core Components, with what he believes to be (as mentioned above), an incomplete development methodology. He mentions that it did, however, assign a unique identifier to each semantic concept.

The *Common Vocabulary* aspect is depicted by "A direct correspondence between words and meaning" (Rawlings et al., 2001). Rawlings (2001[3]) feels that the distinction between this and the Common Semantics is rather subtle. He feels that it is possible for concepts to have a common meaning, yet be called by different names. He states the criterion for a common vocabulary is the use of a distinct name, with the considerations for this criterion not being properly separated from those of the Common Semantics.

The *Common Expression* aspect is depicted as a "Common set of XML element names, attributes and common usage of those attributes, common approach to document structure" (Rawlings et al., 2001). Rawlings (2001[3]) states that this aspect was not addressed at all. He explains that the reason for this was ebXML's strategy to enable several existing XML approaches to interoperate instead of choosing and prescribing a standardized approach. He further feels that the initiative tried to address a very broad scope and tried to make its approach applicable to technologies other than XML.

The three aspects discussed above have the same implementation implications vis-à-vis interoperability and are thus considered together. Extrapolating the notion of the *generic business pattern specifications*, introduced in the previous section, the need for a common understanding of the meanings of terms used in the standardized specifications for the documents (XSD’s and forms), the sequencing of atomic activities/events, the events themselves, and so forth, is of critical importance.

Pattern

The templates (XSD’s) for the standardized documents should reflect the standardized nomenclature. The common semantics and vocabulary must also be published to ensure unequivocal use of the standardized terms used in the specification.

All developers must understand and use terms such as “Provider_Id” and “invoice” (hypothetical examples) in a manner consistent with the specified common semantics and common vocabulary. Even though the back-ends of systems will manipulate locally-adopted/indigenous terms (as identifiers, field-names, and so forth), the meanings of input/output data, obtained from interfacing with complimentary Provider/Requestor systems, should be unequivocal. Consider, for instance, parsing an XML-document and extracting a value (for example, <Provider_Id>AD1234</Provider_Id>). The metadata (the element name and the data-type, in this case) for the XML-document would have to be consistent for every instance of the document processed.

Pattern

To this end, a W3C-standardized taxonomy of GEM terms (a standardized nomenclature), would be essential. This naming system would be incorporated in metadata in business document XSD’s, in identifiers in components, and in vocabulary used in describing business processes and specification documents. (Legacy systems would translate or marshal and un-marshal types). It would probably be more expedient to provide core components, as far as possible, in which this naming system is persisted.

The Common Character Encoding Aspect

Pattern

The *Common Character Encoding* aspect predictably has one candidate: the UNICODE character set.

This is essentially because XML has already been established as the lingua franca for interoperable systems, and UNICODE is specified in the W3C XML Version 1.0 technical specification.

The Common Data Transfer Protocol Aspect

The *Common Data Transfer Protocol* aspect could also be seen to be inadequately addressed, with the Message Service Specification, in its pursuit of “openness”, being protocol-neutral. This is affirmed by Rawlings (2001[3]). ebXML offers protocol-bindings for HTTP and SMTP, with the CPA/CPP Specification only allowing HTTP, SMTP and FTP for utilisation.

Although the range of communication protocols is not vast, interoperability in the GEM context would be far better suited if applications need only be aware of messages using standardised protocol-binding. A fully-automated application would receive a notification of a (SOAP) message having arrived (polling the message service/Listener). It would then interrogate the message service for the data to be manipulated. If the message were being served up by an HTTP service as per a global specification, then the other services (for example, FTP) would not have to be interrogated. The obvious candidates for the transfer protocol are HTTP and SMTP, if only for their ubiquity.

E-mail already provides a global mechanism for the exchange of business information. It is interoperable across disparate platforms; has a tried-and-tested global asynchronous store-and-forward mechanism obviating the need for developing queuing and caching mechanisms; it can employ the most stringent security measures; and disaster-recovery and business-continuance measures for it have already been well-researched, developed and applied. Further, SOAP-messages could be sent as attachments, which could be vetted and processed according to internal (Provider) business policy. A message handler could interrogate the SOAP-message for details of roles of nodes (as per SOAP 1.2 specifications), URI of nodes, and so forth.

HTTP, on the other hand, allows for real-time request-response collaboration between endpoints. The endpoint URI is usually the address of the method that handles the request.

In addition to providing a concrete realization of a SOAP Infoset (XML Information Set, 2004) between adjacent SOAP nodes along a SOAP-message path, *protocol binding* provides the mechanisms to support *features* that are needed by a SOAP application. A SOAP feature, as discussed in Chapter 2, is a specification of a certain functionality provided by a binding. A feature description is identified by a URI, so that all applications referencing it are assured of the same semantics. For example, a typical usage scenario might require many concurrent request-response exchanges between adjacent SOAP nodes, in which case the feature that is required is the ability to correlate a request with a response. Other examples include: "an encrypted channel" feature, or a "reliable delivery channel" feature, or a particular SOAP-message exchange pattern feature. The application could implicitly "inherit" this latter feature provided by the (HTTP) binding, and no further support need be provided at the application or the SOAP level. Note that the use of the SOAP-request-response message exchange pattern in the SOAP HTTP binding is available to all applications, whether they involve the exchange of general XML-data or RPC’s encapsulated in SOAP-messages.

Thus, the obvious choices in this regard are HTTP and SMTP. This will be discussed further in Chapter 5.

The Common Network Layer Aspect

The *Common Network Layer* aspect too is a victim of the pursuit of “openness”, with no standardized network layer protocol, such as IP on the Internet, being specified in ebXML.

Pattern

However, it is hard to contemplate an alternative protocol to IP, even for non-Internet applications; especially with the advent of IPv6 (version 6).

The Common Security Implementations Aspect

The *Common Security Implementations* aspect in ebXML is addressed via a choice of several security implementations, including XML Digital Signatures and XML Encryption. No standardized security model is provided.

In a GEM environment, the following have to be secured by each participant: the internal network, the Web server, the application server, the database, the application, and the messaging system (including the messages) between application interfaces. Each of these can be implemented and configured in a variety of ways. This is a classic foundation for the multiplicity paradox discussed in Chapter 2, since the number of permutations is innumerable (as depicted, for instance, in Meier et al, 2004).

What authentication mechanism should be used? Should it occur at the operating system-level (user-identity and security-context), or at the application-level (generally, user-identity only)? Where should authorization take place? Should this occur at the resource level or at the application level? At the resource level, the following need to be considered: Web Server resources, for example, Web pages and Web Services; Database Resources; Network Resources, for example, the file system, shared folders, Directory Services, and so forth; and System resources, for example, the Registry, Event Logs, configuration files, and so forth.). At the application level, security is generally role-based, typically at the method-level of components. Various implementations for security controls at both levels are available. How will the user's identity be persisted through the various layers/tiers of the application? Will the identity flow from the Web Server to the Application Files to the Database? If so, in what manner will this be accomplished?

An envisaged-GEM calling-application (typically, a SOAP-based Service client) would work via an interface. Thus, a calling-application makes an HTTP-request containing a Web service Service-URI. If the SOAP-based Service is for use by a closed user group (CUG) or bears security differentials defined by the identity or role of the principal, then the principal (application or user) needs to first be authenticated and then be granted the permission-set allowed (appropriate authorization). This authentication-authorization mechanism can be application-level verification of identity and permissions (typically, code-based) or operating system-level verification of identity and permissions (generally, role-based). The final run-time permission-set of the principal is resolved from the permissions granted to the principal by the application (sand-boxing) and the permissions granted by the environment in which the application is executed.

Pattern

It is thus clear that security assertions need to be persisted at all levels of n-tier applications.

The identity credentials of the principal can be obtained during the Request (using request variables such as the logged-in user identity or the IP-address of the calling principle) or by explicitly demanding authentication from the user or from the calling application. If the user is interacting with the provider-service via a Web browser, data can be input as required. However, if the collaboration is to be automated, the consumer-application needs to be able to interact with the provider-application in a pre-defined manner. In a GEM context, the interoperability between applications would be subject to the multiplicity paradox problem described earlier.

Pattern

The case for a *standardised mechanism* for (a) separating document payloads from the actual SOAP-envelope, (b) securing the documents uniformly (despite additional intra-document security measures) and (c) securing transport of SOAP-messages between applications in a GEM-context, therefore becomes relatively obvious.

What are the implications for heterogeneous interacting systems? The possible permutations for security configurations in m:n collaborations are obviously quite formidable. Considering the anticipated gravitation towards SOA’s, which are becoming increasingly Web services-based, it is pertinent to take note of how the Web Services Security standard (WS-Security)[2] addresses interoperability in the wider context of global cooperation. As with ebXML, *WS-Security could contribute to the multiplicity paradox* if explicit protocols to be used (underlining added for emphasis) are not pre-specified.

“This specification is intended to provide a flexible set of mechanisms that can be used to construct a range of security protocols; in other words this specification intentionally does not describe explicit fixed security protocols “. (Goals and Requirements, Lines 133-135).

Thus, the WS-Security standard, although targeting integration and interoperability, by being flexible (protocol-neutral), could actually promote “esoteric” configurations in a global context, if used injudiciously. This argument will be elaborated in Chapter 5.

ebXML: Concluding Remarks

As mentioned earlier, the ebXML initiative provides a reasonable point-of-departure for further consideration of a GEM. It will be seen in subsequent sections that other industry models for B2B-collaboration have steadily gravitated towards the ebXML standard.

Rawlings (2001[3]) suggests that the ebXML message service specification is the hardest one to assess, because it is the most established and goes furthest in satisfying a clearly defined market demand. It provides a way to bundle several XML-documents into one package, route the package to a destination within an organization, ensure the timely and reliable once-and-only delivery (addressing idempotence, unlike SMTP), and specifies security options. However, he considers the specification to be overly complex and suggests that there are several other approaches available. He considers the biggest problem with the messaging service specification to be that it was not produced by a well-known and trusted organization like the W3C. SOAP was incorporated into the ebXML architecture right at the end; yet, SOAP can be considered the de facto standard used for messaging within ebXML.

Rawlings (2001[3]) affirms the fact that ebXML has failed to enable interoperability. He states that it has been a dismal failure in addressing a common usage of XML attributes and a common approach to document structure. He argues that ebXML also failed in accomplishing one of its major goals: that of allowing small and medium sized enterprises (SME's) easy entrance into the world of e-business. This is due to its failure in defining business processes in which SME's participate. It has also made the integration of e-business connecting software with business applications more complicated for SME's. Rawlings also states that ebXML tried to address all the problems that arose with EDI and electronic business. It focussed on solving unknown problems instead of solving known problems. He points out three areas in which major work has yet to be done to enable interoperability: the message services (transport, routing, and packaging), business process definition, and business document definition.

These are critical to the realization of a GEM and will form a major part of the proposed model in Chapter 5.

Other Current B2B Models

RosettaNet, Microsoft's BizTalk, cXML and Covisint will be discussed in this section. The discussion of each will contain comparisons with ebXML, in relation to the CSF's discussed above, where possible.

RosettaNet

RosettaNet (RosettaNet, 2002) was a precursor to ebXML. It is an industry-specific example, comprising a consortium of more than 400 companies representing supply chain participants and solution providers in the semiconductor, electronic components, telecommunications, and information technology industry. The consortium is a non-profit group working to create, implement and promote open e-business process standards.

RosettaNet also uses XML as a common language for the electronic sharing of business information.

RosettaNet comprises

- a common vocabulary provided by *Dictionaries*,
- the *RosettaNet Implementation Framework (RNIF)* that specifies the grammar rules, and
- *Partner Interface Processes (PIP's)* that specify the business processes, in XML format between TP's.

The RosettaNet general Implementation Framework (RNIF) consists of domain-specific "ontologies" (business and technical dictionaries), transport-oriented specifications, and more than seventy specific process examples for supply chain processes (partner interface processes, or PIPs), for example, the update of a purchase order. Instead of describing a generic language for process choreography, the designers of the RosettaNet PIPs chose to specify specific modes of collaboration that may occur in a supply chain scenario.

PIP's describe the activities, decisions and partner role collaborations that comprise a business transaction between TP's. They are used to control the business logic, message flow and content. The time, security, authentication and performance constraints of these collaborations are also specified in PIP's. RosettaNet PIP's are structured as follows: *Seven groups of core business processes* form the basis of all business collaboration. Each group is divided into *segments* and each segment contains *individual PIP's*.

RosettaNet *Dictionaries* are comprised as follows:

- The *RosettaNet Business Dictionary* specifies the properties for defining each business transaction between TP's.
- The *RosettaNet Technical Dictionary* specifies properties for defining products and services.

The *RNIF* specifies common exchange protocols, in XML, and covers: transport, routing and packaging, security, signals and TP agreement. This forms the basis for the implementation of PIP's.

This corresponds well to the requirements for *interoperable collaboration* (the pre-specification of structure and coordination). Since the granularity allowed by PIP's and the RNIF is quite fine – business processes may be specified to be usable between just two TP's, even electronic monopolies can be configured; hence, collaboration *types* are also catered for.

Like ebXML, RosettaNet promotes open e-business standards based on XML messaging. Both standards define support for basic features of B2B integration, which include:

- synchronous and asynchronous communication,
- independence from communication protocols, and
- quality of service.

How does RosettaNet differ from ebXML? They differ in more user-oriented features, like public availability, complexity and effort, flexibility and dynamic changing, tools and guidelines support:

- RosettaNet is targeted at a specific industry segment and large enterprises, whereas ebXML is aimed at creating a global electronic market-space, accessible by businesses of all sizes and location.
- In RosettaNet, both business processes and the data aspects of B2B collaboration are standardized. ebXML does not define specific business processes; TP's are simply provided with the means and tools to specify business processes on their own.
- RosettaNet provides PIP's for building compliant applications, while ebXML had envisaged a set of Core Components which failed to be realized.
- RosettaNet does not specify a registry, as is found in ebXML

(Pušnik, Juriè, Herièko, Rozman, & Šumak, 2003).

In terms of how these models can be applied to a GEM framework, the following is suggested:

The RNIF model can be used to create a GEM-IF for the minimalist-business-pattern suggested in Chapter 2. The RosettaNet dictionaries and ontology are suitable foundations for implementing the metadata nomenclature, the common semantics and common vocabulary. The GEM-equivalent of PIP's can be exposed as Web services.
A consolidated picture will be presented in Chapter 5.

Microsoft's BizTalk

Microsoft's BizTalk (www.BizTalk.org) provides tools, products and services to implement B2B electronic commerce via three means: a BizTalk Framework, a BizTalk web site and the BizTalk Server product.

The BizTalk Framework is designed primarily for B2B electronic exchanges and does not address all phases of e-business. As with ebXML, no specific business processes are specified. The BizTalk Framework also uses XML and SOAP for communication between TP's. The web site is an online resource for businesses to access the BizTalk Framework and allows TP's to exchange documents with each other. This is similar in many ways to the ebXML Registry. The BizTalk Server allows both, disparate systems within an organization and those in TP systems, to communicate securely with each other. The Server tracks business documents to ensure reliable transfer of data. BizTalk also automates the changes that occur within TP systems as business processes change and develop (Lowe, 2001). The Orchestration Designer tool provided by BizTalk allows for the creation of workflows that determine the routing of messages based on various conditions and criteria. An Orchestration workflow generally represents an already established business process, unless one is using BizTalk to automate a previously manual process. Ideally, business analysts would use the Orchestration Designer to model the flow of information as it directly relates to the existing business processes. However, understanding the diagramming symbol conventions used by the BizTalk Orchestration Designer is required; these are different from those used by common process-mapping tools. In order for business analysts to be involved at this level, a standards document may need to be created which defines how existing process models are to be represented using BizTalk Orchestration (Erl, 2004).

Once an incoming document is received into a BizTalk system, it often needs to be transformed to meet specific business or application formats. BizTalk Server *Channels* can perform a number of operations, including document transformation and logging. A Channel is set up to receive incoming documents that conform to a specific document schema, which defines the data structure within the document. When the document arrives through a Receive function, a Channel can log the whole document, or any portion of it, to the *BizTalk Document Tracking* database, for later analysis, or transform the document to conform to the schema of the target application. *BizTalk Messaging Manager* is used to manage Channels via a graphical user interface (GUI) (Microsoft, 2002).

Unlike ebXML, BizTalk is not an open standard but a commercial product directly supported by Microsoft. ebXML allows businesses to build on existing open technologies and provides for plug-and-play interoperability, while BizTalk offers an integrated, vendor-specific solution (Chung, 2002).

As with ebXML, each TP agreement could be unique as in the electronic monopoly type defined in Chapter 2. This suggests multiple agreements (with concomitant technical configurations), multiple business processes, multiple business documents, and so forth, if BizTalk were to be used in a multilateral collaboration scenario. The ramifications include: higher electronic integration is required; sustainable collaboration will require a greater degree of pre-specification (of structure and coordination) or incur a greater risk of conflict arising; interoperability will be difficult to implement as the number of participants grows larger.

However, BizTalk represents a fundamental collaboration platform – in pure application terms – for implementing a GEM. The client and server applications can mimic the capabilities inherent in both ebXML and RosettaNet paradigms. If the software were available as freeware and was platform-independent, it would be excellent in the context of the GEM. On the other hand, once a global standard for documents and business process choreography has been established, this could easily be implemented on the BizTalk Framework.

cXML

Commerce XML (cXML) is another open XML-based standard. It is a joint effort, initiated in 1999, by more than forty companies with the common goal of reducing online business costs, by standardizing the electronic transfer of business documents over the Internet. It is primarily used by trading parties that buy in large quantities.

Standardized XML-document type definitions (DTD's) specify the contents of cXML-documents, which are electronic versions of traditional hardcopy documents used in business. The most commonly-used cXML-documents are:

- *Catalogues,* which are documents that describe the business processes provided by TP's.
- *PunchOut*, which is a set of rules that allow the exchange of cXML messages between TP's. It is divided into:
    - Unlike the static catalogues that only provide business processes, *Procurement punchOut* provides dynamic and interactive catalogues that provide options for the automatic ordering of products or services.
    - *punchOut Chaining* is an addition to procurement punchOut in that it allows all TP's involved in the delivery of a specific order to be notified.
    - *Provider punchOut* allows TP's to assist each other in the completion of a specific transaction. An example of this would be a TP doing a credit card check on another TP. A third TP would be required to authenticate the validity of the buying TP's credit card.
- **Purchase orders**: are requests made by one TP to another for completion of a contract.

(cXML, 2003).

This is a comprehensive set of business documents for which the metadata are pre-specified for all participants, as is envisaged for a GEM. Two major detractors of cXML are: the target users are businesses that deal in large purchase-quantities, and the use of DTD'd instead of XSD's (although the latter conversion could be easily brought about).

By abstracting some of the functionality into the actual documents, the complexity of the individual system-configurations is reduced. This would be a major advantage in a GEM if the component for generating specific standardized XSD's was provided to all participants; this would facilitate the ease-of-use aspect for SME's.

Synthesis

In Figure 4.4 below, an approximate timeline for the evolution of some of the important B2B-collaboration models is illustrated. Electronic Data Interchange (EDI) was initiated by General Motors in 1970, but became a global (UN/EDIFACT) standard in 1997 (incorporating the US X12 standards). The advent of the Internet resulted in EDI being migrated to the Internet (from private networks). The ISO/IEC Open-EDI reference model was released in 1997. The XML/EDI group followed with a specification in 1998. The Open Buying on the Internet (OBI) specification was released in 1999, but was based on EDI X12 rather than XML. (Daum and Schiller, 2000:256-260).

The ebXML initiative started in November 1999 and was completed in May 2001. ebXML used similar fundamentals to Microsoft's BizTalk and RosettaNet. Other contemporary XML initiatives include those mentioned in the preceding sub-sections. Currently, as mentioned in the conclusion of Chapter 2, ebXML is widely accepted as the all-embracing super-standard. Efforts to unify their B2B collaboration models with ebXML are reported on, inter alia, the web sites of BizTalk, RosettaNet and Covisint.

Pattern

Therefore, the point-of-departure for any future GEM model would, logically, have to be ebXML. The supposition at this point, however, is that ebXML in its current form is deficient as a standard.

Only two percent of businesses in the United States adopted the EDI standard. Even though the standard was regarded as functionally feasible, the costs of establishing bilateral business and technical agreement among a large number of prospective TP's, the cost of management of these agreements, and the cost of implementation, have led to the gradual demise of EDI (Ibid). Similarly, one of the architects of ebXML, M. C. Rawlings, argues that ebXML will not reach “critical mass” and justify the effort, as it fails to deliver on its original requirements specifications (for example, in ensuring interoperability and promoting ease-of-use for SMEs) (Rawlings1, 2001).

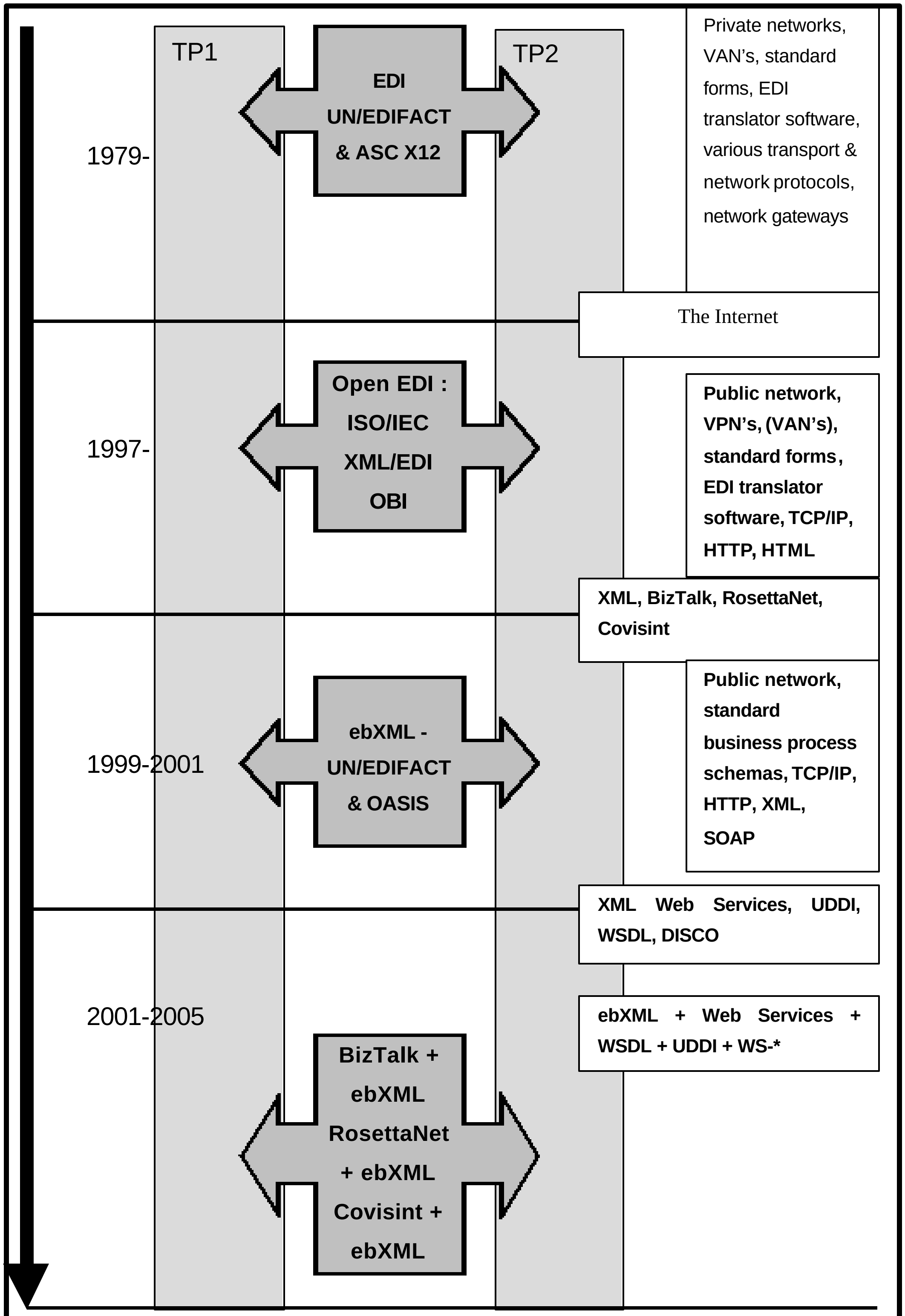
In this chapter some of the aspects of the envisaged GEM that can be considered CSF’s were outlined; many of these have not been adequately addressed by existing B2B-collaboration models. It is suggested here that the complex-systems nature of B2B-integration, as is emphasized by the multiplicity paradox, has been the main reason for the delay in arriving at a suitable GEM model.

In terms of the CSF’s for interoperability, gleaned from existing models, GEM application interfaces *must* therefore be created according to a single, comprehensive, *pre-specified* interoperability (W3C) standard comprising *precise* open standards, such as:

- a common data description language (XML);
- a common character encoding (UNICODE);
- a common network protocol (Internet Protocol, IP);
- a common transport protocol, (HTTP or SMTP over TCP);
- a common messaging protocol, Simple Object Access Protocol (SOAP) with Multimedia Internet Mail Extensions (MIME) attachments;
- a common security model, specified using standards such as WS-Security, WS-SecureConversation and WS-Policy, utilizing specific security protocols, for example, Secure Internet Protocol (IPSec) for internal networks, XML encryption for encrypting parts of the XML attachment at the application level, Advanced Encryption Standard (AES/Rijndael algorithm) for encryption, XML digital signatures for integrity control at the messaging level, and Security Assertion Markup Language (SAML) for Web service access control, and so forth;
- a standardized published nomenclature (taxonomy of business terms and meta-data), based on a common semantics and a common vocabulary;

- a minimalist-set of standardized business patterns (business process XSD documents and exchange-choreography);
- common business documents based on the common nomenclature;
- a common message-handler (the Listener module discussed in Chapter 3) core-component specification for incorporation into participating systems; and
- standardized endpoint services, which are able to process the standardized documents in standardized orchestrated business processes.

A model for the structure and operation of the GEM can then be established based on these common standards.


TP1
TP2
1979-
EDI
UN/EDIFACT
& ASC X12
Private networks, VAN's, standard forms, EDI translator software, various transport & network protocols, network gateways
The Internet
1997-
Open EDI :
ISO/IEC
XML/EDI
OBI
Public network, VPN's, (VAN's), standard forms, EDI translator software, TCP/IP, HTTP, HTML
XML, BizTalk, RosettaNet, Covisint
1999-2001
ebXML -
UN/EDIFACT
& OASIS
Public network, standard business process schemas, TCP/IP, HTTP, XML, SOAP
XML Web Services, UDDI, WSDL, DISCO
2001-2005
ebXML + Web Services + WSDL + UDDI + WS-*
BizTalk +
ebXML
RosettaNet
+ ebXML
Covisint +
ebXML

Key: EDI=Electronic Data Interchange; UN/EDIFACT=United Nations body for EDI Trade Facilitation; ASC X12=Accredited Stds Committee X12 standard; Open-EDI=Internet-based EDI; XML/EDI standard, derived from Open EDI; OBI=Open Buying on the Internet standard; UN/CEFACT: UN Centre for Trade Facilitation and Electronic Business; OASIS=Organization

Fig. 4.4. The progression of B2B-integration standards

Chapter 5

A Proposed Framework for a GEM

Introduction

As acceded to in Chapter 1, the notion of a GEM is a hypothetical construct. However, similar to (less-conceivable) frameworks for alleviating poverty world-wide or for achieving peace on a global scale, one often arrives at theoretical antecedents for achieving the desired ideal (or near-ideal). This, in itself, often provides guidelines for smaller-scoped endeavours. Thus, one might apply the framework discussed in this chapter within a smaller context, such as a particular user-community within a particular collaboration focus area. However, by considering the wider (potential-GEM) integration implications, such a user-community would be able to interact with another user-community that uses the same guidelines.

As suggested in previous chapters, interoperability in a GEM resides in the application-to-application interface mechanism employed. Based on the evidence examined in the previous chapters, it is concluded that the envisaged-GEM should be based on a Web Services-Services-Oriented Architecture (WS-SOA) in which (XML-based) SOAP provides the globally-interoperable messaging-interface and XML provides the globally-process-able data format. However, unlike ebXML, BizTalk or RosettaNet, this framework is an attempt to avert the multiplicity-paradox crisis implicit in protocol-neutral paradigms. It purposely eschews non-deterministic complexity – in avoiding foreseeable attempts to resolve gross disparities resulting from laissez faire neutrality (masked by the availability of multiple standards, the various permutations of which are antithetical to global interoperability).

Also deduced from previous chapters is the fact that it should be allowed for business services to be architected according to individual business requirements from standardized headers/programming-interfaces and method-signatures. Participants should be able to extend and override the functionality of these standardized interfaces and methods, but a Global Message Handler (GMH) module at both ends should subscribe to a global specification. This is illustrated in Figure 5.1 below. The GMH ultimately provides the crucial (deterministic) “frontline” global-interoperability application-interface. The GMH should, therefore, be implemented as a Web service, for which the interface contract needs to be described by a GEM-standardized WSDL specification.

Nevertheless, the broader GEM-context in which the GMH is intended to operate must be conducive to the broader global-integration objective – within the constraints of structural predictability - as must be the approach adopted for superimposing a GEM onto the existing (mainly dyadic) B2B-collaboration status quo.

In essence, patterns inferred in previous chapters suggest that *the common collaboration (virtual) platform* should comprise the following parts (essentially the *a priori* critical success factors):

a) An interoperable *standardized message format* and *an interoperable standard message-delivery infrastructure* encompassing *an application-level message-handler;* these are entirely platform-independent and constitute the *technical (standardized) GEM interoperability-interface.*

b) A *hierarchically-standardized business infrastructure*, pre-specified by experts in the business domain. The top-most tier of the business infrastructure should include:

    i. A minimalist, generic, *standardized business-pattern,* the basic buying-selling commercial business-pattern, comprising

        1) a *pre-specification for standardized generic business processes* (for example, placing an order), pertinent to the minimalist business-pattern;

        2) the *specific atomic activities* (for example, submitting an order-form, acknowledgement of receipt of an order-form, error-handling messages, response messages, and so forth), pertinent to the minimalist business-pattern;

        3) *a standardized generic document-set* (including, for example, a standardized order-form and a standardized invoice) based on

        4) *standardized metadata* (common nomenclature and common semantics used in describing data-fields, for example, elements and attributes in the order-form XSD) and the corresponding

        5) *standard choreography* for atomic activities (for example, acknowledgement of receipt of a form, error-handling messages, response messages, and so forth, within a particular transactional conversation);

    ii. *appropriate data-processing applications*, on the consumer and provider sides, designed to function in accordance with 1) to 5) above (and being pertinent to the minimalist business-pattern).

c) A *standard security model* for securing the messages, the message-delivery and the application endpoints.

d) A *Registry/Repository mechanism* for the discovery of participants and for downloading specifications would expedite the dissemination of the framework specifications. Registries would be arranged in *a hierarchy of related servers*, bearing content in a hierarchical order. At the highest level, generic GEM specifications would occur from which specifications may be “inherited” by lower, specific business domains and sub-domains. Elaboration of specifications at lower levels will also occur in the same manner as described for the top-most tier. This is essential in order to maintain uniformity for integration.

As argued throughout this thesis, *cognisance of the structural complexity of the GEM must be foremost in design-considerations*. As discussed in Chapter 2, in order to maintain a set of S (states) for which the states are deterministic, the sets of K (state variables) and L (laws) must be kept manageable. However, the interoperability state-variables as well as the statements (laws) defining the anatomy (how the GEM will be structured) and physiology (how the GEM will function) must be sufficiently comprehensive to achieve GEM-integration.

Given the diverse nature of the variables in K in current B2B-collaborations, which exist in disparate, isolated pockets, *a staged, hierarchical, minimalist approach* is advocated. XML, SOAP, WS-* standards, HTTP, TCP/IP and other W3C open standards are available for formulating a GEM-specification based on the critical success factors listed above. Newer specifications for extending the functionality of SOAP and Web services are being added on a regular basis. This trend will presumably continue – and, if the GEM-context were to be realized - would facilitate the evolution of the GEM to the point where other dyadic collaborations would gravitate towards incorporating open standards, and ultimately be subsumed within the uniform specifications for the GEM.

Subsequent sections will explore the proposed GEM framework in terms of the CSF's listed above.

Figure 4.1 below illustrates the position of the GMH in an "anatomical" view of the Provider-side application. The fundamental "Listener" component of the Web service is elaborated into the actual GMH (which itself is a Web service). The dashed lines are intended to illustrate the "magnification" of each preceding component.

Thus, the progression is as follows: the common application interface between Provider and Consumer is a Web service, of which the Listener component can be elaborated into a Global Message Handler (GMH) with functions beyond simply acting as a listener-daemon. The GMH both receives and generates standardized SOAP-envelopes, which it processes in a specific manner. The standard SOAP-envelope, in fact, is mainly intended to invoke, by RPC – and in a standardised fashion - the GMH-service on the recipient side. Therefore, the GMH-service should have a common, standardized nomenclature.

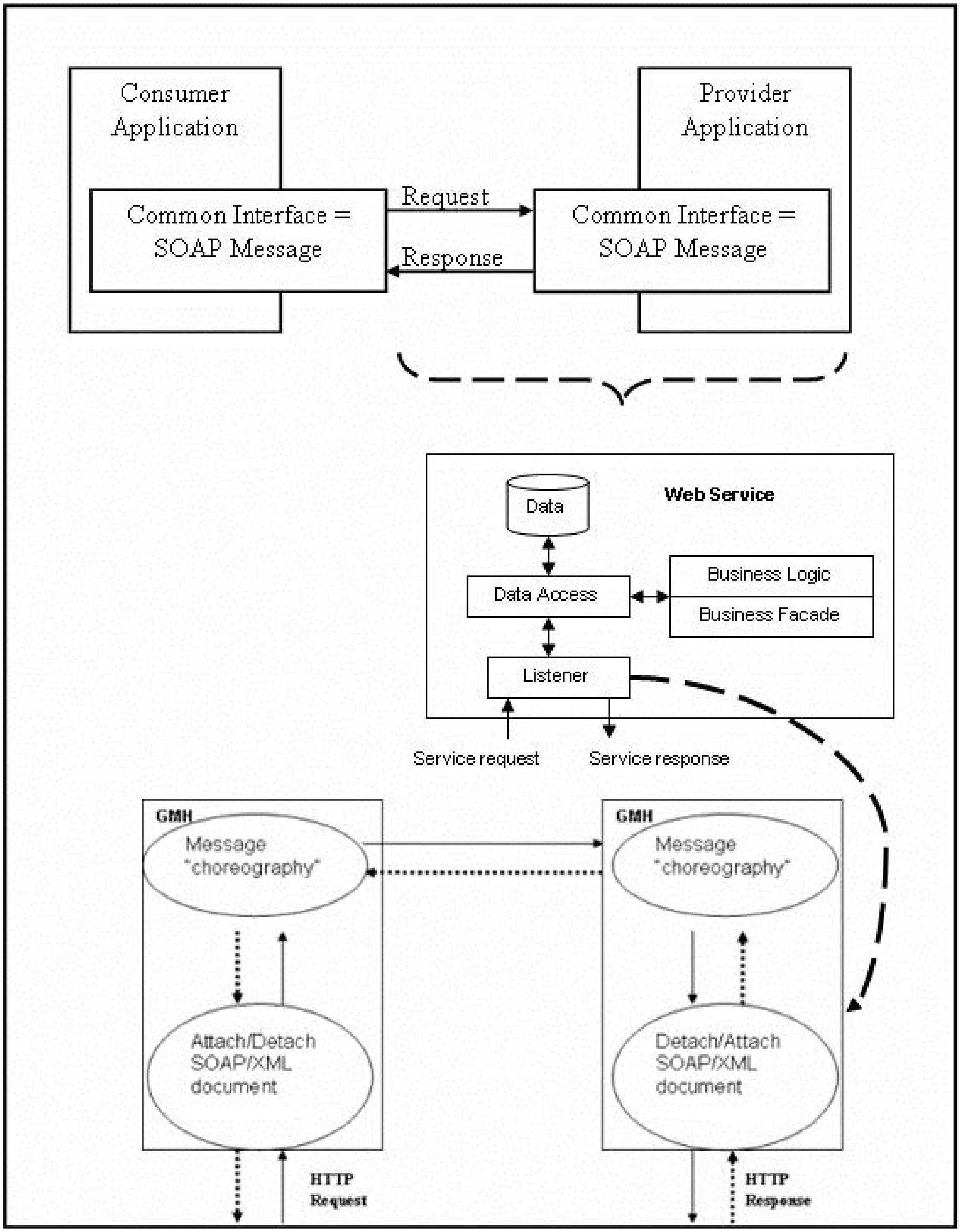


Figure 4.1. A Global Message Handler provides the active GEM interface

The CSF's alluded to above will be discussed in further detail in the sections below.

A Minimalist-Business-Pattern Approach

As discussed in previous chapters, the *essential (minimalist-approach) business pattern* to be used in the incipient-GEM comprises the following business processes: placing an order, fulfilling an order and payment for the order. The *motivations* for this minimalist approach include: to avoid the multiplicity paradox, to ensure that the GEM has a predictable structure (by reducing the “fuzziness” of K), to facilitate easy entry for SME’s and to allow for centralized (W3C) control. Therefore, the *key constraints* on deciding to what extent each of the listed business processes should be elaborated (for example, viewing and selecting from a product-catalogue, completing an order-form, updating/cancelling an order, returning (part of) an order, issuing credit notes and debit notes, and so forth) are comprised of the same factors. An obvious rule-of-thumb that can be applied here is: *the broader the specification, the greater the required facilitation*. By facilitation is implied the provision of (programming) object libraries, in addition to the global specifications; this would be similar, in principle, to ebXML’s Core Components library. This would have the effect of removing configuration-items (patterns that influence the dynamics of the structural complexity of the entity).

One might ask: How practicable is this notion of facilitation? The W3C SOAP 1.2 specification specifies an XML Infoset (metadata and structure for the elements, attributes, data-types, and so forth) (XML Information Set, 2004), and explains how a SOAP-processor should interpret and execute documents/messages of this object type. This specification has been used in developing ubiquitous applications, such as Web browsers, to enable the functionality implicit in the specification. Developers of software-development platforms, such as the .NET IDE (Visual Studio .NET), have also incorporated object libraries (such as Microsoft’s Web services Extensions 1, 2 and 3) to facilitate the building of software applications based on the SOAP 1.2 standard. The scenario depicted here holds true for most, if not all, W3C specifications.

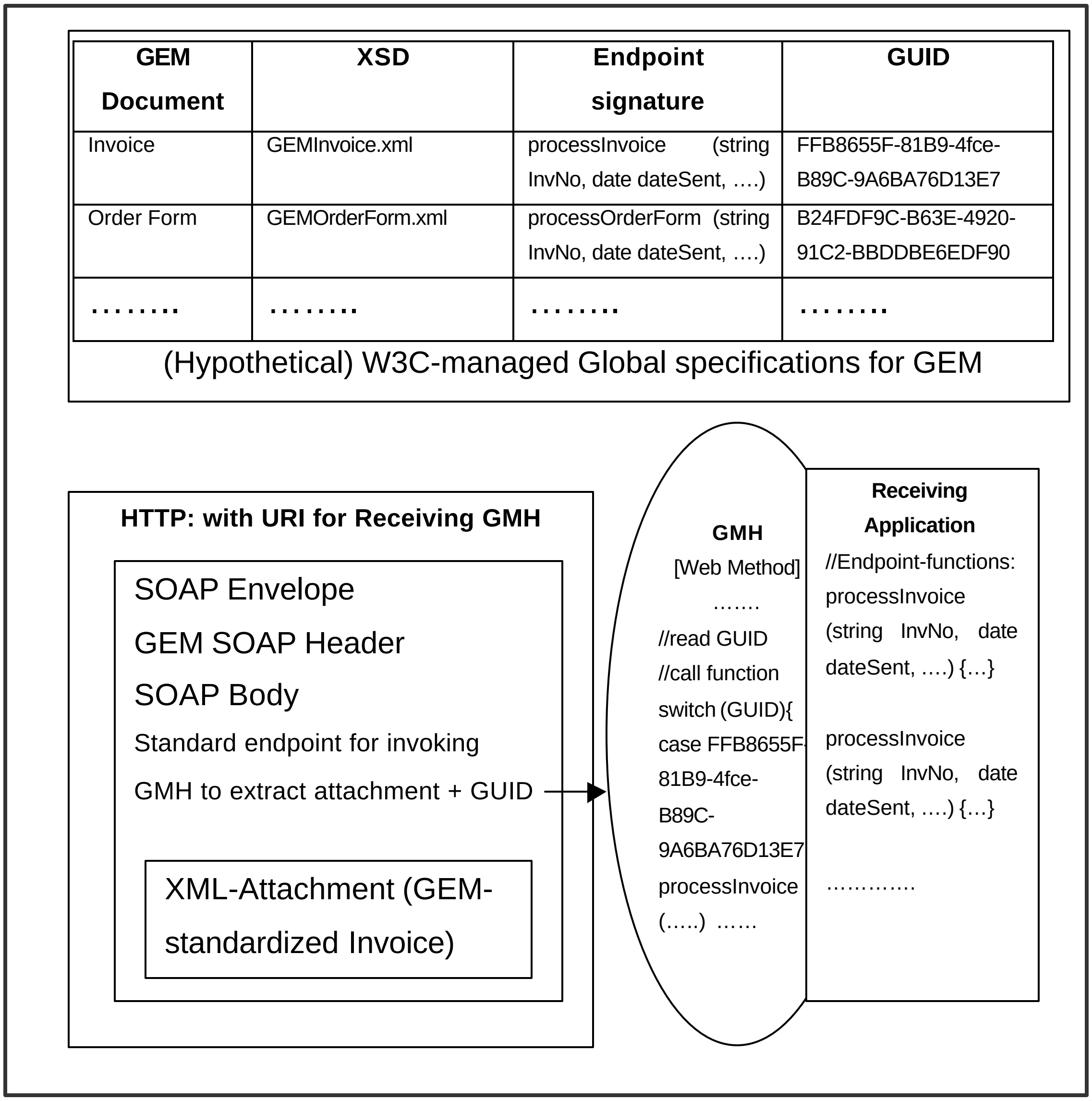
Thus, it can be argued, that the W3C can specify the minimalist-business-pattern for a GEM, in a similar manner. The specification would, ideally, encompass all the requirements specified in CSF (c) in the list provided in the Introduction section of this chapter. In fact, existing W3C specifications for specifying business processes, namely BPEL4WS, and transactions, notably, WS-Coordination, WS-AtomicTransaction and WS-BusinessActivity, could be used to specify a single global standard for the minimalist pattern.

At this stage, one might be tempted to conclude that the minimalist approach is tantamount to a simple extrapolation of Business-to-Consumer-collaboration (B2C-collaboration). Indeed, the simplicity implicit in B2C-collaboration is one of its important CSF's, as is the fact that the Provider unilaterally determines the standards and technologies used. The incipient-GEM objective in fact should not transcend the simple dyadic B2C-collaboration by too great a margin in terms of complexity. Of course, there is no reason why the client cannot be a single consumer, or why B2C-collaboration cannot use the GEM. In other words, *the envisaged-GEM does not preclude B2C-collaboration*; provision must simply be made to establish the *contractual nature* of B2B-relationships (comprising dealership agreements, such as availability, minimum order quantities, fulfilment and logistics arrangements, payment arrangements, discount rates, and so forth) beforehand.

Thus, *there is room for a Collaboration Partner Agreement similar to the one used in the ebXML model, but without the need to resolve differences in protocol-profiles and other technical variables*; the latter are abstracted into the common interface represented by the GMH. Only business details (not technical ones) are required in the "CPA". It is incumbent upon each Provider to provide the individualized logic for handling "CPA" contents, in non-GMH modules of the service. For instance, if Company X wishes to categorize clients and allocate discount differentials according to a specific company policy, then their internal system should reflect this. This *should not impact on the technical construction and operation of the GMH*; no data or documents beyond the GEM standard should alter the functionality of the GMH within the GEM. The CPA-form could, however, be a GEM-standardized XML-form based on a standardised trading partner agreement XSD. It could even be transferred via the GMH as an XML-attachment in a SOAP-message. Like all GEM-standardized XML-attachments, it could have a corresponding globally unique identifier (GUID) pointing to a corresponding standardized endpoint-function header. In Object-Oriented Programming, "header" refers to the "Interface" type, similar to a class; the identifier, member variables and data-types are provided, but the implementation is left to the developer. A default implementation is often provided, but can be overridden. Similarly, headers can be provided for endpoint-functions. Endpoint-functions would probably be best implemented as virtual methods (or functions), which can be overridden by the user. By providing the header information, the function-identifier, the data-types of parameters, sequence of parameters and number of parameters are kept standardized.

Thus, as illustrated in Figure 4.2, if a company should send a particular XML-attachment (based on a standardized XSD) for, say, an invoice, the receiving-GMH would read the corresponding GUID and call the (standardized) endpoint-function (header) that can process it (in a company-specific implementation); the recipient application may even simply be designed to place the document in a repository for manual processing. The success of the interoperability depends on the use of a single set of standardized documents, standardized endpoint-function signatures and corresponding standardized GUID's, and standardized choreography.

The table in Figure 4.2 illustrates the relationship between the GEM documents, the corresponding GUID's, the corresponding XML schema documents and the corresponding endpoint-function headers; all of these are intended to be standardized.
The illustration indicates that the GMH is invoked by SOAP-RPC; the business data is contained in an attachment (typically, a standardized XML document). The SOAP-envelope is also indicated as being carried by HTTP; whether this is the obvious choice is debated in the next section.



GEM Document	XSD	Endpoint signature	GUID
Invoice	GEMInvoice.xml	processInvoice (string InvNo, date dateSent, ….)	FFB8655F-81B9-4fce-B89C-9A6BA76D13E7
Order Form	GEMOrderForm.xml	processOrderForm (string InvNo, date dateSent, ….)	B24FDF9C-B63E-4920-91C2-BBDDBE6EDF90
……..	……..	……..	……..

(Hypothetical) W3C-managed Global specifications for GEM

HTTP: with URI for Receiving GMH

SOAP Envelope

GEM SOAP Header

SOAP Body

Standard endpoint for invoking GMH to extract attachment + GUID

XML-Attachment (GEM-standardized Invoice)

GMH

```
[Web Method]
…….
//read GUID
//call function
switch (GUID){
case FFB8655F-
81B9-4fce-
B89C-
9A6BA76D13E7
processInvoice
(…..) ……
```

Receiving Application

```
//Endpoint-functions:
processInvoice
(string InvNo, date
dateSent, ….) {…}

processInvoice
(string InvNo, date
dateSent, ….) {…}

…………
```

Figure 4.2 How the hypothetical GEM standard would operate

The precise standardized specification for the minimalist-business-pattern set of endpoint collaboration-functions, the corresponding business processes and atomic business activities, and the corresponding document schemas (XSD's) can be inferred from existing B2B-collaboration models, such as BizTalk, RosettaNet, cXML and ebXML. The following set of collaboration-functions does not represent an exhaustive list – which would more-appropriately be drawn up by business analysts - and is presented here in generic terms: Searching for relevant suppliers; applying for collaboration; viewing product catalogues; negotiating/querying a collaboration/transaction agreement (including availability, pricing, logistics and payment arrangements); placing an order; updating an order; returning part of, or an entire, order; acknowledgement of message-received; acknowledgement of document-received; notification of order details; notification of order status; sending an invoice; sending an account statement; requesting a transaction-history statement; requesting a transaction-history statement; submitting a product complaint; submitting a service complaint; notification of transaction status; and so forth. Any number of these, or variations of them, could be standardized in terms of documents, semantics and nomenclature, atomic activities and choreography, ensuring that the common-business-process CSF (implying Consumer and Provider applications acting complementarily, in a seamless fashion) is complied with.

Since all participants would be using the same GMH (the same protocol-profile and settings), the same documents, the same message-format and the same set of business processes, the sets of S and L are “fixed”. In this ideal scenario, the structural distance between entities in the GEM tends to one, since they are all based on the same standard and each GEM-participant entity comprises virtually the same patterns (little or no “fuzziness”), resulting in a high degree of structural predictability in the emergence.

The business-information processing infrastructure discussion will be continued in a subsequent section.

The Message Format and Message-Delivery Infrastructure

Fundamental (Experiential) Issues

This section examines some of the more salient issues which arose during the course of developing a proof-of-concept infrastructure.

Synchronous Processing versus Asynchronous Processing

For reasons delineated in previous sections and chapters, GEM-messages would typically be SOAP-messages transferred over HTTP-connections. SOAP-Web services provide a synchronous means for sending parameter-values to a remote method and receiving real-time responses. However, synchronous processing in a GEM-context has obvious ramifications for complexity (not to mention bandwidth) considerations. This evokes the question:

Should the XML-data be processed synchronously – in real-time, using the request-response SOAP MEP – or should it be processed asynchronously, allowing time for validation and ratification of the input-data and the response?

Synchronous processing involves multiple exchanges of business process data in real-time. The messages would have to be packaged in a standardized manner for SOAP-handling and for security, on both sides, over a persistent request-response MEP. SOAP does not specify algorithms for the use of optimistic concurrency, roll back, or other transaction-processing techniques. However, WS-AtomicTransaction (W3C, 2004[6]), makes provision for such features; additional elements have to be added to the SOAP-envelope to enable this functionality. The more complex the standardized business processes are specified (in terms of processing and number of exchanges, for instance), the greater the potential for problems in synchronous transmission. Delayed business processes are one such problem. Further, maintaining state is considered not “Web friendly” in terms of REpresentational State Transfer (REST) architecture (Chatterjee and Webber, 2004: 96). However, a major stumbling block arises in using XML-data in asynchronous processing. In synchronous messaging, only the Web server need have a valid Internet address (IP address and host name); the response messages are simply returned via the open link. In asynchronous messaging, the response messages have to be sent to a server with a valid Internet-Assigned Numbers Authority (IANA) address, to be accessible via the Internet Domain Name System (DNS); this would require every consumer-participant to have either an HTTP-server or an SMTP-server available on the Internet. Client systems are, however, likely to act from behind a firewall on which Network Address Translation (NAT) is configured and where IP-addresses are allocated dynamically.

Thus, each SOAP-request sent to the Provider-GMH would require a synchronous SOAP-response (request-response MEP) or each Client would have to make provision for an Internet server for asynchronous processing. The HTTP request-response MEP used in B2C-transactions, generally based on the minimalist business pattern described earlier provides a strong case for success in synchronous environments. It is however, *incumbent on clients to initiate exchanges* – as part of the standardized choreography for specific business activities – to establish the connection, in each transaction. Further, each exchange must result in standardized, orchestrated closure (in terms of choreography) and also be able to link to the next exchange involving the same client, continuing the same transaction if necessary (be transaction-bound). This is tantamount to “continuing the same conversation” between the same participants upon each re-connection.

SOAP-RPC/Encoded versus Literal/Document versus XOP Attachments

One of the extolled virtues of SOAP is its RPC-functionality across disparate platforms, which is a direct consequence of its XML structure. Communication between XML Web services and their clients is dictated primarily by two specifications: SOAP and WSDL. SOAP defines a formatting scheme for the data that appears beneath the Body element and a formatting scheme for how parameters are formatted within that Body element. The former is referred to as *SOAP section 7* or simply *RPC*. The latter is referred to as *SOAP section 5* or simply *Encoded*. WSDL, which is used to describe the SOAP messages expected by an XML Web service, allows XML Web services to indicate that they accept RPC messages with encoded parameters, but it also defines two other terms: *Literal* and *Document*. Literal, like Encoded, refers to how the parameters are formatted, by mapping the parameters to XML elements using a predefined XSD schema for describing each parameter. Document, like RPC, refers to how the overall Body element is formatted. The formatting rules of the SOAP specification in sections 5 and 7 allow for variations. Thus, a recipient of a SOAP request using the SOAP encoding rules would have to handle all the possible variations. By defining an XSD schema for the parameters used in XML Web service methods, the multiplicity paradox (as described in Chapter 3) could be obviated.

Since the parameters to an XML Web service method can make up the majority of the data passed in a SOAP request or response, how the parameters are mapped to XML elements determines how the XML document will look.

The synchronous-versus-asynchronous (processing) dilemma is heightened by the following point-of-consideration. An XML-document provides a means for sending parameter-values to a remote method allowing for either synchronous or asynchronous (offline) processing of the data. XML business process data could therefore simply be incorporated within a pre-specified SOAP-envelope, as a SOAP-document *attachment* to the external SOAP-envelope (W3C, 2002[2,1]), rather than for it to comprise part of the actual body of the SOAP-envelope body itself . (The document type would dictate whether the attachment is either encoded/literal or document/RPC). The external SOAP-envelope would thus provide only the RPC boundary layer to the receiving-GMH. The receiving-GMH would relay the attachment to the appropriate endpoint-service, for either synchronous or asynchronous handling.

In asynchronous processing, both the input and the output can be validated and ratified offline before responding, on the receiving-end. In synchronous processing, there could be a performance drop in using SOAP-attachments as opposed to XML-data being embedded in the SOAP-body. If the data were carried in the external SOAP-body, the RPC-method for processing the data could have direct access to the data (SOAP-section 7-encoded RPC). The meta-data for the SOAP-body would thus have to be standardized (and its namespace referenced), and the corresponding RPC method-name would therefore need to be standardized, for each standardized SOAP-document type (order-form, invoice, and so forth). In either case, the SOAP-header would have to be standardized for GEM purposes, for example, by adding features such as WS-AtomicTransaction. Each standardized document (order-form, invoice, and so forth) would be specified as a SOAP-document, incorporating the standardized SOAP-header.

The advantages of using XML-attachments would include separating the specifications for the standardized SOAP-envelope and the document. Not all document XSD’s would have to be altered if the SOAP-envelope specifications were to change. Further, the SOAP-envelope would simply be used to invoke the message handler and to indicate the standardized features being included. This would also allow the entire XML-document to be encrypted (with XML-Encryption) and be signed with XML-Digital Signature, as a standardized GEM security mechanism. Separating the XML-data from the SOAP-envelope would allow separate standardisation for the SOAP-envelope and the XML-documents. Also, REST recommends that SOAP be used for its Literal rather than its RPC functionality (Chatterjee and Webber, 2004:120).

Further, this mechanism would accommodate strategic 1:n and 1:1 dyadic relationships beyond the GEM-context, by allowing non-GEM-standardized document-formats to be attached as well. For GEM purposes, the attachments would be GEM-standardized XML-documents (for example, an order request) based on pre-specified XSD's. The GMH-module of the receiving-applications would simply identify and remove the XML-attachment(s) for specific endpoint-processing. Each document type could have a corresponding GEM-specified GUID (or equivalent) for the function that handles the document, to allow for automated processing of individual document types, for example, a specific component could be called to process an order document, as discussed in terms of Figure 4.2 above.

Resolving the choices

The choice-matrix may be represented as follows:

	Synchronous processing	Asynchronous processing
Literal/encoded Document/RPC XML-data in SOAP-body	1. Over HTTP: Client has to initiate each session; many sessions may be required, which complicates orchestration. Idempotence and persistence issues.	3. Over HTTP: Requires queuing by every sending and receiving application. Requires each participant to have a Web server (static IP address)
Literal/encoded Document/RPC XML-data in attachment	2. Over HTTP: Requires attachment to be persisted on receiving end (non-repudiation). Possible	4. Over SMTP: Standardized SOAP-Envelope (containing XML-document

	performance issues. Idempotence issues.	attachment) is attached to e-mail message.

Table: SOAP Envelope options in a GEM

After careful consideration, the following is offered as a possible means to resolve the available choices. If the advantages of asynchronous processing are to be sought, then an SMTP model would provide the better infrastructure (Option 4 in the table). Its store-and-forwarding mechanism makes it a better candidate than HTTP in this regard; for asynchronous SOAP-over-HTTP (Option 3), queuing would have to be implemented. The GMH for the SMTP model would be similar in function to that of the HTTP model (a standardized SOAP-envelope with XML-attachments); the GMH would simply process/generate SMTP messages rather than HTTP ones.

For synchronous processing, XML-data in an attachment is preferred. The primary reason for this is the use of a GMH as the single interface for interoperability, via which integration can be effected (by way of the other CSF's mentioned). Using SOAP-envelopes, as the direct means for invoking specific RPC's, devolves the first line interoperability-interface function to the standardized SOAP-envelope, with its own Infoset (XML Information Set, 2004) and functionality. Since SOAP-documents are merely (extended) XML documents, a SOAP-document could also be attached, forming a SOAP-envelope within a SOAP-envelope. (In cases where the internal SOAP document is intended for invoking a Web method by RPC, the GMH would use a URI reference rather than a GUID).

The Provider-side GMH, therefore, would provide for connecting to specific endpoint-services, which handle standardized client-side requests via standardized client-side XML-documents (for example, an order-form) as either document-literal XML documents or SOAP-RPC documents. The client-side GMH, on the other hand, would provide for connecting to specific endpoint-services that handle standardized server-side responses via similar, standardized server-side XML-documents (for example, an invoice). The GMH on both sides would be capable of generating standardized SOAP messages – some with user-selected standardized

XML-attachments; others simply as an acknowledgement or a response to an error – for which only the receiving host URI (Uniform Resource Identifier) is required to be entered by the user; the XML-attachments would be chosen and completed via user interface forms.

Which part of the GEM Infrastructure should be local?

Another issue to consider was whether the GMH and the pertinent endpoint services should be hosted on each participant’s own Web server, to allow for asynchronous and synchronous processing. Alternatively, should the GMH simply be called as a Web service by all participants that require its functionality?

In the latter case, each participant would, in typical Web service fashion, use a proxy (typically generated by development tools) through which the client-side application would invoke the GMH service. The GMH service (adding a standardised header, standardised attachment packaging/opening and relaying the attachment to endpoint services) could be hosted on several selected servers world-wide – even provided as a Grid Service. A major advantage would be the potential to centrally control the evolution of the specifications of the GMH. The major disadvantage would be the orchestration of asynchronous transaction activities, as discussed above. If this problem were to be resolved – in overcoming network address translation (NAT) restrictions, for instance – then this would be the obvious preferred-mode.

Since (Web services-based) endpoint services (in B2B-collaboration scenarios) would have to allow for Web access using a Web server interface, the GMH could quite conceivable be hosted locally. All internal workstations could, however, use GMH proxies to use the same locally-hosted GMH.

Additional Considerations

The SOAP-message excerpt below is from the example used in Chapter 2.

```
POST /Reservations?code=FT35ZBQ HTTP/1.1
Host: travelcompany.example.org
```

```
3.  Content-Type: application/soap+xml; charset="utf-8"
4.  Content-Length: nnnn
5.  <?xml version='1.0'?>
6.  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
7.  <env:Header>
8.  <t:transaction
9.  xmlns:t="http://thirdparty.example.org/transaction"
10. Env:encodingStyle="http://example.com/encoding"
11. Env:mustUnderstand="true" >
12. </t:transaction>
13. </env:Header>
14. <env:Body>
15. <m:chargeReservation
16. env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
17. xmlns:m="http://travelcompany.example.org/">
18. <m:reservation xmlns:m="http://travelcompany.example.org/reservation">
19. <m:code>FT35ZBQ</m:code>
20. …….
```

In the example above, the collaboration involves making a reservation with a travel company (in a previous message) and subsequently (this message) invoking a chargeReservation endpoint-function (line 15). Firstly, in the minimalist-business-pattern approach, the exchange of messages involving reservations would not necessarily be governed by the top-level generic GEM standard. Secondly, in the GEM, the SOAP-envelope header references (Infoset namespaces) would be standardized.

The endpoint URI (line 2) is indicated in the HTTP POST envelope and is generally inherited by the SOAP-envelope beneath. In order to comply with the common business process CSF, all such SOAP-requests and all corresponding SOAP-responses must be able to interact seamlessly (in an m x n way), in an automated fashion. For this to occur in the scenario depicted, standardized metadata (including the same endpoint-function signature) is required for each request-type. The actual implementation of the endpoint-function may vary from user-community to user-community or from company to company. *The endpoint-function could even call a legacy application, providing that the return-values and message-format conform to GEM-standardized specifications.*

As mentioned above, placing the data for processing in attached XML-documents promotes scalability and upgradeability of the standardized SOAP-envelope. This is in accordance with current modular, incremental development-methodologies. *Using this (XML-attachment) message format, the message-handler (Listener) could even poll both the SMTP server and the HTTP server for standardized attachments.*

This (XML-attachment) mechanism also allows a pre-specified security model to be imposed. A GEM-standardized SOAP-envelope could be used with a pre-specified standardized set of features, including WS-Security. The SOAP-envelope could be used to specify a reference to WS-Security, which in turn could be used to specify, for instance, that XML digital signatures will be used for ensuring the integrity of the attachment and that XML Encryption will be used to encrypt the attachment. The TCP connection could be secured by TLS (simply set on the Web server providing the service).

Most authentication mechanisms, including client certificates, rely on HTTP transfer, whereas SOAP is transfer-independent. Therefore, to create a custom authentication mechanism in order to decouple authentication from the transfer protocol, one could pass authentication credentials (for example, X509 digital certificates) in the SOAP-header rather than at the HTTP/SMTP level. In the minimalist-pattern model, fine-grained access control and differential encryption would not be required (at least, at the generic level); receiving applications would need to be aware of the details for each document type. *Whichever mechanism is used, it must obviously become part of the GEM standard.* The security model for the GEM is discussed further in a section below.

Message-Flow in the GEM

In the minimalist-pattern proposed model, *business documents* would, typically, be XML-documents, for example, order-forms and invoices, based on GEM-standardized generic XML-Schema documents (XSD’s). The generic XSD’s would not be comprehensive enough to accommodate all nuances required, but XSD’s standardized at lower levels would extend them. The GEM-standardized *orchestration* (pre-specified messaging sequence) for the exchange of GEM-standardized XML business documents (request messages, such as order-forms; acknowledgement messages; error messages; and response messages) would also be based on the minimalist-pattern, at the generic level. Figure 4.3 illustrates a typical choreography for a general atomic business activity.

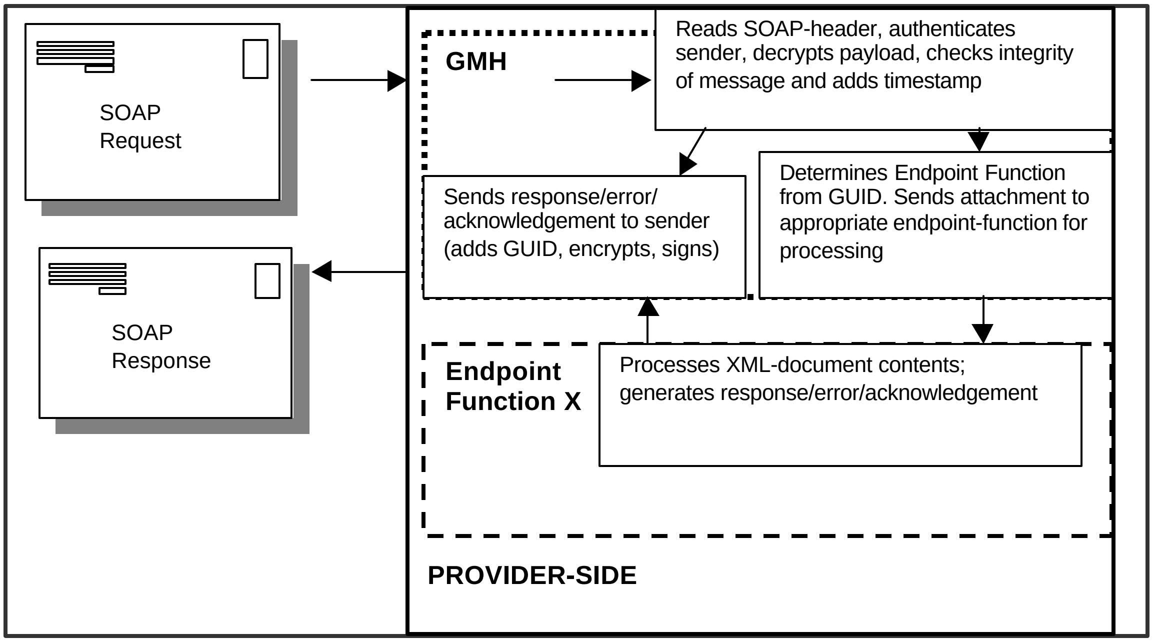

SOAP Request
SOAP Response
GMH
Reads SOAP-header, authenticates sender, decrypts payload, checks integrity of message and adds timestamp
Sends response/error/ acknowledgement to sender (adds GUID, encrypts, signs)
Determines Endpoint Function from GUID. Sends attachment to appropriate endpoint-function for processing
Endpoint Function X
Processes XML-document contents; generates response/error/acknowledgement
PROVIDER-SIDE

Figure 4.3 Provider-end of the GEM Message Pathway

The Global Message Handler (GMH) in the GEM would handle all the incoming and outgoing messages. All communication within the GEM must take place via the GMH. The GMH interfaces with an endpoint-service in order to process/generate SOAP-messages. Although the endpoint-services (Web service methods on the Provider-side) may comprise custom implementations, collaboration with the GMH must comply with the standardized GEM specification (for example, in the use of GUID’s/URI’s). *The GMH could even be used in a non-commercial context – as, for instance, in the education system – by simply installing the list of applicable GUID’s and the corresponding endpoint function libraries.*

The GMH will add security controls and perform packaging-actions on SOAP-messages and send them over pre-specified protocols (HTTP/SMTP) to a receiving

GMH. The GMH at both the Client and the Provider also provides a means for explicit error-handling in addition to the implicit HTTP/SMTP and SOAP standardized error messages.

Minimum Functionality Requirements of the GMH

Extending the basic requirements for the ebXML message service, the following description represents the minimum functionality to be included in a GMH-specification.

WSDL for GMH-Invocation Service

The “bare-bones” service is first represented as a WSDL document, as it is a Web service. In essence, it is a message-relay service; it is both a secure Web service and a secure Web service client. In the WSDL listing below, hypothetical “standardized nomenclature” is used.

```
<?xml version = “1.0” encoding=”UTF-8”?>
<!The type definitions>
<wsdl:definitions name = “GMHInvokeDescription”
targetNamespace=”urn:GMHInvoke”
xmlns:tns=”urn:GMHInvoke”
xmlns:types=”urn:GEMTypes”
xmlns:wsdl=http://www.w3.org/2003/02/wsdl>

<!The Request and Response messages for the GMHInvoke >
<wsdl:message name=” GMHInvokeRequest”>
<part name=”CompanyName” type=”xsd:string”/>
</wsdl:message>
<wsdl:message name=” GMHInvokeResponse>
<part name=”Result” type=”xsd:string”/>
</wsdl:message>

<!The GMH portTypes/Interfaces>
<wsdl:portType name=”GMHInvokeInterface”>
<wsdl:operation name=”GMHInvoke”>
```

```
        <wsdl:input message=”tns:GMHInvokeRequest”/>
        <wsdl:output message=”tns:GMHInvokeReponse”/>
</wsdl:operation>
</wsdl:portType>

<!The service endpoint>
<wsdl:binding name= “GMHInvokeHTTPBinding”
        type=”tns: GMHInvokeInterface”>
 <soap:binding  style=”rpc”
        transport = “http://www.w3.org/2003/12/soap/bindings/HTTP”/>
<wsdl:operation name=”GMHInvoke”>
        <soap:operation soapAction=”urn:GMHInvokeMethod”/>
<wsdl:input>
        <soap:body use=”encoded” namespace=”urn:GMHInvoke”
        encodingStyle=http://.... <!URN for GMH encodingStyle”> />
</wsdl:input>
<wsdl:output>
<soap:body=”encoded” namespace=”urn:GMHInvoke”
        encodingStyle=http://.... <!URN for GMH encodingStyle”> />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>

<!Service location>
<wsdl:service name=”GMHInvokeService”>
<wsdl:port name=”GMHInvokePort” binding = “tns:GMHInvokeHTTPBinding”
<soap:address location=http://ReceivingHost:8080/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

In order to distinguish GEM messages from other messages, different ports could be standardized for HTTP and SMTP (instead of 80 and 25, respectively). As can be inferred from the WSDL definition, the GMH-client is intended to invoke the GMH-service on the Receiver’s end by SOAP-RPC. (Only HTTP-binding is illustrated). No implementation for the service is given. A standardised implementation is proposed in subsequent sections.

Service Client: GMH-Invocation and MTOM-Attachments

Each GMH would have a Sending (Web service Client) module and a Receiving (Web service) module. The Receiving module is described in the WSDL listing above. The Sending module would use the URI supplied by the user to invoke the Receiver’s GMHInvokeService module by SOAP-RPC. If these modules are to be implemented for both HTTP and SMTP, then two modules for Receiving and two modules for Sending will be required. The Sending application for invoking the GMHInvokeService over HTTP would typically send a Request message such as the following:

```
POST /GMHInvokeService HTTP/1.1
Content-Type:text/xml
Content-Length:nnnn
SOAPAction:”urn:GMHInvokeService#GMHInvoke”

<soap:Envelope xmlns:soap='http://www.w3.org/2003/05/soap-envelope' >

 <soap:Body>
<GMHInvoke xmlns=” urn:GMHInvoke”>

  <x:Data xmlns:x='http://GEM.org/data'>

         IK44HhIvWXSX2NIeoJyjiUfI5+ynntOwSmsYyf29ks0NuVSwaHWQedq6kn/qDql6R
         mnu5W2 a44HaiNSnF5B22g==

</x:Data>

 </soap:Body>

</soap:Envelope>
```

Typically, the GMH Web service would be invoked on the Receiver’s end and a standardized XML document (based on the arbitrarily-defined namespace, “http://GEM.org/docs”) would be “attached” using MTOM. In the listing above, the attached XML file occurs within the <x:Data></x:Data> elements. Elements with the namespace name "http://GEM.org/data" and a local name of "Data" will be of a type derived from xs:base64Binary (as defined in that namespace). Such elements will have an xop:Include element child in the MTOM messages and base64 text as children in the case of XML-data. A schema for the Data element used may be represented as follows:

```
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'
           targetNamespace='http://GEM.org/data'
           xmlns:x='http://GEM.org/data' >
  <xs:import namespace='http://www.w3.org/2004/06/xmlmime' />
  <xs:element name='Data' >
   <xs:complexType>
    <xs:simpleContent>
     <xs:extension base='xs:base64Binary' >
      <xs:attribute ref='xmime:contentType' />
     </xs:extension>
    </xs:simpleContent>
   </xs:complexType>
  </xs:element>
  </xs:schema>
```

After XOP, the MIME part of the message will appear as follows:

```
MIME-Version: 1.0
Content-Type: Multipart/Related;boundary=MIME_boundary;
    type="application/xop+xml";
    start="<GMHInvoke.xml@GEM.org>";
    startinfo="application/soap+xml; action=\"ProcessData\""
Content-Description: A SOAP message with an XML part/attachment

--MIME_boundary
Content-Type: application/xop+xml;
    charset=UTF-8;
    type="application/soap+xml; action=\"ProcessData\""
Content-Transfer-Encoding: 8bit
Content-ID: <GMHInvoke.xml@GEM.org>

<soap:Envelope
    xmlns:soap='http://www.w3.org/2003/05/soap-envelope'
    xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>
  <soap:Body>
          <gem:data xmlns='http://GEM.org/docTypes'
          <gem:doc xmlmime:contentType='application/soap+xml'
          <xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
          href='cid:http://Sender.org/enclosedFile.xml'/><gem:doc>
          </gem:data>
     </soap:Body>
```

```
</soap:Envelope>

--MIME_boundary
Content-Type: image/png’application/soap+xml’
Content-Transfer-Encoding: binary
Content-ID: < http://Sender.org/enclosedFile.xml >

// binary octets for xml attachment

--MIME_boundary--
```

If the XML attachment were a SOAP-RPC document, the contentType would be of the format:

```
"application/soap+xml;action=\"http://www.ServiceLocation.net/Method\"".
```

The GMHInvokeMethod (specified in the WSDL listing) would then retrieve the reconstituted XML file and forward it to the appropriate endpoint function – identified by the corresponding GUID or URI – for the specific document type. The GMH will be elaborated further in the next section.

GMH Implementation: Modules in Brief

The following modules would be required to implement the required functionality discussed in the previous subsections.

A *Header Processing module* would required to handle the creation of the SOAP-header elements for the SOAP-envelope, using pre-specified information governing the SOAP-header modules/features to be added, and generated information such as a digital signature, timestamps and GUID’s. This module would also process data in a received SOAP-message's headers, such as authentication data.

The GMH would need to provide comprehensive *Security Services*, including digital signature creation and verification, encryption, authentication and authorization. The Header Processing component would implement generic standardized security services at the external SOAP-envelope level. The encrypted payload (the XML-attachment) would be decrypted, and the digital signature verified, before being passed to the appropriate endpoint-service. The WS-ReliableMessaging feature, placed in the standardized SOAP-header, could be used as a standard means for providing *reliable messaging* (for the delivery and acknowledgment of SOAP

Messages) in the GMH. The service would include persistence, retry, error notification and acknowledgment of messages.

A *Message Packaging component* in the GMH, where the final enveloping of a SOAP Message (HTTP/SMTP elements, SOAP-header elements and payload) occurs, would be required. Here, again, a WS-standard, namely MTOM, could be referenced in the SOAP-header, to provide support for attachments.

An *Error Handling component* would handle the reporting of errors encountered during GMH processing of a message. This would have to handle errors beyond the standard SOAP/XML errors.

An *Endpoint Service Interface* would be an abstract service interface that endpoint-functions would use to interact with the GMH, to send and receive messages (for example, error messages) and which the GMH would use to interface with endpoint-functions handling received attachments. The GMH would be extended to handle non-GEM attachments. In allowing this, the regression to the multiplicity paradox is not necessarily inevitable, provided that GUID's and corresponding XSD's are specified only by a standards-body such as the W3C. As the GEM evolves, so more GUID's will be made available and, hence more standardized atomic activities (including those defining dyadic relationships) will be catered for. Applications and relationships that "stray" too far from the evolving standard will simply lack the interoperability features to participate comprehensively in the GEM.

GMH Operational Framework

Figure 4.4 below illustrates the general envisaged operation of the GMH. In this scenario, a local client first communicates with its local GMH residing on a Web Server. This depicts an asynchronous environment in which Web Servers communicate via HTTP (or SMTP). The messages are queued for sending and also upon arrival. Error messages are relayed for standard SOAP errors (as outgoing SOAP-response messages). The GMH forwards the XML-data from remote SOAP-requests to the appropriate endpoint functions, which in turn produce output that can be packaged by the GMH into SOAP-responses. The GMH also handles incoming

SOAP-responses. It can be implemented in any language, and can also be extended to have custom functionality, such as logging of all activities to a database (for non-repudiation and follow-up reasons, for example).

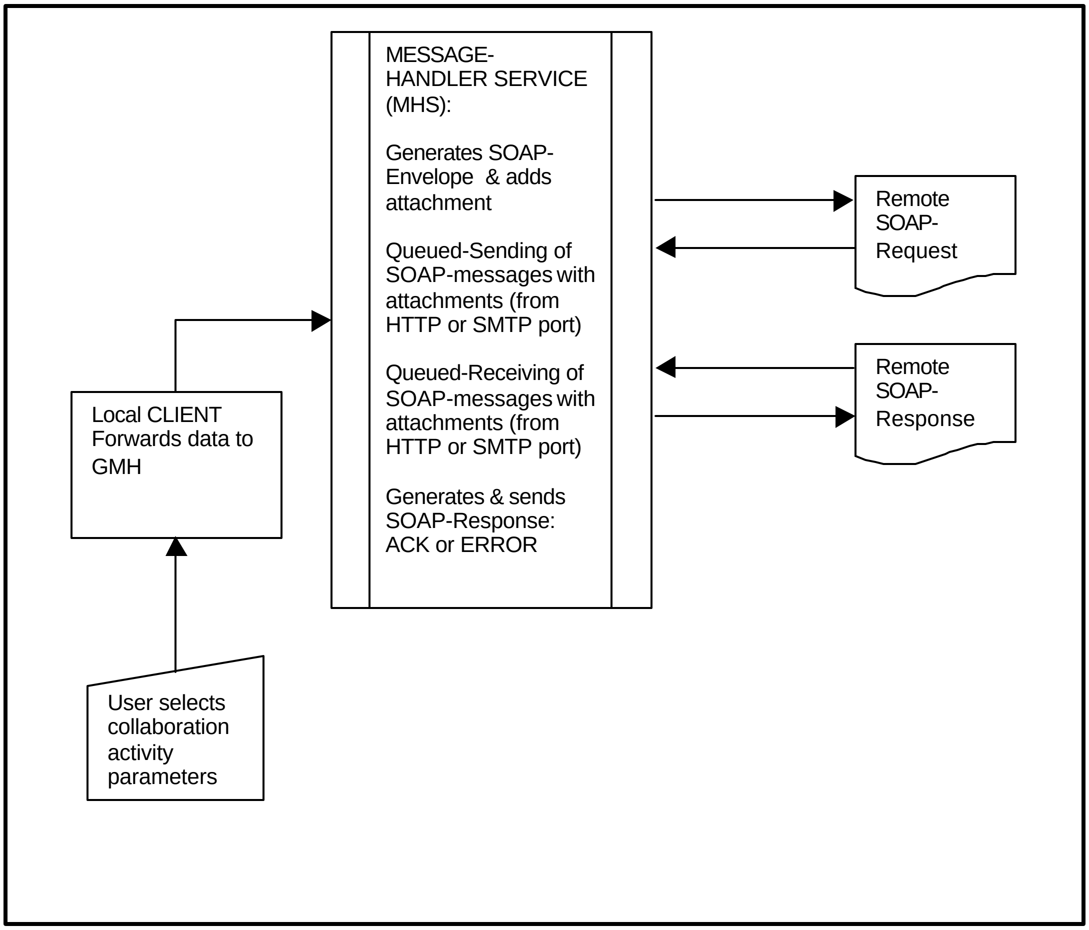


Figure 4.4. High-level scheme of how the Message Service Handler operates

Figure 4.5 below illustrates how a SOAP-Request is sent to a GEM-participant. The objects indicated are based on types available in the .NET Common Language Interface (and include objects from the Web Services Extensions 2 library from Microsoft); the identifiers are fairly intuitive.

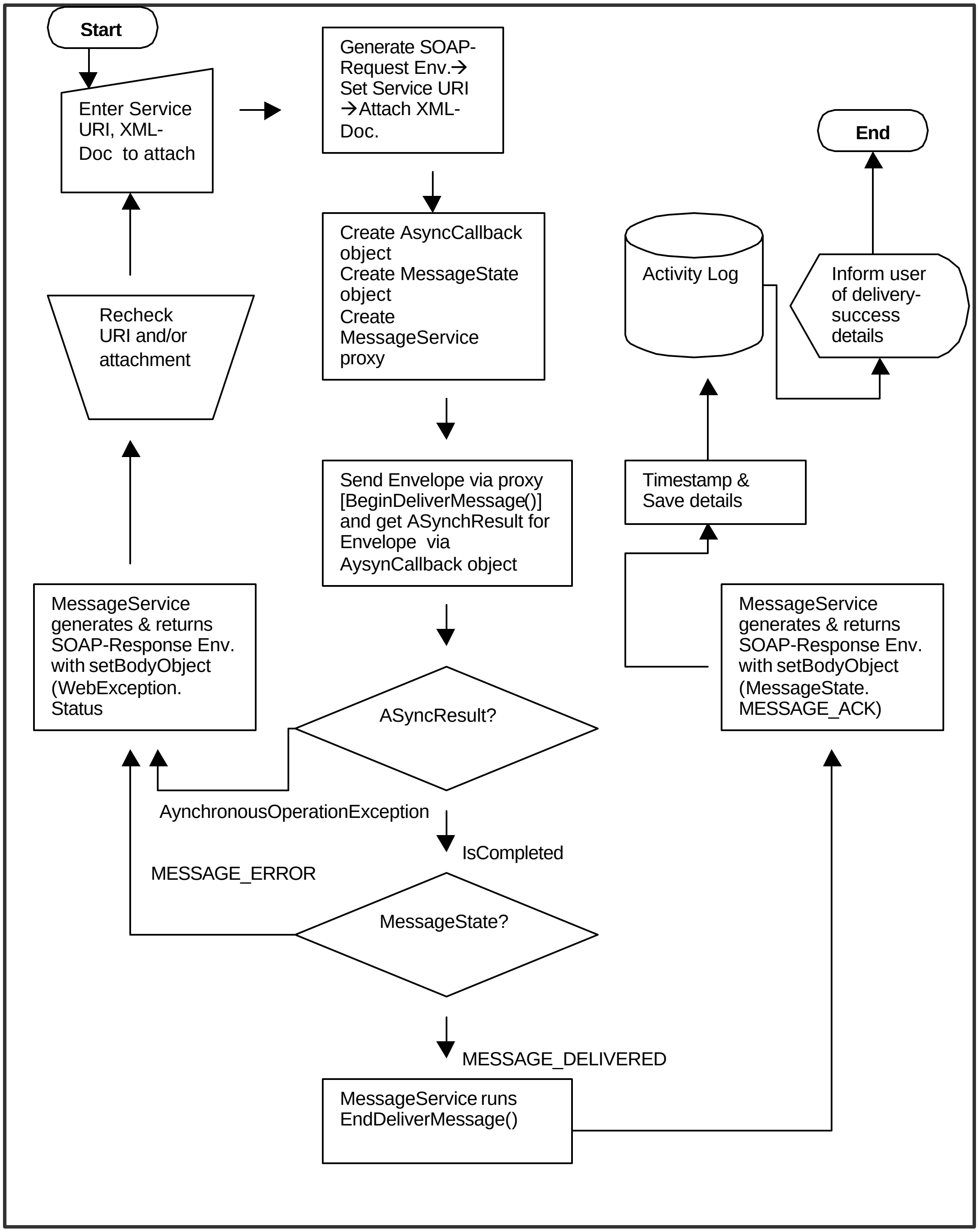

Start
Enter Service URI, XML-Doc to attach
Generate SOAP-Request Env.→ Set Service URI →Attach XML-Doc.
End
Create AsyncCallback object
Create MessageState object
Create MessageService proxy
Activity Log
Inform user of delivery-success details
Recheck URI and/or attachment
Send Envelope via proxy [BeginDeliverMessage()] and get ASynchResult for Envelope via AysynCallback object
Timestamp & Save details
MessageService generates & returns SOAP-Response Env. with setBodyObject (WebException. Status
MessageService generates & returns SOAP-Response Env. with setBodyObject (MessageState. MESSAGE_ACK)
ASyncResult?
AynchronousOperationException
IsCompleted
MESSAGE_ERROR
MessageState?
MESSAGE_DELIVERED
MessageService runs EndDeliverMessage()

Figure 4.5. Example of a Client module for sending an asynchronous SOAP-Envelope with attachment

Similarly, Figure 4.6 below illustrates how SOAP messages are received by the envisaged-GMH.

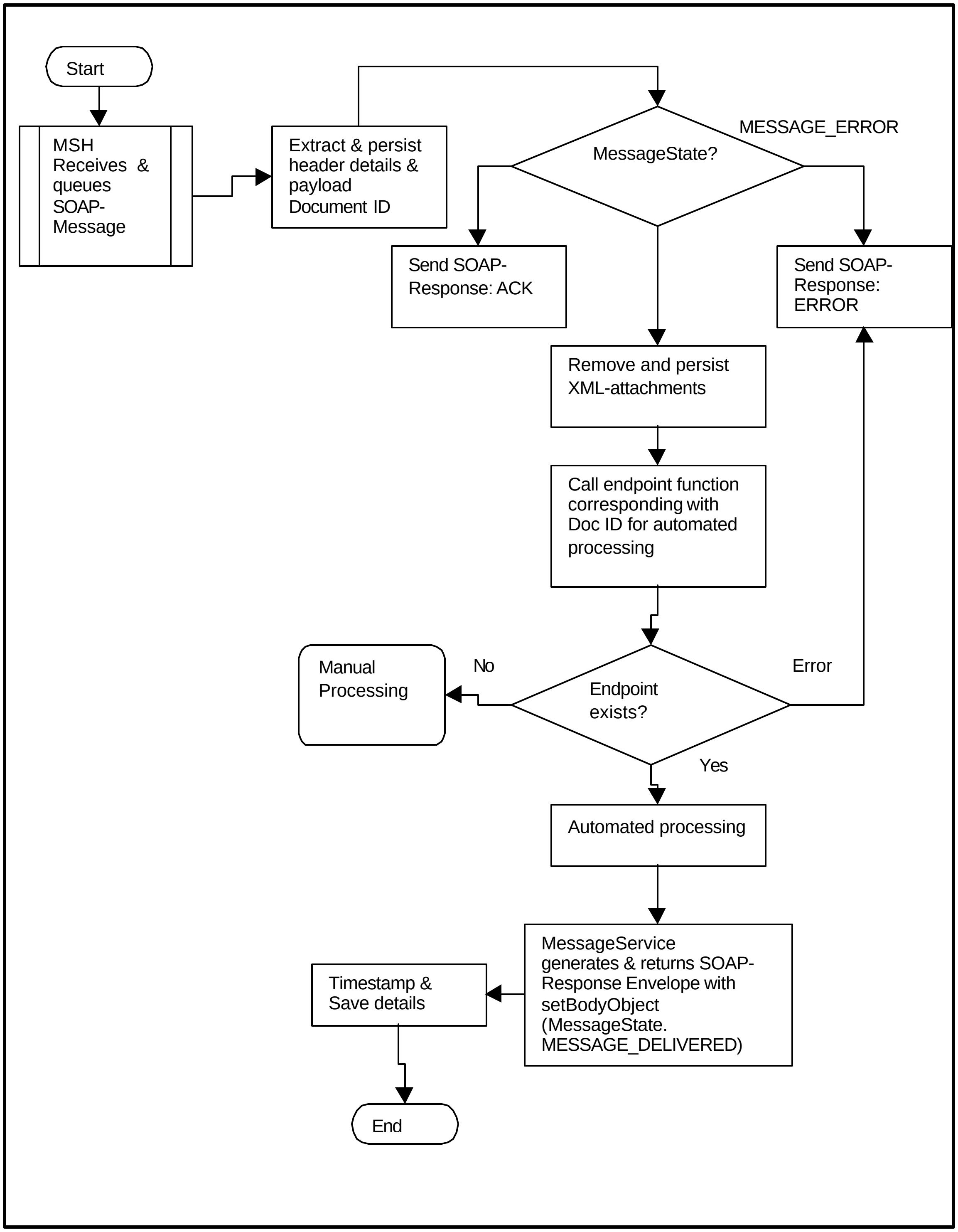

Start
MSH Receives & queues SOAP-Message
Extract & persist header details & payload Document ID
MessageState?
MESSAGE_ERROR
Send SOAP-Response: ACK
Send SOAP-Response: ERROR
Remove and persist XML-attachments
Call endpoint function corresponding with Doc ID for automated processing
Manual Processing
No
Endpoint exists?
Error
Yes
Automated processing
MessageService generates & returns SOAP-Response Envelope with setBodyObject (MessageState. MESSAGE_DELIVERED)
Timestamp & Save details
End

Figure 4.6 Example of receiving an asynchronous SOAP-Envelope with attachment by the GMH

A Security Model for the GEM

The specific security vulnerabilities for general Web services were discussed in chapter 2, as were the various technologies and standards available for securing Web services. What follows, is an emergence of the patterns arrived at in previous chapters. The actual implementation of the standardisation of the SOAP-Envelope is discussed in the next section.

In general, a Web service typically works via an RPC between a SOAP-based Service client and a SOAP-based Service, on the consumer and provider sides, respectively. The calling application makes an HTTP request containing a SOAP-based Service URI. If the SOAP-based Service is for use by a closed user group (CUG) or bears security differentials defined by the identity or role of the principal, then the principal (application or user) needs to first be authenticated and then be granted the allowed permission-set (granted authorization).

This authentication-authorization mechanism can be application-level verification of identity and permissions (code-based) or operating system-level verification of identity and permissions (role-based). The final run-time permission-set of the principal is resolved from the permissions granted to the principal by the application (sand-boxing) and the permissions granted by the environment in which the application is executed.

The identity credentials of the principal can be obtained during the HTTP-Request (using request variables such as the logged-in user identity or the IP-address of the calling principle) or by explicitly demanding authentication from the user (role-based) or from the calling application (evidence-based). If the user is interacting with the provider-service via a Web browser, data can be input as required. However, if the collaboration is to be automated, the consumer-application needs to be able to interact with the provider-application in a pre-defined manner.

In a GEM-context, the security-interoperability between applications would be subject to the multiplicity paradox problem, as described in Chapter 2. The case for a standardised mechanism for (a) separating document payloads from the actual soap envelope (b) securing the documents uniformly (despite additional intra-document security measures) and (c) securing transport of SOAP-messages between applications in a global marketplace context therefore becomes relatively obvious.

GMH applications would service the SOAP-messages moving between consumer- and provider-applications and thus would have to be congruent in respect of protocols used; the protocols for transport, messaging (packaging and encapsulation of data) and security on both sides would have to be interoperable. Again, the multiplicity paradox would stultify the effectiveness of GMH's in a global context, unless a standardized GMH security model was used uniformly throughout.

In addition to the specific Web service vulnerabilities discussed in Chapter 2, the general Client-Server security paradigm – that is, securing the client, the server and the communication line between them – needs to be considered for general specification. As Web servers in a GEM-context would generally be made available in a demilitarized zone (DMZ), general security measures apply. Of relevance to this thesis is that the general measures do not conflict with GEM interoperability. Thus, firewalls and protocol filters must not be set to reject SOAP attachments, for instance.

The security model proposed here recommends that WS-SecureConversation be used for communication between client and server for HTTP connections. This will ensure that a Server-determined encrypted tunnel ensues between client and server. To ensure *identification and authentication* of the sender, a GEM-standardized token (for example, X509 certificates) must be decided upon by the standards body. This will be reflected in the standard WS-Services security elements used in the standardized SOAP-envelope header.

In the minimalist approach, standardized roles could be specified for the set of business processes. Security assertions could be enforced using SAML, which could also be specified in WS-Security. This could then be used for *authorization* (access-control) to specific endpoint-services. Membership of roles is determined by the Provider, with members being added as per Collaboration Partner Agreement. Each Provider is therefore responsible for maintaining its own directory services for pre-negotiated partnering arrangements.

Secrecy and integrity could be implemented using XML-Encryption and XML-Digital Signatures, respectively. A GEM-standardized encryption algorithm – Rijndael (NIST, 2001) is recommended; the WS-Security EncryptedKey element could be used for encrypting a key with the receiver’s public key – and a GEM-standardized encryption mechanism should be uniformly employed.

First, a GEM-standardized hash-algorithm could be used to create a hash of the XML-attachment before XOP packaging. This is to ensure integrity of the attachment. The XML-attachment and the hash could then be encrypted using the sending application’s private key. (This would be used to authenticate the sender). The attachment could then be converted to Binary64 and added to a MIME package with MTOM.

Public keys could be distributed by XKMS (once the “CPA” has been accepted by both sides). Once the XOP:Include element for each attachment has been added, a hash value could then be created of the entire SOAP-body, using the GEM-standardized hash-algorithm. This is to ensure the integrity of the data in the SOAP-body. The SOAP-body (including the encrypted attachment) and the hash value could then be encrypted with the receiver’s public key. (This would ensure confidentiality between the sender and the receiver; only the receiving application would be able to decrypt the body with its private key).

Upon receipt of the SOAP-message, the receiver would have to be authenticated by its security token (as per the SOAP-header specification). Thereafter, it would be able to decrypt the body of the SOAP-message using its private key (as per the SOAP-header specification). It would then create a hash from the entire SOAP-body, using the GEM-standardized hash algorithm. If the two hash values are the same, the integrity of the SOAP-body would be deemed to have been preserved. The XML-attachment would then be decrypted and a hash created, using the GEM-standardized hash-algorithm. If the two hash values are the same, the integrity of the attachment would be deemed to have been preserved. The GUID in the SOAP-body would be used to determine which endpoint-service to pass the XML-attachment to for processing.

Standardized error messages, as per WS-Security and SOAP 1.2, would be generated as appropriate.

The formalisation of specific standards (K variables) to be used in the WS-Security specification can be described using WS-Policy (L laws). By standardising K and L, the “system” becomes highly deterministic.

The implementation of services and the corresponding local security features – beyond the standardised message and standardised messaging infrastructure - are application-specific, but the following guidelines are recommended. To prevent luring attacks, security assertions should be checked at the method-level. To prevent code-injection attacks, all input should be validated comprehensively. Errors should not be allowed to disclose information that might be usable by an attacker; provide appropriate custom errors. Spoofing and masquerading attacks can be eliminated through the use of digital certificates, provided that private keys are kept secret and rigorous authentication is applied.

The GEM SOAP Envelope

The need to de-couple business information (the XML-attachment) from messaging information (the SOAP-envelope) in a GEM serves to mitigate (if not circumvent) the multiplicity-paradox. It also reduces the structural complexity of the GEM. Thus, to reiterate, a standardized GEM-standardized SOAP-envelope would be used, with appropriate SOAP-headers to include features (as defined, but not included, in the SOAP version 1.2 specification) such as "reliability", "security", "correlation", "routing", and "Message Exchange Patterns" (MEPs); see Figure 4.5 below. As two major design goals for SOAP are simplicity and extensibility, the standardized SOAP specification (version 1.2) omits, from the messaging framework, these features; SOAP Version 1.2 provides specifics only for two MEPs. Other features are defined as extensions by other specifications, such as WS-Security (discussed briefly in Chapter 2).

In a GEM, SOAP-headers could be used to pass all out-of-band (not pre-negotiated or related to the semantics of the business process) information. Unlike the *Body* element of a SOAP-message, which generally includes the in and out parameters for the XML Web service method, the *Header* element is optional and can thus be processed by the message-handling infrastructure. However, by providing a standardized (GEM) infrastructure, most out-of-band message-handling requirements (*a la* ebXML CPA's) become redundant. A SOAP-header will provide the transfer protocol binding.

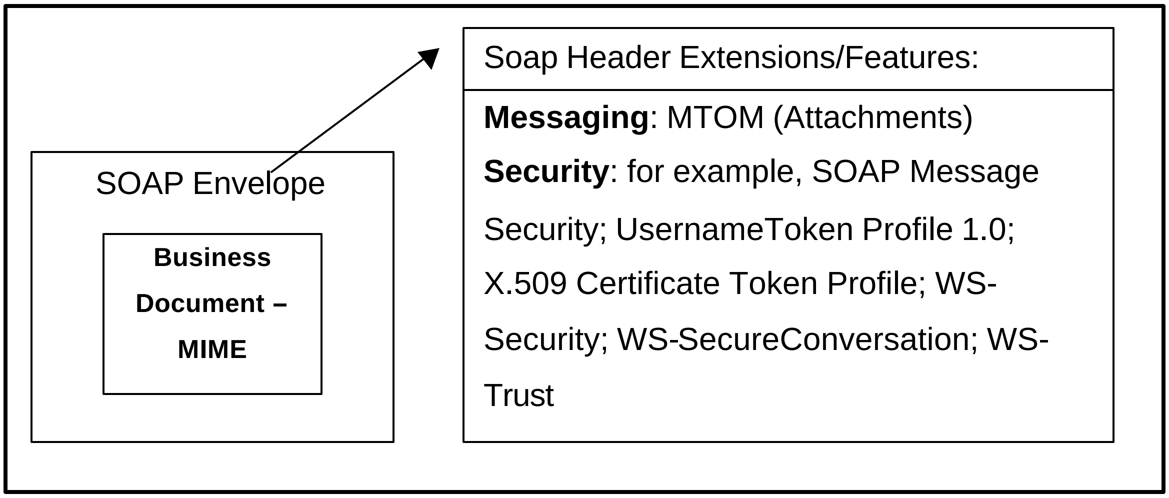

Soap Header Extensions/Features:
Messaging: MTOM (Attachments)
Security: for example, SOAP Message Security; UsernameToken Profile 1.0; X.509 Certificate Token Profile; WS-Security; WS-SecureConversation; WS-Trust
SOAP Envelope
Business Document – MIME

Figure 4.5. The SOAP-envelope header can be extended with standard features

As discussed in Chapter 3, the functionality of the SOAP-Header can be extended with other XML-Infosets, such as those defined for the WS-* (Web Services standard) *features* (Microsoft, 2004). WS-* features, such as WS-Security, WS-Addressing and MTOM (Chatterjee and Webber, 2004:307-344) need to be used to define standard values for security protocols, addressing parameters, and messaging parameters. MTOM allows for files (including XML-documents) to be carried as attachments in the Body-section of a SOAP-Envelope.

A sample of the hypothetical GEM-standardized SOAP-Envelope is presented below:

```
<?xml version=”1.0” encoding = “UTF-8”?>
<S:Envelope xmlns:S=’http://www.w3.org/2001/12/soap-envelope’
        xmlns:ds=’http://www.w3.org/2000/09/xmldsig#’
        xmlns:wsse=’http://schemas.xmlsoap.org/ws/2002/04/secext”
        xmlns:enc=http://www.w3.org/2001/04/xmlenc#
```

The references in the SOAP envelope are for referencing: the standard SOAP-Envelope namespace, the XML Digital Signature namespace, the WS-Security namespace and the XML Encryption namespace; all of these are W3C standards.
Next, standard features will be added to allow the creation of a standardized Envelope header.

```
<S:Header>
        <m:path xmlns:m=http://schemas.xmlsoap.org/rp/>
        ……..
        </m:path>
```

This is a reference to the WS-Addressing namespace. This is generally used when there is more than one Receiver. For the GMH, it is possible that the same message could be sent to various Receivers simultaneously. No standardisation is envisaged for this feature, at this point.

Next, comes the WS-Security header element.

```
        <wsse:Security wsse:actor = “urn: receiverGMH”>
        <wsse:BinarySecurityToken Id=”CertToken”
                ValueType=”wsse:X509v3”
                EncodingType=”wsse:Base64Binary”>
                ……..
        <wsse:BinarySecurityToken>
```

For each receiver (wsse:actor), a set of Security elements is required. Authentication could be standardized on a UserNameToken element (with basic username, password and message sub-elements), or a BinarySecurityToken element such as the one above. Various tokens are permissible, including Kerberos tickets and X.509 certificates. The string-value Id could be standardized in a GEM, since the token-type must be standardized (within the proposed framework). The values for ValueType and EncodingType can be standardized from available pre-defined types, e.g. X509v3 could be used as the standard.

If, as proposed earlier, the entire body of the external SOAP-envelope is ultimately encrypted, the details of the encryption must appear in the unencrypted header. This includes the encryption method (the algorithm used), the key, the resulting cipher-value and the location of the original document-part that was encrypted (the DataReference element).

```
<xenc:EncryptedKey>
        <xenc:EncryptionMethod Algorithm =”…”/>
        <ds:KeyInfo>
                <ds:KeyName> …</ds:KeyName>
        <ds:KeyInfo>
        <xenc:CipherData>
                <xenc:CipherValue> … </xenc:CipherValue>
        </xenc:CipherData>
        <xenc:ReferenceList>
                <xenc:DataReference URI=”#encryptedBodypart”/>
        </xenc:ReferenceList>
</xenc:EncryptedKey>
```

The encryption algorithm could be standardized (Rijndael is suggested in the previous section) as can the DataReference URI, when the entire body (except the Body elements) is encrypted.

The elements for the digital signatures used come next. In the case of the GMH, the attachment (before XOP) and the entire SOAP-body will have References, as indicated in the previous section.

```
<ds:Signature>
        <ds:sSignedInfo>
                <ds:CanonicalizationMethod
                Algorithm=’http://www.w3.org/TR/2001/REC-xml-c14n-20010315’>
                <ds:SignatureMethod
                Algorithm= ‘http://www.w3.org/2000/09/xmldsig#dsa-sha1’/>
                <ds:Reference>
                <ds:Transforms>
                        <ds:Transform Algorithm=”…”/>
                </ds:Transforms><
                </ds:Reference>
        </ds:SignedInfo>
                <ds:SignatureValue>….</ds:SignatureValue>
        <ds:KeyInfo>
            <wsse:SecurityTokenReference>
                <wsse:Reference URI=’#CertToken”/>
            <wsse:SecurityTokenReference>
        </ds:KeyInfo>
</ds:Signature>
</wsse:Security>
</S:Header>
```

The listings above are merely to indicate the manner in which the SOAP-Envelope could be extended to include (WS-*) features, which enable standardisation of the Envelope. The actual features and corresponding values for this standardisation (in a GEM) can only be specified by the standards-body assuming responsibility for this. Comprehensive proof-of-concept testing in this regard is currently subject to the paucity of support for the newer WS-* standards in existing development environments.

The Business Information Processing Infrastructure

Similar to the ebXML Registry/Repository, the GEM Registry/Repository would provide UDDI services for GEM-standardized generic specifications, XSD’s, document templates, and core components (object libraries). Users would be required to register for services. The GEM Registry/Repository, the contents and the corresponding services, would constitute the *business information processing infrastructure of the proposed framework*.

The Registry/Repository would need to be secured from denial-of-service and integrity attacks. Only members authorized by the standards body (W3C) would be able to add or edit the contents of the GEM. As with all open standards, copyright would exist, but free dissemination of documents would be allowed.

In practice, a hierarchy of Registry servers would probably exist, allowing for systematic “inheritance” of generic standardized documents, generic business processes, generic common components, and so forth. The hierarchy would mimic the refined requirements – from the more generic to the more specific – in business patterns (and related documents, businesses processes and so forth), from cross-industry (m:n) collaboration to monopolistic dyadic collaboration. The authority for managing each level of the hierarchy would logically devolve to major standards bodies at each level, provided that original standardisation is maintained (similar to PKI); the most granular-defined business activities/documents should still be interoperable with the most generally-defined ones. In subsequent paragraphs, user-groups occupying a particular level are referred to as a "*user-community*".

The *logical message-handling procedure* is illustrated in Figure 4.6 below. In the illustration, data for processing is sent to the receiving application by way of a global standardized message format (the GEM Request Envelope of SOAP-RPC type). The message-handler reads the *logical* "address" (URI for Receiving GMH) on the envelope of the message. Interoperable security for the message is provided in the standardized header of the GEM Envelope. The data being transferred is carried as an XML-binary Optimised Package (with message-specific security, where applicable). The (XOP) document-type has a corresponding globally-unique identifier (or URI, for the endpoint-function ”address”) in a standard list for that user-community; this is the physical location of the actual data-processing function that handles that specific standardized document-type (and the data within). The data-processing function-library for that specific user-community would be commonly-specified within that community, based on signatures/programming interfaces inherited from the generic upper layers of the business pattern standardization hierarchy. Nonetheless, the function-signatures (the function identifier, the function parameters and corresponding data-types) must be pre-specified; inputs and outputs to the function must be standardized for uniform automated processing of data and the *set of functions* for a particular user-community must be standardised. Thus, what ultimately distinguishes one user-community from another, irrespective of which industry group/sub-group/sub-sub-group/etc they belong to, is the set of endpoint-functions. The message-handling infrastructure (GMH and its modus operandi) and the message-format (the XOP attachments and the standardized WS-feature SOAP-header) and the backward compatibility (by way of generic elements, nomenclature and choreography) would have sufficient commonness to allow electronic integration across user-communities.

Following the processing of the data, the message-handler packages the function-output in a standardized format and returns it to the sender's message-handler, where complementary (consumer) processing will occur. The proposed framework extends the common nomenclature to include programming interface- and method-signatures; the idea of a common typology (a standardized common-object hierarchy without actual implementations) being specified in XML (with pre-specified element and attribute parameters) which can be manipulated via equivalent variables in company-specific implementations, is considered an ideal goal.

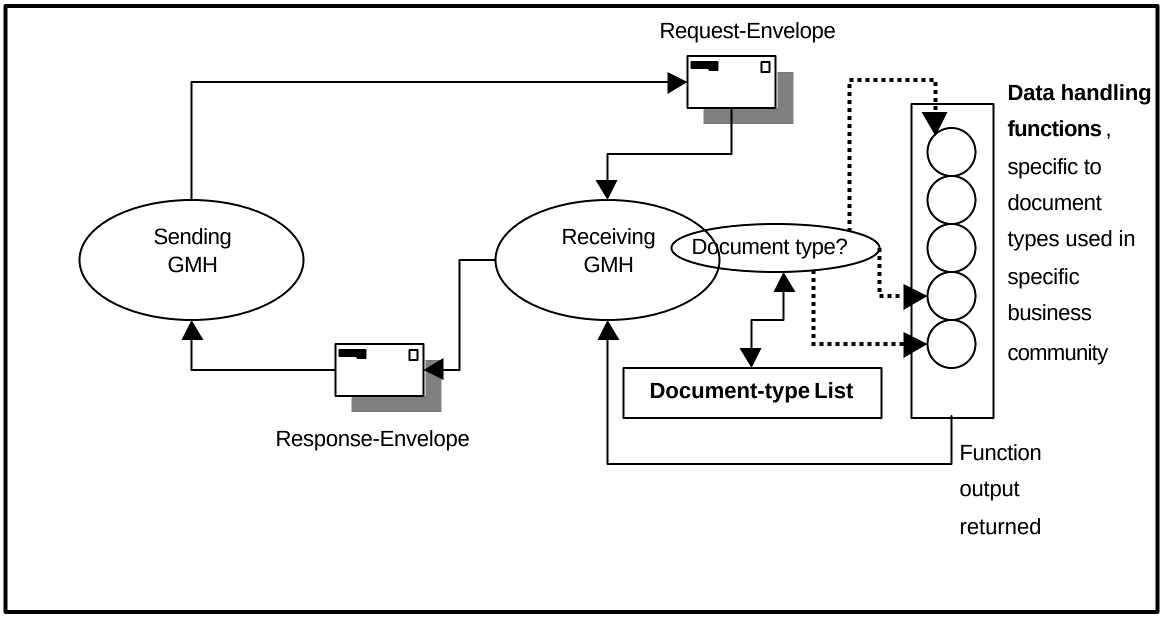


Figure 4.6. The common message-handler interface operates within a pre-defined collaboration-context

It is not difficult to conceive of documents inheriting from a generic standard (in a hierarchical manner) being handled in a recursive fashion by endpoint-functions (in the processing of XML data). If user-community-specific elements and attributes are not found, the processor merely searches for the next equivalent (generic) element or attribute in the hierarchy.

Conclusion

In this chapter, the "anatomical-physiological" basis of the proposed interoperability framework for a GEM was described. The underpinning philosophy of the anatomical structure (in the emergence/gestalt) is the (deterministic) specification of a common set of variables, patterns and "laws" for the sets K and L depicting and constraining the limits of the global-integration "system". The framework proposed in this chapter collates variables, patterns and "laws" within a Web services-based Services-Oriented Architecture, which promotes interoperability in a global context.

In Chapters 1, 2 and 3, the need for structural determinism was emphasized. The nature and justification for a GEM were explored and the requirements (critical success factors) for a workable integration solution were sought (Chapters 3 and 4).

The basis for a global "system" comprising a standardised business information processing infrastructure (a hierarchy of specifications, based on a generic minimalist business pattern), a standardised message-and-messaging infrastructure (a standardised SOAP-header, standardised message handler, and a standardised message-carrying format) was arrived at by examining existing models, technologies, standards and development paradigms, in relation to the "system" requirements and critical success factors.

Development-support for the implementation of the proposed framework is growing steadily and current prototypes will benefit a great deal from SOAP-Processor support for XOP and MTOM, for instance. While SMTP already supports MIME-packaging, as used in XOP, SOAP-over-SMTP support is still in its infancy.

The following chapter analyses the validity of the framework in terms of the requirements for a GEM, delineated in earlier chapters, and in terms of critical success factors for the envisaged GEM.

Chapter 6

Evaluation of the Framework and Conclusions

Introduction

In this chapter the validity of the proposed framework using a critical success factor (CSF) analysis is explored. The CSF's are based on common literature critique of existing models as well as observed patterns and requirements depicting the pros and cons of existing systems (as discussed in previous chapters).

A Critical Success Factor Analysis of the GEM Framework

Rawlings (2001) identified the following critical aspects for ebXML (as discussed in Chapter 4), which are largely applicable to the GEM Framework proposed in this thesis:

Common business process: The framework addresses this aspect by advocating a universally-applicable GEM standard specifying a standardised generic minimalist-business-pattern, standardized process flows, standardized exchange choreography, and standardized business documents. Business processes of a more context-specific nature will inherit from the more generic business processes higher up the hierarchy. Unlike with ebXML, participants will not be allowed to create their own business processes – although they will be able to customise the operation of endpoint services – as this would have the potential to escalate the various permutations of variables, as discussed in Chapter 3. It is suggested that the standardisation be undertaken and maintained by the W3C.

Common nomenclature: The entire framework is founded in a published standardized nomenclature (taxonomy and naming system) based on a standardized vocabulary and standardized semantics. In concrete terms, this is manifested in a set of specifications – including a glossary of terms – in which all items listed in the "Common business process" above, are specified. All standardized business documents would be specified as XML Schema Documents (XSD's) to ensure that common meta-data persists throughout, in keeping with the standardized nomenclature. Just as WS-* namespaces (specifications) are equivalent to referenced programming-classes, so too XSD's for standardised documents corresponding with standardised business programming-classes are proposed as a referenced typology of generic business programming interfaces and method-signatures; XSD's will reflect the standardised parameters to methods and constructors. Although the actual level of standardisation would be a colossal undertaking – EDI was similar in this regard – this would provide the unequivocality required by this CSF.

Common character encoding: As the framework is grounded in XML, the W3C standard for XML, UNICODE, is the only logical choice at present.

Common data transfer protocol: The use of SOAP in the context of a GEM is almost inevitable at this point in time. As it is based on XML, SOAP is the veritable "universal translator", being accessible to all software processing platforms. The Remote Procedure Call functionality of SOAP makes it possible to invoke remote methods on disparate platforms; it is therefore ideal for invoking the Global Message Handler proposed in the framework. While communication parameters are passed in SOAP and the underlying HTTP/SMTP headers, the actual business data for processing is contained in an XML-attachment. The "wire" data transfer protocol is suggested as HTTP and SMTP. The global message handler should be designed to allow users to toggle between listening/polling for messages on HTTP (for synchronous processing) or SMTP (for asynchronous processing) or both.

The use of XOP with MIME inclusions and a standardised SOAP envelope is an attempt to further abstract differences into the common interface. Everyone uses the same sending and receiving mechanisms – the same pre-specified features for addressing, security, and so forth, as well as the same message-carrying mode – right up to the supply point (standardised endpoint interface). The messages of a particular type are structured the same, but need not be processed in the same way (implementation of endpoints is organisation-specific); however, the standardised data types are returned in standardised responses.

Common network protocol: The standard Internet transport/network protocol-combination TCP/IP should be specified as a GEM standard.

Common security protocol: A model for standardizing security in the GEM architecture is proposed. It is suggested in the framework that the multiplicity paradox, discussed in Chapter 3, be avoided by abstracting configuration items into a single model. For instance, the standardized symmetric encryption protocol should be specified as Rijndael (the Advanced Encryption Standard, which is also the Federal Information Processing standard in the USA). A standardized mechanism for security the message and the messaging (using XML-Encryption, XML-Digital Signatures and security tokens) for privacy, authentication, authorization, integrity and non-repudiation, is proposed. The framework appears to address this CSF adequately, in principle.

Easy access for Small-to-Medium Enterprises: The proposed framework is intended to commence with using, inter alia, a generic minimalist-business-pattern, with freely-available specifications (including component libraries) and template documents. Free distribution of Web service implementations of the global message handler for the common platforms would no doubt accelerate access by SME's.

By inference, from existing models and theorising, further critical success factors were alluded to throughout the progression towards the GEM *gestalt*. These are presented in italics below.

Two critical success factors of paramount significance in the GEM context are: A *standardized common interoperability-interface* and a *predictable structure*.

The human body is a complex system, from an anatomical-physiological perspective, comprising numerous other systems (for example, the circulatory system, the digestive system, the neuro-endocrine system, and so forth). However, the "state variables" of each system are closely related, being specified by the same standard blue-print (the human genome; equivalent to the governing laws in set L, which specify norm values for variables such as body temperature, blood-glucose levels, and so forth) and being grounded in the same basic protocols (for example, for metabolic biochemical processes, homeostatic mechanisms, and so forth). Thus, although diverse, the structural distance between patterns in each “entity” (system) is closer to one than it is to zero; implying a tendency towards orderliness/structure rather than towards chaos/disorderliness (as exists in certain pathological states). Similarly, the GEM would be part of a much larger “system”, the Internet. A simplistic analogous deduction would be that a common (W3C-specified) blue-print would promote orderliness on the Internet; thus, all subsystems should gravitate towards using the same standardized specifications (deterministic structure).

In the proposed GEM-framework, it was deduced that in order to promote a predictable structure, state variables between entities (disparate participating systems) would be reconciled by specifying a new standardized interoperability-interface that reduces or eliminates the significance of existing state variables. Conceptually, the interface comprises the following:

- An interoperable standardized *message format:* a standardized SOAP-envelope with attached standardized XML-documents.
- This is delivered by an interoperable standardized technical infrastructure: a *message-delivery infrastructure* encompassing an *application-level message-handler* - a Web service which can operate in synchronous and/or asynchronous mode, which uses standardized endpoint nomenclature and GUID’s – *and a business information processing infrastructure* which accommodates custom implementations, based on standardized programming interfaces and virtual methods.
- The message-delivery infrastructure is scalable and, in turn, interfaces with a standardised business infrastructure: a *generic*, *standardized minimalist-business-pattern* comprising a *global pre-specification for business processes standardized specific atomic business activities*, a *standardized document-set* based on *standardized metadata* (common nomenclature and common semantics used in XSD’s), the *standardized corresponding choreography for atomic activities* and a standard *typology of business programming-objects* (interfaces and class signatures that correspond with XSD parameters).

Unfortunately, as argued (in Chapter 3), an opposing momentum has been created by a multiplicity-paradox phenomenon. In seeking to provide dyadic integration, existing B2B-collaboration models have promoted discrete pockets of collaboration configurations, which are dotted across the B2B space. Therefore, the proposed framework is an attempt to also provide for *co-existence with current dyadic relationships* and allow for *existing systems to migrate towards the new standard in an incremental fashion*.

Web services are becoming ubiquitous across all platforms and, using WS-* specifications to extend and standardize the functionality of SOAP, is fast becoming the norm. To specify a set of standards for a GEM, based on existing standards, is what is being proposed. The migration towards an ebXML-like environment (described in Chapter 3) is likely to continue; a GEM standard – without the multiplicity-paradox conditions – should enhance this migration. As discussed in Chapter 5, the framework accommodates non-GEM collaboration, since endpoint applications may be extended beyond the GEM specifications and non-standard files may be delivered by the GMH, provided that the functioning of the GMH within the GEM is not altered. An *ebXML-like, UDDI Registry/Repository mechanism* for the discovery of TP's and for downloading GEM specifications, is also recommended as part of the framework.

The GMH is based on open standards, notably TCP/IP, HTTP, SMTP, XML, SOAP, WS-*, XML-Encryption and XML-Digital Signatures; it is therefore accessible to everyone. It allows for flexible and adaptable processing of the (standardized) data contained in the XML attachments, and even accommodates customer-specific processing (using non-standardized documents and business processes). It uses native XML as much as possible and allows the integration of existing middleware.

Possible Future Developments

It is quite conceivable that Application Service Providers and Grid Service Providers would participate in the GEM as outsourcers of B2B-integration endpoint services, very much in the way that telecommunication services and cellular telephone networks outsource such services. Clearly, a GEM standard would greatly facilitate this. A predictable GEM-structure is of vital importance for this to be realized.

Conclusions

The primary contributions to the area of electronic business integration, propounded by this thesis, are (in no particular order):

- *A novel examination of global Business-to-Business (B2B) interoperability in terms of a "multiplicity paradox" and of a "global electronic market-space" from a Complex Systems Science perspective.*
- *A framework for an, integrated, global electronic market-space*, which is based on a hierarchical, incremental, minimalist-business-pattern approach. A Web Services-SOA forms the basis of application-to-application integration within the framework. The framework is founded in a comprehensive study of existing technologies, standards and models for secure interoperability and the SOA paradigm. The Complex Systems Science concepts of "predictable structure" and "structural complexity" are used consistently throughout the progressive formulation of the framework.
- *A model for a global message handler* (including a standards-based message-format) which obviates the common problems implicit in standard SOAP-RPC. It is formulated around the "standardized, common, abstract application interface" critical success factor, deduced from examining existing models. The model can be used in any collaboration context.
- *An open standards-based security model* for the global message handler.

Conceptually, the framework comprises the following:

- An interoperable standardized message format: a *standardized SOAP-envelope* with *standardized attachments* (8-bit binary MIME-serialized XOP packages).
- An interoperable standardized message-delivery infrastructure encompassing an *RPC-invoked message-handler* - a Web service, operating in synchronous and/or asynchronous mode, which relays attachments to service endpoints.
- A business information processing infrastructure comprised of: a *standardized generic minimalist-business-pattern* (simple buying/selling), comprising

global pre-specifications for business processes (for example, placing an order), *standardized specific atomic business activities* (e.g. completing an order-form), a *standardized document-set* (including, e.g. an order-form) based on *standardized metadata* (common nomenclature and common semantics used in XSD’s, e.g. the order-form), the *standardized corresponding choreography for atomic activities* (e.g. acknowledgement of receipt of order-form) and service endpoints (based on *standardized programming interfaces and virtual methods* with customized implementations).

It is, humbly, concluded that the proposed framework represents a feasible gestalt upon which to build a global electronic market-space. It is acknowledged, however, that implementations, based on the framework, in smaller user-communities are more conceivable.

References

- Agarwal, A., Shankar, R. (2002). Analyzing alternatives for performance in supply chain performance; Work Study Volume 51 Number 1 2002 pp. 32-37; MCB University Press ISSN 0043-8022
- Ahmed, Z., Balabine, I., Berwanger, R., Boseman, A., Brown, A., Bussey, A., et al. (2001). ebXML Technical Architecture Risk Assessment v 1.0. Retrieved March 8, 2003, from http://www.ebxml.org/specs/secRISK.pdf
- Apshankar K. (2002, July 24). WS-Security: Security for Web Services. Retrieved February 18, 2003 from http://www.webservicesarchitect.com/content/articles/apshankar04.asp
- Arciniegas FA. (2001). XML Developer's Guide. McGraw-Hill.
- Benatallah B, Casati F (eds). (2002). Special Issue on Web Services. *Distributed Parallel Databases* 12 (2).
- Bouguettaya A, Benatallah B, Elmagarmid AK .(1998). Interconnecting heterogeneous information systems. Kluwer, Boston, Mass., USA
- British Standards Institute (1999), BS 7799: Code of Practice for Information Security.
- Brodie M . (2000). The B2B E-commerce revolution: convergence, chaos, and holistic computing. In: Brinkkemper S, Lindencrona E, SolvbergA (eds) Information system engineering: state of the art and research themes. Springer, London, UK
- Brown, T, M. (2002). The Final Word: Software in 2002. EAI Journal, February 2002, p. 48. Retrieved July 30, 2003, from http://www.goendymion.com/expertise/resources/articles/software2002.pdf
- Budd, F & Gray, D (2002). Security – A Focus on IPSec. Retrieved August, 19, 2003 from http://www.wiltelcommunications.com/vpn
- Canavan JE (2001). *Fundamentals of Network Security*. Artech House.
- Chatterjee S and Webber J. (2004). Developing Enterprise Web Services - An Architect's Guide. Prentice-Hall PTR.

- Chen M. (1995). A Model-Driven Approach to Accessing Managerial Information: The Development of a Repository-Based Executive Information System. *Journal of Management Information Systems 11(4): 33-64.*
- Choudury V. (1997). Strategic Choices in the Development of Inter-organizational Information Systems. *Information Systems Research 8(1): 1-24.*
- Cisco (2003). IPSec Network Security. Retrieved September 16, 2003 from http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113t/113t_3/ipsec.htm
- Czajkowski K., Ferguson, D., Foster, I., Frey, J., Graham, S., Maguire, T., Snelling D., Tuecke, S. (2004). From Open Grid Services Infrastructure to WSResourceFramework: Refactoring & Evolution. Version 1.1 Retrieved October 30 from Globus.org website (ogsi_to_wsrf_1.0.pdf).
- D. Trastour et al. (2003). Computer Networks 42: 661–673.
- Daum B. and Scheller M. (2000). Success with Electronic Business. Addison-Wesley.
- Dogac A (ed) (1999) Special Issue on Electronic Commerce. *Distributed Parallel Databases* 7(2)
- Drummond R (2001). Put SOAP and ebXML to Work. *E-Business Advisor Magazine, June/July 2001.* Retrieved July 16, 2003 from http://advisor.com
- ebMS (2002). Message Service Specification Version 2.0 revision C. Retrieved March 12, 2003 from http://www.ebxml.org
- ebRIM. (2001). ebXML Registry Information Model v1.0. Retrieved 23 April from the World Wide Web: http://www.ebxml.org/specs/ebRIM.pdf.
- ebRS. (2001). ebXML Registry Services Specification v1.0. Retrieved 23 April from the World Wide Web: http://www.ebxml.org/specs/ebRS.pdf.

- ebXML_TA. (2001). ebXML Technical Architecture Specification v1.0.2. Retrieved 26 April from the World Wide Web: http://www.ebxml.org/specs/ebXMLTA.pdf.
- Enabling Electronic Business with ebXML.– *White Paper.* (2001). Retrieved March 14, 2003 from http://www.ebxml.org/specs/index.htm#white_papers
- Erl T. (2004). BizTalk Best Practices. Retrieved 31October 2004 from http://www.XMLTC.com/BizTalkExpert.
- Fastwater (2003). Interoperability. Retrieved 15 April 2003 from www.fastwater.com
- Franks, J. (2000). "*Supply chain innovation*", Work Study, 49, 4, 152-5.
- Geotrust (2003). What is a Dun and Bradstreet Number (DUNS)? Retrieved September 8, 2003 from http://www.geotrust.com/true_businessid/order/duns_number.htm
- Goertzel B (1994). Chaotic Logic – Language, Thought, and Reality from the Perspective of Complex Systems Science. Plenum Press.
- Gottschalk, K, et al. (2002). Introduction to web services architecture. IBM Systems Journal 41.2 :170-177.
- Grangard, A., Eisenberg, B., Nickull, D., Barham, C., Boseman, A., Barret, C., et al. (2001). ebXML Technical Architecture Specification v 1.0.4. Retrieved March 8, 2003, from http://www.ebxml.org/specs/ebTA.pdf.
- Greenstein M, Feinman TM. (2000). Electronic Commerce: Security, Risk Management and Control. McGraw-Hill.
- Hasselbrig, W. (2000). “The role of standards for interoperating information systems”, Jakobs, K, Information Technology standards and Standardization: A Global Perspective, Idea Group Publishing, Hershey, PA, 116-30.
- Hasselbring, W., & Weigand, H. (2001). Languages for electronic business communication: state of the art. *Industrial Management & Data Systems*, 101, 5, 217-227.
- Ibbotson, J. (2002). XML Technology and Messaging Course notes.

- Institute of Electrical and Electronics Engineers (1990). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* New York, NY. Retrieved July 11, 2003, from http://www.sei.cmu.edu/str/indexes/glossary/interoperability.html
- IONA (2001). IONA Completes End-to-End Business Scenario Based on ebXML. Retrieved March, 12, 2003, from http://www.iona.com
- Kay R. (2003), XACML, Retrieved August 12, 2003 from http://www.computerworld.com/securitytopics/security/story/0,10801,81295,00.html, 19 May, 2003)
- Kiely, D (2001). E-business Language of Choice? *Information Week, 836*, 79.
- Kumar K. and Dissel H.G. (1996). Sustainable Collaboration: Managing Conflict and Cooperation in Inter-organizational Systems. MIS Quarterly September: 279-299
- Leymann, F. & Roller, D. (2003). Business Processes in a Web Services World. Retrieved from the World Wide Web 2003 : http://www-106.ibm.com/developerworks/webservices/library/ws-bpelwp
- Lichtenstein, S., & Swatman, P.M.C (2000). Issues in E-Business Security Management & Policy. Presented at the 1st Australian Information Security Management Workshop. Retrieved July 6, 2003, from http://www.cm.deakin.edu.au/mwarren/secman/pdf/3.pdf
- Loeb, L (2002). Donald Eastlake on XML digital signatures. Retrieved September 7, 2003 from http://www-106.ibm.com/developerworks/xml/library/s-east.html?dwzone=xml
- Ma, M, 1999, "Agents in E-commerce", Communications of the ACM, 42, 3, 79-80.
- Mak, K., & Ramaprasad, A. (2001). An Interpretation of the Changing IS/IT Standard Game, Circa 2001. *Knowledge, Technology & Policy, 14(2),* 20-30. Retrieved June 5, 2003, from Academic Search Premier database.
- Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A.H.H., Elmagarmid, A.K. (2003). Business-to-business collaborations: issues

and enabling technologies. *The VLDB Journal, 12,* 59+. Retrieved March 15, 2003, from ACM Digital Library.

- Microsoft Corporation. (2001). XML Web Services Overview. Retrieved 23 April 2003 from the world wide web: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconwebservicesoverview.asp
- Microsoft Inc. (2003, May 15). What are Web Services? Retrieved December 8, 2003, from http://www.microsoft.com/net/basics/webservices.asp
- Microsoft Inc. (2004, MSDN Library). Messaging Specifications. Retrieved December 8, 2004, from http://www.microsoft.com/webservices/ understanding/specs/default.asp
- Nagel, E (1974). The Structure of Science – Problems in the Logic of Scientific Explanation. Routledge and Kegan Paul.
- Naujok, K & Berwanger, R (2001). Electronic data interchange – Standards; Organization for the Advancement of Structural Information Standards; XML (Document markup language). *Interactive Week, Vol. 8 Issue 21*, p40.
- NIST (2001). AES Home Page. Available [online] at http://www.nist.gov/aes/. Accessed: 31/08/01
- Nurmilaakso, J., Kettunen, J., Seilonen, I. (2002). XML-based supply chain integration: a case study, Integrated Manufacturing Systems, volume 13 Number 3 pp 586-595.
- O' Neill et al. (2003). Web Services Security. McGraw-Hill/Osborne.
- OASIS. (2000). ebXML Whitepapers: *Enabling Electronic Business with ebXML*. Retrieved September 3, 2003, from http://www.ebxml.org/white_papers/whitepaper.html
- Oppliger R. (2000). Security Technologies for the World Wide Web. Artech House.
- Pabray UO and Gurbani VK. (1996). Internet & TCP/IP Network Security: Securing Protocols and Applications. McGraw Hill.

- Phaltankar KM. (2000). Practical Guide for Implementing Secure Intranets and Extranets. Artech House.
- Rawlings, M. (2001). Setting the Stage – Understanding ebXML in Context retrieved from the World Wide Web: http://www.rawlingsseccondsulting.com.
- Rawlings, M., Crawford, M., Rudie, D., Warner, T., Itoh, K., Kubler, J., et al. (2001). ebXML Requirements Specification v 1.06. Retrieved March 8, 2003, from http://www.ebxml.org/specs/ebREQ.pdf.
- Rawlings, M.C. (2001^{2}). Market Impact of the ebXML Infrastructure Specifications. Retrieved March 14, 2003 from http://www.rawlinsecconsulting.com
- Rawlings, M.C. (2001^{1}). Overview of the ebXML Architectures. Retrieved March 14, 2003 from http://www.rawlinsecconsulting.com
- Rawlings, M.C. (2001^{3}). ebXML and Interoperability. Retrieved March 14, 2003 from http://www.rawlinsecconsulting.com
- Rawlings, M.C. (2002). ebXML – The Road Ahead. Retrieved March 14, 2003 from http://www.rawlinsecconsulting.com
- Rayman, C. (1999). NTCIP Introduction/ NTCIP: Infrastructure Interoperability. Retrieved July 11, 2003, from http://www.itscanada.ca/html/AGM1999/present/ntcip/
- *Rosencrance, L. (2002). SAML Secures Web Services. Retrieved July 20, 2003, from http://www.computerworld.com/securitytopics/security/story/0,10801,73712,00.html*
- RosettaNet Consortium. (2002). RosettaNet Implementation Framework: Core Specification, Standards Specification Version: V02.00.01, RosettaNet.
- Sankar, K., & Najmi, F. (2001). ebXML Registry Security Proposal. Retrieved March 8, 2003, from http://www.ebxml.org/specs/ebTA.pdf.
- Schneider, G.P., & Perry, J.T. (2001). Electronic Commerce. Course Technology'
- Seely S (2002). SOAP – Cross-platform Web Service Development Using XML. Prentice Hall PTR.

- Siddiqui, B. (2002). Exploring XML Encryption. Retrieved September 7, 2003 from http://www-106.ibm.com/developerworks/xml/library/x-encrypt/
- Snell J, Tidwel D and Kulchenko P (2002). Programming Web services with SOAP. O'Reilly.
- Taschek, J. (2002). The Big hype for the New Year: ebXML. EBSCO Host Research Database, e-week. Vol.19. Issue 2.
- Taylor, D. (2002). Data Interoperability. *The S.A e-Enterprise Map*, 42-43.
- TechTarget Network (2003). File Transfer Protocol (FTP). Retrieved August 5, 2003 from http://searchtechtarget.techtarget.com
- W3C (1985). File Transfer Protocol (FTP) - File Transfer Protocol Specification. Retrieved August 5, 2003 from http://www.wc3.org
- W3C (2000). SOAP (Simple Object Access Protocol) 1.1 Specification. Retrieved May 15, 2003 from http://www.w3.org/TR/2000/NOTE-SOAP-20000508
- W3C (2001). SMTP (Simple Mail Transfer Protocol) Specification. Retrieved August 5, 2003 from http://www.w3.org
- W3C (2002) ^{1}. SOAP Version 1.2 Email Binding. W3C Note 26 June 2002; Retrieved July 2002 from http://www.w3.org/TR/2002/NOTE-soap12-email-20020626
- W3C (2002). SOAP 1.2 Attachment Feature. W3C Working Draft 24 September 2002 http://www.w3.org/TR/2002/WD-soap12-af-20020924
- W3C (2003)^{1}. SOAP ver 1.2 Part 0: Primer. Retrieved December 15, 2003 from http://www.w3.org/TR/2003/REC-soap12-part0-20030624/
- W3C (2003)^{2}. SOAP ver 1.2 Part 1: Primer. Retrieved December 15, 2003 from http://www.w3.org/TR/2003/REC-soap12-part0-20030624/
- W3C (2003)$^{3.}$ SOAP ver 1.2 Part 2: Primer. Retrieved December 15, 2003 from http://www.w3.org/TR/2003/REC-soap12-part0-20030624/
- W3C (2004)^{1}. MTOM (SOAP Message Transmission Optimization Mechanism). Retrieved August 15, 2004, from http://www.w3.org/TR/2004/WD-soap12-mtom-20040608/

- W3C (2004)^{2}. Extensible Markup Language (XML) 1.1. W3C Recommendation 04 February 2004, edited in place 15 April 2004. Retrieved: 30 October 2004 from http://www.w3.org/TR/2004/REC-xml11-20040204/ Editors: Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan
- W3C (2004)$^{3.}$ Web Services Choreography Requirements. W3C Working Draft, 11 March 2004. Editors: Austin, D., Barbir, A., Peters, W. and Ross-Talbot, S. Retrieved 25 October, 2004 from http://www.w3c.org/TR/2004/WD-ws-chor-reqs-20040311/
- W3C (2004)^{4}. Web Services Architecture. Working Group Note 11 February 2004. Retrieved 20 June 2004 from http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/
- W3C (2004)^{5}. XML Information Set. W3C Recommendation, 4 February 2004. Retrieved June 2, 2004 from http://www.w3.org/TR/2004/REC-xml-infoset-20040204
- W3C (2004)^{6}. XML-binary Optimized Packaging. W3C Candidate Recommendation $Date: 2004/09/15 12:33:28 $ Retrieved December 17, 2004 from http://www.w3.org/TR/xop10/
- W3C (2005). XML-binary Optimized Packaging Recommendation. 25 January 2005. Retrieved from http://www.w3.org/TR/2005/REC-xop10-20050125/ 30 January 2005.
- W3C. (1997). HTTP (Hypertext Transfer Protocol) 1.1. Retrieved August 5, 2003.
- Wall L and Lader A (2002). Building Web Services and .NET Applications. McGraw-Hill/Osborne.
- Webopedia (2003). IPSec. Retrieved August 6, 2003 from http://www.webopedia.com
- Wiehler G (2004). Mobility, Security and Web Services. Siemens Business Services. Publicus Corporate Publishing.
- Whitman, M. E., & Mattord, H. J. (2003). Principles of Information Security. Canada: Thomson Course Technology.
- Williams, H., Whalley, J., & Li, F. (2000). Interoperability and Electronic Commerce: A New Policy Framework for Evaluating Strategic Options.

Journal of Computer-Mediated Communication 5, (3). Retrieved June 2, 2003, from http://www.ascusc.org/jcmc/vol5/issue3/williams.html

- Zhao X., Xie J., Zhang W.J. (2003). The impact of information sharing and ordering co-ordination on supply chain performance. MCB University Press ISSN 1359-8546 Supply Chain Management: An International Journal; Volume 7 Number 1 pp. 24-40

Acknowledgement

The financial assistance of the National Research Foundation (NRF) in South Africa towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the National Research Foundation.

APPENDIX A:

Sample (C#) code for the “Smart” Global Message Handler:

```
/*
 *  The Smart Messenger System .
*/

/*
 * The MessageHTTPHandler which functions as the global Message Handler
*/

using System;
using System.IO;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Messaging;

using System.Xml;
using System.Xml.Serialization;

using Microsoft.Web.Services2;
using Microsoft.Web.Services2.Attachments;
using Microsoft.Web.Services2.Messaging;
using Microsoft.Web.Services2.Addressing;

using MessageHandler;
using SM.Data.HandlerAccess;

namespace MessageService
{
        [SoapService("http://localhost/MessageService")]
        public class MessageHttpHandler : SoapService
        {
                HandlerAccess access;

                #region validate( string To, string From, XmlDocument Message, string SoapID )
                /// <summary>
                /// The method validates the User before sending the message to
                /// the business partner
                /// </summary>
                /// <param name="To">The Business partners Username</param>
                /// <param name="From">The Users Username</param>
                /// <param name="Message">The XmlDocument contained in the
body object</param>
                /// <param name="SoapID">The ID of the Soap message</param>
                /// <returns>bool valid\invalid</returns>
                private bool validate( string To, string From, XmlDocument Message,
string SoapID )
                {
                    bool valid = false;
                    access = new HandlerAccess();
                    string message = Message.InnerXml;

                    //Validate User
                    valid = access.validUser( From );

                    //Log success\failure into the database
                    if ( valid )
                        access.logLoginSuccess( To, From, message, "Success",
SoapID );
                    else
                        access.logLoginFailed( To, From, message, "Failed",
SoapID );

                    return valid;
                }
                #endregion

                #region validateReceived( string To, string From, XmlDocument
Message, string SoapID )
                /// <summary>
                /// The method validates the business partner before sending the
message
                /// to the end point function
                /// </summary>
                /// <param name="To">The Business partners Username</param>
                /// <param name="From">The Users (senders) Username</param>
                /// <param name="Message">The XmlDocument contained in the
body object</param>
                /// <param name="SoapID">The ID of the Soap message</param>
                /// <returns>bool valid\invalid</returns>
                private bool validateReceived( string To, string From, XmlDocument
Message, string SoapID )
                {
                    bool valid = false;
                    access = new HandlerAccess();
                    string message = Message.InnerXml;

                    //Validate business partner
                    valid = access.validUser( To );
                    //Log success\failure into the database
                    if ( valid )
                        access.logReceivedSuccess(    To,    From,    message,
"Success", SoapID );
                    else
                        access.logReceivedFailed( To, From, message, "Failed",
SoapID );

                    return valid;
                }
                #endregion

                #region SendMessage( SoapEnvelope envelope )
                /// <summary>
                /// The method is used to send Soap messages to business partners
                /// </summary>
                /// <param name="envelope">The Soap envelope</param>
                /// <returns>The Soap envelope</returns>
                [SoapMethod("SendMessage")]
                public SoapEnvelope SendMessage( SoapEnvelope envelope )
                {
                    //Create new envelope to send response back to the User
                    SoapEnvelope responseEnvelope = new SoapEnvelope();

                    string To = "";
                    string From = "";
                    string ToUri = "";
                    string FromUri = "";
                    string ValidateParts = "";
                    bool valid = false;

                    //Verify message parts
                    ValidateParts = VerifyMessageParts( envelope.Context );

                    //Return error if invalid
                    if ( !ValidateParts.Equals( "null" ) )
                    {
                        responseEnvelope.SetBodyObject( ValidateParts );
                        return responseEnvelope;
                    }

                    //Create new Xml document and assign sent Xml document
                    XmlDocument doc = new XmlDocument();
                    doc.InnerXml = envelope.Body.InnerText;

                    //To simplify name resolution Usernames To\From are added to
the Xml
                    //document, Usernames are extracted
                    //TODO: implement name resolution table
                    To = doc.DocumentElement["To"].InnerText;
                    From = doc.DocumentElement["From"].InnerText;

                    //Retreive Soap ID, validate the User and log information
                    string SoapID =
envelope.Context.Addressing.MessageID.Value.ToString();
                    valid = validate( To, From, doc, SoapID );

                    //Return error if invalid
                    if ( !valid )
                    {
                        responseEnvelope.SetBodyObject( "Unable to validate
User" );
                        return responseEnvelope;
                    }

                    //Create new envelope to send response to business partner
                    SoapEnvelope requestEnvelope = new SoapEnvelope();
                    requestEnvelope.SetBodyObject( envelope.Body.InnerText );

                    //Set To\From string of the Message Handler
                    ToUri = doc.DocumentElement["ToUri"].InnerText;
                    FromUri = doc.DocumentElement["ToUri"].InnerText;

                    MessageHandler.MessageHttpHandler remoteMessageService
= new MessageHandler.MessageHttpHandler( ToUri );

                    //Testing not yet implemented
                    //SoapSender sender1 = new SoapSender( new Uri( ToUri ) );
                    //sender1.Send( requestEnvelope );

                    //Try sending the message to business partners Message
Handler
                    try
                    {
                        IAsyncResult ar =
remoteMessageService.BeginReceiveMessages( requestEnvelope, null, null );

                        ar.AsyncWaitHandle.WaitOne();

                        if (
remoteMessageService.EndDeliverMessage(ar).Body.InnerText.Equals("Message
received") )
                    responseEnvelope.SetBodyObject( "Message
sent" );
                        else
                    responseEnvelope.SetBodyObject( "Unable to
deliver message" );
                    }
                    catch( AsynchronousOperationException
asynchronousException )
```

```
            {
                System.Net.WebException webException =
(System.Net.WebException)asynchronousException.InnerException;

                responseEnvelope.SetBodyObject(
webException.ToString() );
            }

            //Send response to User indicating success\failure
            return responseEnvelope;
        }
        #endregion

        #region ReceiveMessages( SoapEnvelope envelope )
        /// <summary>
        /// The method is used to receive Soap messages from business partners
        /// </summary>
        /// <param name="envelope">The Soap envelope</param>
        /// <returns>The Soap envelope</returns>
        [SoapMethod("ReceiveMessages")]
        public SoapEnvelope ReceiveMessages( SoapEnvelope envelope )
        {
            string To = "";
            string From = "";
            bool valid = false;

            //Create Xml document
            XmlDocument doc = new XmlDocument();
            doc.InnerXml = envelope.Body.InnerText;

            //Load Usernames
            To = doc.DocumentElement["To"].InnerText;
            From = doc.DocumentElement["From"].InnerText;

            //Retreive Soap ID
            string SoapID =
envelope.Context.Addressing.MessageID.Value.ToString();

            //Validate User and log information
            valid = validateReceived( To, From, doc, SoapID );

            //Create Soap envelope to send response to sending user
            SoapEnvelope responseEnvelope = new SoapEnvelope();

            if ( valid )
                responseEnvelope.SetBodyObject( "Message received"
);
            else
                responseEnvelope.SetBodyObject( "Unable to process
received message" );
```

```
                    //... send to end point function

                    //Send response message to sending User (Mid project judging;
                    //code removed for handin to show system implementation)

                    //Send notification of success/failure
                    return responseEnvelope;
            }
            #endregion

            #region VerifyMessageParts( SoapContext context )
            /// <summary>
            /// The method verifies that the Soap header contains the correct
            /// attributes
            /// </summary>
            /// <param name="context">Soap envelope context</param>
            /// <returns>string error\null</returns>
            public string VerifyMessageParts( SoapContext context )
            {
                    if ( context.Envelope.Body.Equals( null ) )
                            return "The Message does must contain a Body
element";

                    if ( context.Addressing.To.Equals( null ) )
                            return "The Message does must contain a Addressing
To element";

                    if ( context.Addressing.Action.Equals( null ) )
                            return "The Message does must contain a Addressing
Action element";

                    if ( context.Addressing.MessageID.Equals( null ) )
                            return "The Message does must contain a Addressing
MessageID element";

                    if ( context.Security.Timestamp.Equals( null )  )
                            return "The Message does must contain a Security
Timestamp element";

                    return "null";
            }
            #endregion
    }
}
```