SWWS – Semantic Web Enabled Web Services

Title:
Analysis of B2B Standards and Systems

Version: 2.0
Date: 2004-03-29
Pages: 77

Responsible Author: FZI
Andreas Abecker

Co-Author(s):
Rainer Tellmann
Stephan Grimm

Status:	**Confidentiality:**		
[ ] Draft	[X]	Public	- for public use
[ ] To be reviewed	[ ]	INT	- for SWWS consortium (and Project Officer if requested)
[ ] Proposal			
[X] Final / Released to CEC	[ ]	Restricted	- for SWWS consortium and Project Officer only

Project ID: IST-2001-37134

Deliverable ID: D1.1

Workpackage No: WP1

Title: Analysis of B2B Standards and Systems

Summary / Contents:

In this deliverable we describe a state-of-the-art overview and detailed analysis of existing B2B standards and systems. The term B2B refers to communication of business data over the network. At its simplest, B2B integration is the automated exchange of information between different organizations. This includes product catalogs, classification systems, B2B protocol standards, synchronous/asynchronous communication, process coordination, backend integration as well as further topics.

SWWS Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2002-37134. The partners in this project are: Leopold-Franzens Universität Innsbruck (IFI, Austria)); National University of Ireland, Galway (NUI, Galway, Ireland); Forschungszentrum Informatik (FZI, Germany); Intelligent Software Components S.A. (iSOCO, Spain); OntoText Lab. - Sirma AI Ltd. (SAI, Bulgaria); Hewlett Packard (HP, UK), British Telecom (BT, UK)

Leopold-Franzens Universität Innsbruck (IFI)

Institut für Informatik
Technikerstrasse 13
A-6020 Innsbruck Austria

Tel: +43 512 507 6489
Fax: +43 512 507 9872

Contact person: Holger Lausen
E-mail: holger.lausen@deri.ie

National University of Ireland, Galway (NUI)

National University of Ireland,
University Road
Galway, Ireland

Tel: +353 91 512603
Fax: +353 91 512541

Contact person: Juan Miguel Gomez
E-mail: juan.miguel@deri.ie

FZI – Forschungszentrum Informatik

Haid-und-Neu-Str. 10-14
76131 Karlsruhe, Germany

Tel: +49 721 9654816
Fax: +49 721 9654817
Contact person: Stephan Grimm
E-mail: stephan.grimm@fzi.de

Intelligent Software Components S.A. (iSOCO)

Francisco Delgado 11, 2nd Flor
28108 Alcobendas, Madrid, Spain

Tel: +34 913 349797
Fax: +34 913 349799

Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

OntoText Lab.- Sirma AI Ltd. (SAI)

OntoText Lab.

38A Chr. Botev Blvd.
Sofia 1000, Bulgaria

Tel: +35 92 9768 303,
Fax: +35 92 9768 311

Contact person: Atanas Kiryakov
E-mail: Atanas.Kiryakov@sirma.bg

Hewlett Packard (HP)

HP European Laboratories
Filton Road, Stoke Gifford
BS34 8QZ Bristol, UK

Tel: +44 117 3128631
Fax: +44 117 3129285

Contact person: Janet Bruten
E-mail: janet.bruten@hp.com

Associated Partner:

British Telecommunications plc. (BT)

Orion 5/12, Adastral Park
Ipswich ip5 3RE, UK

Tel: +44 1473 609583
Fax: +44 1473 609832

Contact person: John Davies
E-mail: john.nj.davies@bt.com

Table of Contents

List of Figures

Change Log

Vers.	Date	Author	Description
1.1	08.10.2002	A. Maedche	Document created
1.2	30.01.2003	R. Tellmann	First version of document finished
1.3	28.02.2003	R. Tellmann	Final version
2.0	29.03.2004	S. Grimm	First Refinement

List of Acronyms and Abbreviations

Acronym/ abbreviation	Resolution
A2A	Administration-to(2)-Administration
B2B	Business-to(2)-Business
B2C	Business-to(2)-Customer
BIE	business information entity
BOD	Business Object Documents
BPD	Business Process Diagram
BPEL4WS	Business Process Execution Language for Web Services
BPML	Business Process Modelling Language
BPMN	Business Process Modelling Notation
BPMS	Business Process Management Systems
BPQL	Business Process Query Language
BPSS	Business Process Specification Schema
BTP	Business Transaction Protocol
CCA	Component Collaboration Architecture
CPA	Collaboration Protocol Agreements
CPP	Collaboration Protocol Profiles
cXML	commerce eXtensible Markup Language
DTD	data type definition
EAI	Enterprise Application Integration
ebXML	Electronic Business XML
eCX	Electronic Catalog XML
EDI	Electronic Data Interchange
EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport
EDOC	Enterprise Distributed Object Computing
ERP	Enterprise Resource Planning
FIX	Financial Information Exchange
GTIN	Global Trade Item Number
HTTP	Hypertext Transfer Protocol
IAPSO	Inter-Agency Procurement Services Organization
JMS	Java Message Service
MDA	Model Driven Architecture
MIME	Multipurpose Internet Mail Extensions
OAGIS	OAGI Integration Specification
OAMAS	Open Application Middleware API Specification

OASIS	Organization for the Advancement of Structured Information Standards
OCF	Open Catalog Format
OCP	Open Catalog Protocol
PIM	Platform-Independent Model
PIP	Partner Interface Processes
PSL	Process Specification Language
PSM	Platform-Specific Model
RNIF	RosettaNet Implementation Framework
SMIME	Secure Multipurpose Internet Mail Extensions
SOAP	Simple Object Access Protocol
STEP	Standard for the Exchange of Product Model Data
STP	Simple Transport Protocol
SWIFT	Society for Worldwide Interbank Financial Telecommunication
TPA	Tading Partner Agreement
UBL	Universal Business Language
UML	Unified Modeling Language
UMM	Unified Modeling Methodology
UNCCS	United Nations Common Coding System
UNSPSC	United Nations Standard Products and Services Code
VAN	Value Added Network
VPN	Virtual Private Network
WfMC	Workflow Management Coalition
WFMS	Workflow Management System
WSCI	Web Service Choreography Interface
WSFL	Web Services Flow Language
XAML	Transaction Authority Markup Language
xCBL	XML Common Business Library
XLANG	XML-based language
XML	eXtensible Markup Language
XPDL	XML Processing Description Language

Executive Summary

In this deliverable we describe a state-of-the-art overview and detailed analysis of existing B2B standards and systems. The term B2B refers to communication of business data over the network. At its simplest, B2B integration is the automated exchange of information between different organizations. This includes product catalogs, classification systems, B2B protocol standards, synchronous/asynchronous communication, process coordination, backend integration as well as further topics.

Occurring independent of or alongside manual processes, B2B integration is most accurately described as application-to-application integration that crosses corporate boundaries (e.g. firewalls). Increasingly, this integration is being done over the Internet, rather than over proprietary Value Added Networks (VANs), and the dominant trend is towards the use of open standards such as XML and HTTP, rather than proprietary protocols that are not well suited to the Internet.

At its most effective, B2B integration improves external processes such as supply chain integration or shipping/logistics tracking by enabling rapid, cost-effective real-time links between business partners. It enables new business paradigms such as e-commerce initiatives. It reduces costs and inefficiencies by facilitating initiatives such as multi-vendor catalogs and electronic procurement - promoting comparison shopping and dramatically reducing the costs associated with traditional procurement. And it strengthens customer relationships by enabling capabilities such as real-time order management and customer service.

B2B standards can be roughly classified according to the following topics:

- Catalogue & Classification
- Document Exchange
- Collaboration
- Business Processes
- Business Transactions

Besides a detailed and elaborated discussion about B2B standards, this report gives an overview on the current state-of-the-art of B2B systems available on the market.

1 Introduction

In this deliverable document we describe a state-of-the-art overview and detailed analysis of existing B2B standards and systems. The term B2B refers to communication of business data over the network. At its simplest, B2B integration is the automated exchange of information between different organizations. This includes product catalogs, classification systems, B2B protocol standards, synchronous/asynchronous communication, process coordination, backend integration as well as further topics.

Occurring independent of or alongside manual processes, B2B integration is most accurately described as application-to-application integration that crosses corporate boundaries (e.g. firewalls). Increasingly, this integration is being done over the Internet, rather than over proprietary Value Added Networks (VANs), and the dominant trend is towards the use of open standards such as XML (Extensible Markup Language) and HTTP (Hypertext Transfer Protocol), rather than proprietary protocols that are not well suited to the Internet.

At its most effective, B2B integration improves external processes such as supply chain integration or shipping/logistics tracking by enabling rapid, cost-effective real-time links between business partners. It enables new business paradigms such as e-commerce initiatives. It reduces costs and inefficiencies by facilitating initiatives such as multi-vendor catalogs and electronic procurement - promoting comparison shopping and dramatically reducing the costs associated with traditional procurement. And it strengthens customer relationships by enabling capabilities such as real-time order management and customer service.

The B2B standards described within this deliverable enable an enterprise to become an efficient E-Business. Many heterogeneous applications as illustrated in Figure 1 have to be integrated, including E-commerce web sites, portals, supply chain management, procurement management, online market places, customer relationship management and enterprise resource planning.

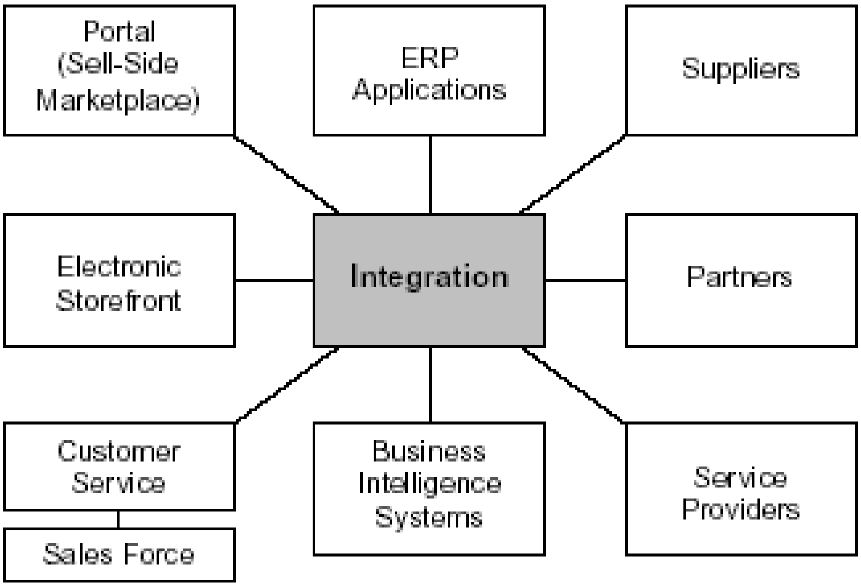


Figure 1 E-Business Integration

Section 3 gives a detailed overview about Workflow Systems and section 5 provides an overview about B2B integration systems which use the B2B standards described in section 2. Business Integration software gives you the ability to integrate the diverse data and information sources both within and outside your enterprise into a single coherent framework. An integrated information infrastructure can then be shared by mission-critical applications such as CRM, executive information portals, and automated supply chain systems.

2 B2B Standards

Modern B2B standards are associated to levels in a stack like the one depicted in Figure 2. It includes transport, catalog- and classification standards. On top of these standards the "Business Process-Transaction" standards are layered. Additionally overarching topics like security in two different views (business view, technical view) are taken into account.
<blabla - welche Aussagen will man denn hier eigentlich treffen???>

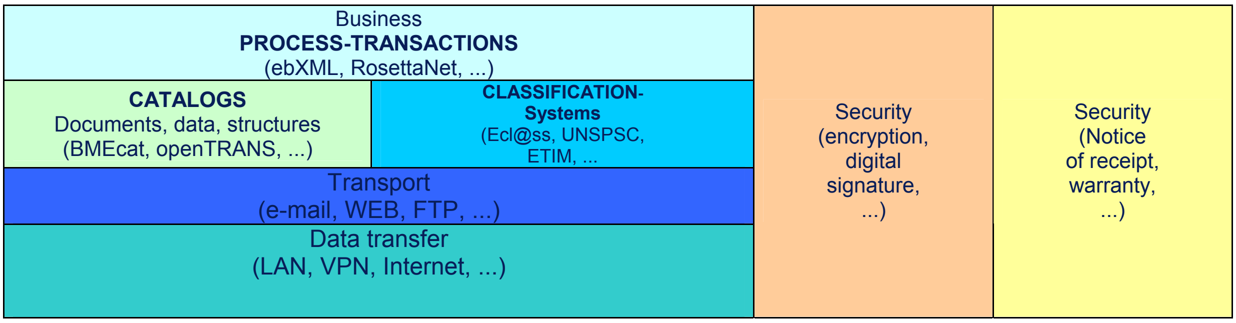


Figure 2: Overview on B2B Standards

B2B standards can be roughly classified according to the following topics:

- Catalogue & Classification
- Document Exchange
- Collaboration
- Business Processes
- Business Transactions

Figure 3 shows how all these aspects affect electronically automated cooperation in an abstract B2B eCommerce scenario.

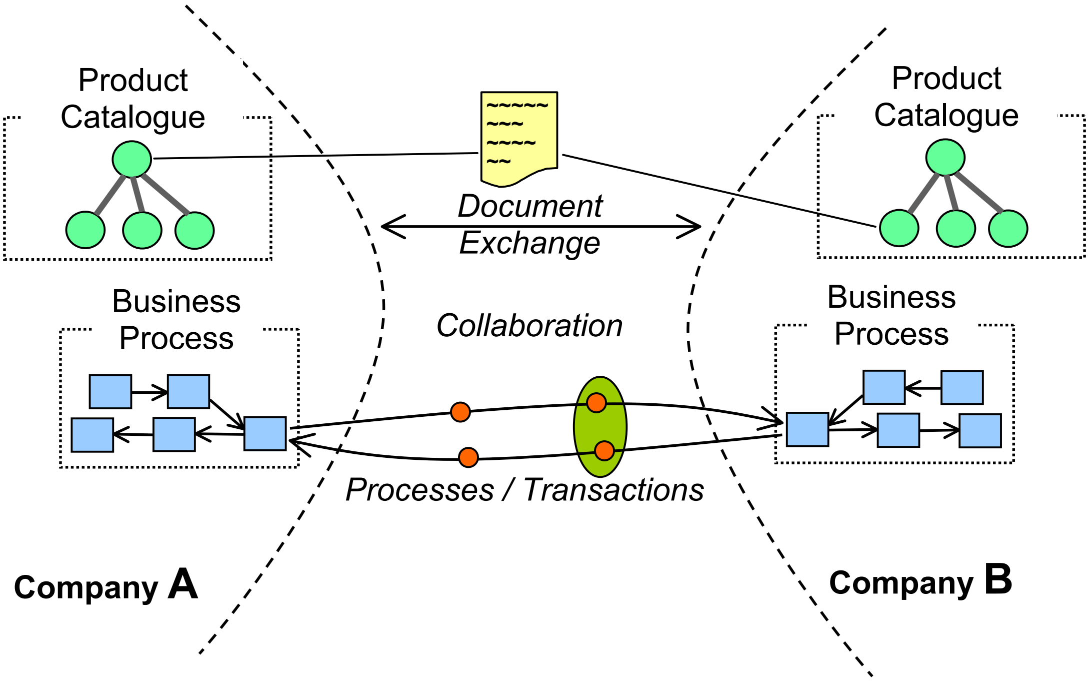


Figure 3: An abstract B2B eCommerce scenario

To achieve B2B collaboration companies use standards for document exchange and business processes and transactions. This way the internal processes of several companies can be integrated with each other while maintaining a transaction context. Document exchange is supported by standards for product catalogue classification.
This section gives an overview on modern B2B standards with regard to the above classification.

2.1 Catalog & Classification standards

This section describes catalog as well as classification standards. Topics related to catalogs standards handle problems like the exchange of product data catalogues. Complementary to catalog standards, classification codes are used to group similar things into common categories. With classification, similar things are members of a class. Similar classes are members of yet a more general class or family, and so on. The relationship among things and the relationship of a thing to its class are information signals that are necessary for item discovery, spend analysis, and product awareness. In other words, classification codes are necessary for effectively searching and finding appropriate products and services, for identifying where expenditures are being made, and for promoting ones products to real buying prospects [UNSPSCwp].

There are more but less wide spreaded standards (peculiar to any particular trade - e.g. ETIM in the electro-technics, etc) that are not further discussed in this document.

2.1.1 Catalogue-Systems

The goal of standardization efforts in the field of catalog-systems is the facilitation of trade related to the exchange of product data catalogues between suppliers and purchasing organizations or to solve the problem of exchanging product information and catalog structures between different catalog systems.

A product catalog is a collection of product information. Often a catalog consists of a hierarchy of product categories. Each product category contains a list of products. Each product is described by a set of attributes which are defined by the product's category.

For non-standardized product catalogs, or in situations where several standards have to be integrated, there is the problem of mediation between different product description schemata. Facts that are expressed syntactically different in two schemas can be sematically equal. Since the classification hierarchies in product catalogs are very similar to taxonomic structures in ontologies, their integration can be realized by ontology mapping techniques, as mentioned in [B2BOnt]. There, semi-automatic mapping tools such as Chimeara [Chimaera] or PROMPT [PRMT] are suggested to be used for supporting a user in establishing the mapping between two different schemas, however, they don’t meet the requirements of providing being fully automated product data integration on the fly.

2.1.1.1 BMEcat

The BMEcat-format [BMEcat] was developed with the objective of standardizing the exchange of product data catalogues between suppliers and purchasing organizations, thus simplifying this procedure. In the basic model, a supplier compiles a catalogue in electronic form which complies to the BMEcat standard. In the following text this catalogue is termed the catalogue document. This catalogue document also enables the integration of multi-media product data such as photos, graphics, technical documentation, video data etc.

Typically, a supplier transfers the catalogue document to a purchasing organization which further processes the contents of the catalogue document and integrates it, for example, into an existing shop system (suppliers of such shop systems for the field of procurement are, for example, SAP, Intershop, Harbinger, Ariba, Commerce One, Procure Network, Healy Hudson etc). This procedure is known as product data exchange. The BMEcat- format enables the supplier not only to transfer his complete product data using such a product data exchange procedure, but also to update price data, for example.

BMEcat offers even more possibilities for the sales side. Apart from being used to transfer data, the standardized BMEcat catalogue document can also be put to use in order to compile or update a purchaser's own online shop for sales support.

2.1.1.2 eCX (Electronic Catalog XML)

The XML-based catalog format eCX [eCX] was developed to solve the problem of exchanging product information and catalog structure between different catalog systems. Its associated data type definition (DTD) is concerned with the description and definition of catalog structure, or schema, and their related items. eCX is currently used by Requisite Technology and its Partners for e-Catalog exchange.

It is based exclusively on the exchange of catalog information and multi-vendor catalog interoperability. The purpose of the e-Catalog XML specification is to provide a method of updating a catalog's structure, or schema, and its e-Content from a variety of sources and content formats. The XML format has been designed to allow import of e-Content into any catalog format. It supports the following functionality:

- **Modification of catalog schema**
  Add, update, and delete a category
  Add, update, and delete a common attribute, or a category attribute

- **Modification of catalog products**
  Add, update, and delete products
  Move products between categories
  Copy products to new categories

- **Support for Internationalized data**
  This specification allows the setting of default languages by use of the *xml lang* attribute on the *CATALOG element*.

Electronic Catalog XML (eCX XML) 3.0 Specifications [eCLSp]

2.1.1.3 CatXML

CatXML™ is a public open software solution developed by a team of XML consultants to the US Government Defense Logistics Agency (DLA) EMALL project, aided in part by XML Global Technologies, Inc.[1] who have implemented a live server site.

The DLA EMALL requirements are targeted particularly at small businesses with minimal E-Business resources, but who provide a very significant level of service to the U.S. armed services. Specifically, the EMALL project is designed to offer the four services (Army, Navy, Airforce, and Marines) a single requisition system for common parts and replenishable supplies.

2.1.1.4 OCP (Open Catalog Protocol)

OCP[2] (Open Catalog Protocol) [MART] is an XML-based software protocol that enables the exchange of complex data between product/service catalogs. OCP consists of a language-independent representation of catalog data based on XML 1.0, and a set of protocol opcodes for specifying a wide variety of operations on a catalog. OCP itself does not include any transport mechanisms. It relies on lower level transport protocols such as HTTP or STP

[1] http://www.xmlglobal.com

[2] http://www.martsoft.com/ocp/

(Simple Transport Protocol). The catalog content is described by Open Catalog Format (OCF).

The Open Catalog Format (OCF) is an open standard for describing product catalogs. OCF is used to represent, store, and transport product information. OCF may represent a single product, a single category, or a subset of, or the entire product catalog.

OCF is a generic product representation language. It does not define any specific categorization schemes. For instance, OCF does not define the category hierarchy or the product attribute names of a specific catalog. Instead, they are defined by the users. Therefore, OCF can be used to implement content specific standards such as the specifications from RosettaNet (see 2.3.2).

For further information related to OCP, e.g. the "Elements of OCF": http://www.martsoft.com/ocf/.

2.1.2 Classification-Standards

The goal of classification-standards is the numerically identification of products and services. Often they are composed of multilevel, hierarchical classification-keys. Coding products and services according to a standardized classification convention is necessary for streamlining commerce among companies. Products and services that are unambiguously identified with industry-agreed upon names allows purchasing management to effectively source and analyse expenditures. In addition, machine-readable product names assists marketing and sales functions to find customers and provide better customer and distribution channel services.

By inserting the codes in various electronic trade documents and media such as product catalogs, Web sites, purchase orders, invoices, inventory/sales advices, and others, computer applications throughout an extended supply chain (seller, buyer, distributor, independent sales representative, end user) can process transaction data automatically and can perform management, analysis and decision functions in time-critical and labor-efficient ways that would not be possible without the codes.

2.1.2.1 Ecl@ss

eCl@ss [ECLASS] is a classification system with standard sets of attributes and key words tailored to the needs of industrial customers and their suppliers. It supports the flow of products and information along the supply chain of an industrial enterprise.

eCl@ss is characterised by a 4-level hierarchical classification system with a key-word register of 12,000 words. eCl@ss maps market structure for industrial buyers and supports engineers at development, planning and maintenance. Through the access either via the hierarchy or over the key words both the expert as well as the occasional user can navigate in the classification. An unique feature of eCl@ss is the integration of attribute lists for the description of material and service specifications.

2.1.2.2 UNSPSC

The United Nations Standard Products and Services Code (UNSPSC[3]) enables users to consistently classify the products and services they buy and sell. UNSPSC is the result of a merger of the United Nations Common Coding System (UNCCS) and Dun & Bradstreet's Standard Product and Services Codes (SPSC). The merger was completed in 1999 through the efforts of a team of analysts and researchers from both D&B and the Inter-Agency Procurement Services Organization (IAPSO) of the UNDP. UNSPSC is considered an open standard, and companies and other organizations worldwide are encouraged to apply the codes in their business systems.

2.1.3 Concluding remark

In SWWS product classification and catalog systems are not directly focused on since the project case studies are more service oriented than related to purchase and e-procurement. Within the modeling of semantic web services they are contained in domain ontologies covering knowledge about classification hierarchies and product relations. Taxonomies, as used for product classification hierarchies, perfectly fit into the modeling techniques offered by ontology languages such that all their aspects are fully covered.

2.2 Document exchange

An interesting aspect of business to business Internet commerce is that of open trading communities, marketplaces, or "virtual enterprises" in which buyers and suppliers of goods and services discover each other, exchange information, conduct transactions, etc.

The essential benefit of open trading communities or marketplaces is that they offer buyers the largest set of possible suppliers, each of whom has the largest possible market. Each relationship between a supplier's catalog and "back end" processing system and a buyer's purchasing application no longer requires a point-to-point custom integration and yet another document format. Instead, once a company joins an community, its requests for quotes, catalogs and services are potentially available to every other participant, with no incremental integration cost to itself as new companies join, regardless of the buying or selling application each uses.

The goal of creating marketplaces or virtual enterprises by interconnecting business systems is not new. Ideally, companies could conduct electronic commerce in a completely ad hoc fashion, without prior agreement of any kind, and proposals for "Open EDI" and "Plug and Play Commerce" on the Internet predate the XML groundswell of the past few years. But prior to XML, the technology foundations for this vision of electronic commerce simply weren't capable of making it happen.

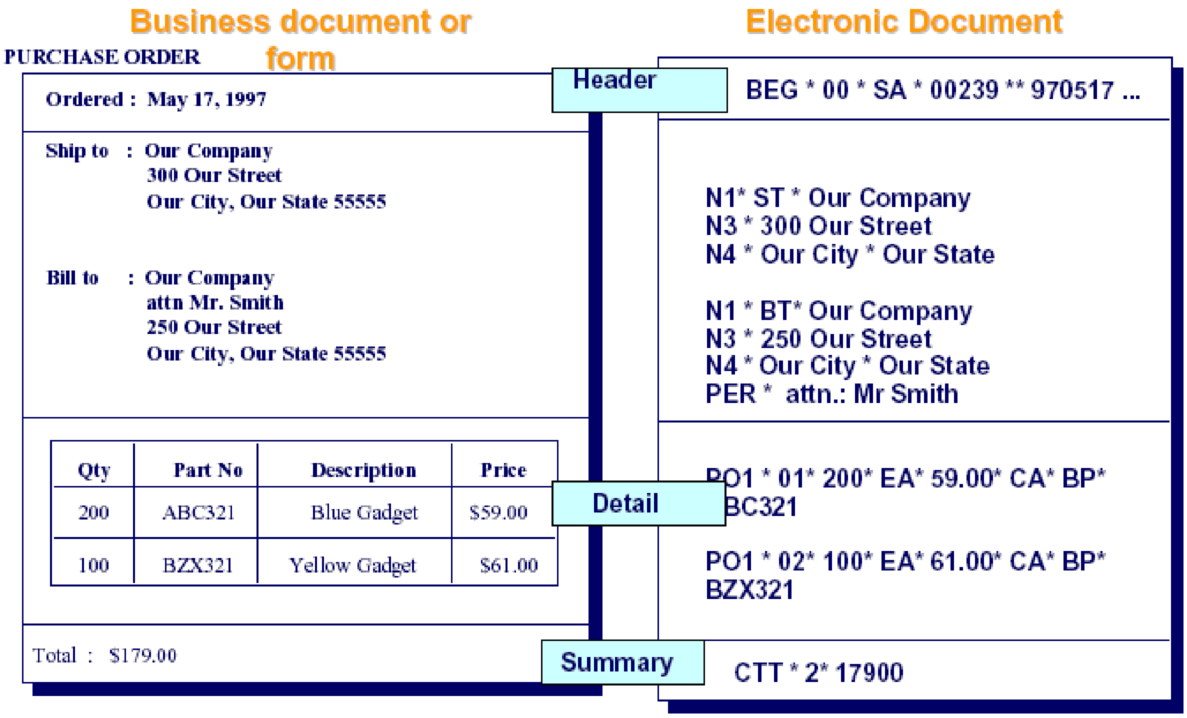
[3] United Nations Standard Products and Services Code

2.2.1 EDI

EDI (Electronic Data Interchange) is the process for exchanging data in electronic formats between heterogeneous applications in a manner that can be processed without manual intervention. Originally it was invented as an substitute for paper. This origin is depicted in Figure 4.
Because of some limitations of EDI, for example:

- High expense factor because most companies have to develop their own conversation programs

- EDI is based on fixed transaction sets and the business rules are embedded into these sets. But business rules may vary from company to company. Thus two trading partners must agree on a common standards.



PURCHASE ORDER

Ordered : May 17, 1997

Ship to : Our Company
300 Our Street
Our City, Our State 55555

Bill to : Our Company
attn Mr. Smith
250 Our Street
Our City, Our State 55555

Qty	Part No	Description	Price
200	ABC321	Blue Gadget	$59.00
100	BZX321	Yellow Gadget	$61.00

Total : $179.00

BEG * 00 * SA * 00239 ** 970517 ...

N1* ST * Our Company
N3 * 300 Our Street
N4 * Our City * Our State

N1 * BT* Our Company
N3 * 250 Our Street
N4 * Our City * Our State
PER * attn.: Mr Smith

PO1 * 01* 200* EA* 59.00* CA* BP* BC321

PO1 * 02* 100* EA* 61.00* CA* BP* BZX321

CTT * 2* 17900

Figure 4 Paper vs EDI document

necessary enhancements were required. This raises the need for modern (e.g. based on XML) languages as well as composition standards such as BPEL4WS, WSCI, or BPML.

For further information about some traditional EDI standards look at: www.diffuse.org/edi.html

2.2.2 EDIFACT

The acronym EDIFACT stems from the United Nations rules for **Electronic Data Interchange For Administration, Commerce and Transport**. They comprise a set of internationally agreed standards, directories and guidelines for the electronic interchange of structured data, and in particular that related to trade in goods and services between independent, computerized information systems (because there exists more up to date standards like e.g. ebXML we refer for further information to: http://www.unece.org/trade/untdid/welcome.htm)

2.2.3 XML

Almost every standard mentioned in this deliverable relies on XML. The essence of using XML to implement a trading community or marketplace is for a "market operator" or "market maker" to define the "community standards" for business documents and the protocols for exchanging and routing messages within the community. Then, buyers, suppliers, or other service providers like shippers or payment acquirers can participate if they can produce and consume those documents.

XML (eXtensible Markup Language) is a meta-markup language used to create syntaxes for languages. It is also a standard for passing data between applications, particularly those that communicate across the Internet.

XML documents contain data in the form of tag/value pairs, for example [cXMLUG]:

<DeliverTo>Joe Smith</DeliverTo>

XML has a structure similar to HTML (HyperText Markup Language), which is an implementation of SGML, XML's parent meta language. But, applications can extract and use data from XML documents more easily than from HTML ones, because in XML, all data is tagged according to its purpose. XML contains only data, while HTML contains both data and presentation information.

Defining interfaces in terms of XML documents also allows for an incremental path to business automation, whereby browser-based tasks are gradually transferred to computer processes. A supplier with a small product catalog and a few sales a day can use a web browser to receive orders and send acknowledgments until increased transaction volume justifies integration with ERP or database applications. Likewise, a buyer who buys only a few items "off the shelf" can rely on a browser to send orders and receive acknowledgments, and only integrate with purchasing or accounting systems when scale justifies it. In each case, since the same XML documents are going in and out, the changes to the implementation are invisible to the marketplace and other trading partners.

2.2.4 xCBL

One of the oldest attempts to solve the problem of interoperability among vertical XML commerce applications is Commerce One's Common Business Library [CBL]. CBL proposes a set of reusable components that are common to many business domains, along with a set of document frameworks for creating documents with a common architecture.

The goal of xCBL (XML Common Business Library) is the development of reusable components which speed the implementation of standards and facilitate their interoperation by providing a common framework. Characteristics of xCBL: [CBL]:

- XML component library for B2B commerce, developed and modeled after EDI semantics such as X12 and EDIFACT.
- Available in numerous formats - for example SOX (Schema Language for Object-Oriented XML), XDR (XML Data-Reduced), etc.)
- Avoid the limitations of DTD (add strong typing, inheritance, global namespaces, etc.)
- Mappings (for certain documents) to ANSI X12 and UN/EDIFACT exists

2.2.5 cXML

Commerce cXML (commerce eXtensible Markup Language) [cXMLUG] is a streamlined protocol intended for consistent communication / transaction of business documents. It is based upon catalogs and purchase orders. The goal is to integrate small and medium sized companies into procurement processes.

cXML is an open language for the transaction requirements of:

- Network e-commerce hubs
- Electronic product catalogs
- PunchOut catalogs
- Procurement applications
- Buying communities
- E-commerce service providers

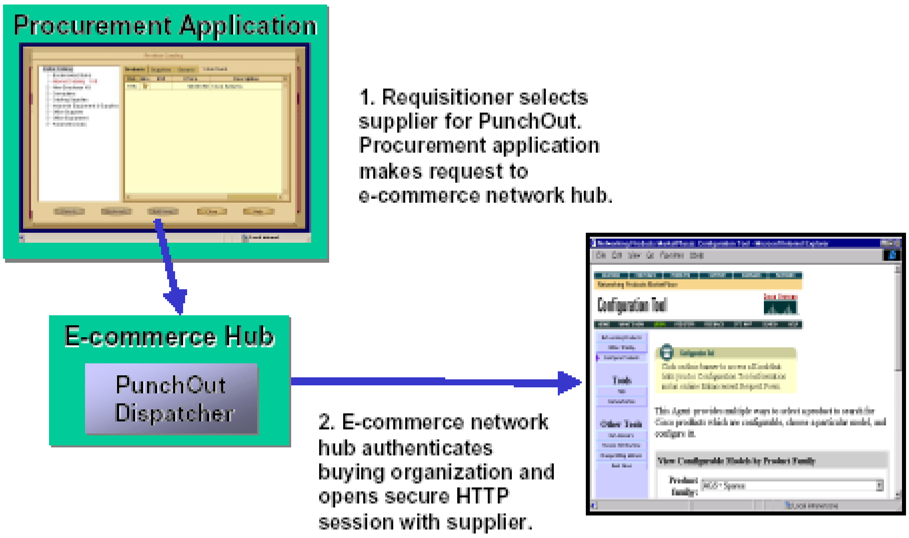
Each cXML document is constructed based XML Document Type Definitions (DTDs). Acting as templates, DTDs define the content model of a cXML document, for example, the valid order and nesting of elements, and the data types of attributes. The DTDs for cXML are files available on the www.cXML.org Website.

cXML transactions consist of *documents*, which are simple text files containing values enclosed by predefined tags. Most types of cXML documents are analogous to hardcopy documents traditionally used in business.

Example for a **PunchOut** scenario [cXMLUG]:

Steps 1 & 2: PunchOut Request

Users log in to a procurement application and open new purchase requisitions. They find desired items by searching their local catalogs by commodity, supplier, or product description. When they select a PunchOut item, the procurement application opens a new browser window and logs them into their accounts at the supplier's Website (see Figure 5 [cXMLUG]).



1. Requisitioner selects supplier for PunchOut. Procurement application makes request to e-commerce network hub.

2. E-commerce network hub authenticates buying organization and opens secure HTTP session with supplier.

Figure 5 PunchOut steps 1 and 2

How does it work? When a user clicks a PunchOut item, the procurement application sends a cXML PunchOutSetupRequest document to a network e-commerce hub. Acting as the trusted third party, the hub accepts the request, verifies the buying organization, and passes the request to the supplier's PunchOut Website.

Step 3: Product Selection

Users select items from the supplier's inventory using all the features and services provided by the supplier's Website (see Figure 6 [cXMLUG]).

Configuration Tool

This Agent provides multiple ways to select a product to search for Cisco products which are configurable, choose a particular model, and configure it.

View Configurable Models by Product Family

Product family: AS5 - Spares

Figure 6 PunchOut step 3

How does it work? After the procurement application directs users to the supplier’s Website, the shopping experience is the same as if they had logged on to the supplier’s Website directly. Thus, none of the previously listed features and services require modification.

Step 4: Check Out

The supplier’s Website calculates the total cost of the user’s selections, including tax, freight, and customer-specific discounts. Users then click the supplier’s Website’s “Check Out” button to send the contents of the shopping cart to the their purchase requisitions within the procurement application.

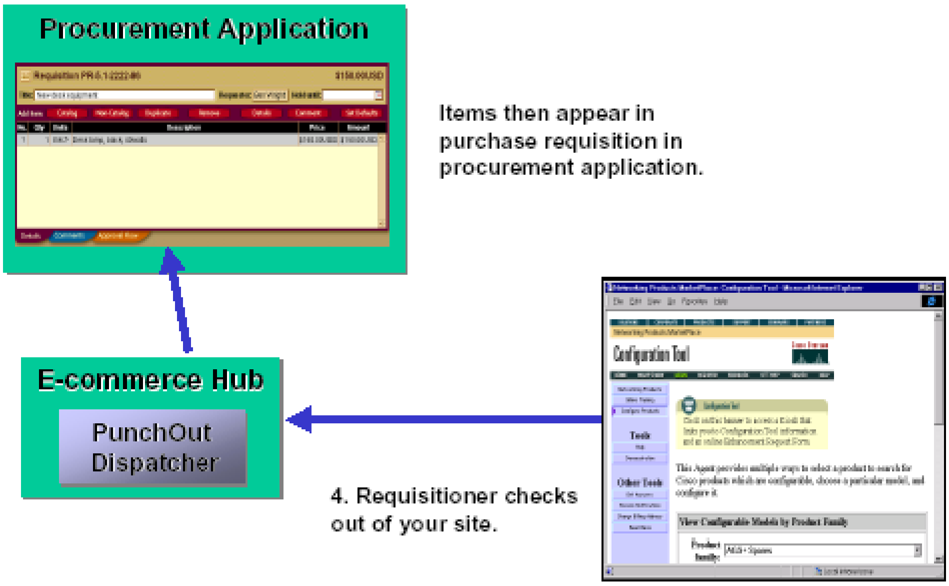
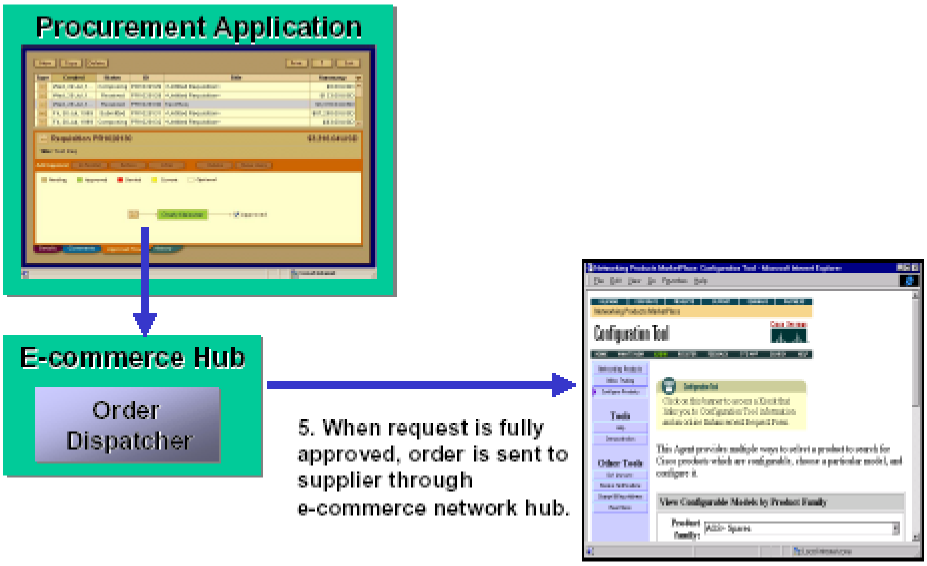

Procurement Application
Items then appear in purchase requisition in procurement application.
E-commerce Hub
PunchOut Dispatcher
4. Requisitioner checks out of your site.

Figure 7 PunchOut step 4 [cXMLUG]

How does it work? When users click the supplier’s “Check Out” button, they submit an HTML form back to their procurement application. One form field consists of a cXML PunchOutOrderMessage containing product details and prices. The supplier can also send hidden supplier cookies, which can later associate items with a specific shopping session.

Step 5: Transmittal of Purchase Order

After the contents of the shopping cart have been passed from the supplier’s Website to the user's purchase requisition, the procurement application approval processes take over. When the purchase requisition is approved, the procurement application converts it into a purchase order and sends it back to the supplier’s Website for fulfillment. Purchasing card data can be transmitted along with the order, or the supplier can invoice the order separately.



5. When request is fully approved, order is sent to supplier through e-commerce network hub.

Figure 8 PunchOut step 5

How does it work? The procurement application sends all purchase orders to the e-commerce hub in cXML format. The hub then routes them to the supplier, using the supplier's preferred order-routing method. When the supplier acknowledges the receipt of a purchase order, the supplier has effectively booked the order.

Specification: cXML Version 1.2.008 (http://cxml.org/)

2.2.6 OAGIS

The OAGI Integration Specification (OAGIS[4]) includes a broad set of Business Object Documents (BODs) and integration scenarios that can be used in different business environments, such as A2A and B2B. A BOD uses meta data to describe itself to other software components. It is itself not an object. It is an application architecture that is used to convey the communication and the necessary data to fulfill the carry out the requested business event.

BODs are message definitions that can be used broadly across many different industries (for example, telecommunications and automotive) and aspects of Supply Chain Automation (for example, Ordering, Catalog Exchange, Quotes, etc.). OAGI also defines the OAMAS (Open Application Middleware API Specification), which is an application programming interface (API) for application integration that provides an abstraction from specific vendor solutions.

2.2.7 RNIF

The RosettaNet Implementation Framework (RNIF) [RNIF] defines the overall RosettaNet business message format for exchange of the business documents, with elements to support

[4] http://www.openapplications.org/global/intro.htm

authentication, authorization, encryption and non-repudiation; details of the bindings for the transfer protocols (e.g. HTTP); and the specification for a reliable exchange of messages between partners.

Purpose of RNIF: The Partner Interface Processes (PIP) specifications define the document exchange choreography and the XML schemas for the individual business documents involved. The format of these schemas varies on per PIP and the specific business document type basis, based on the underlying business purpose that the document serves. Hence it is necessary to define an overall envelope/container format that stays constant and consistent for all exchanges within which all business documents are exchanged as payload. RNIF specifies such an envelope format that is also independent of the specific transfer protocol used to transmit the message between partner nodes.

It is necessary to capture the context information for the specific step in PIP process that the payload business document executes and the attributes of the payload document(s), in a PIP and business document independent way.

RNIF specifies:

- the XML schema for a header document called the Service-Header for this purpose, an instance of which must always precede a business document instance, in a RosettaNet message.

- and provides for a consistent mechanism to digitally sign and or encrypt all RosettaNet messages (as needed), independent of the transfer protocol, PIP and the specific business document being exchanged.

- a reliable messaging mechanism based on Acknowledgements and supplies a set of standard choreography models that all PIPs must follow.

- the schemas for the Acknowledgement and Exception (error) messages.

- the transfer protocol level bindings for all the supported transports so that RosettaNet messages are exchanged in a consistent and interoperable way.

Specification available at: http://www.rosettanet.org/standards

2.2.8 SWIFT

The Society for Worldwide Interbank Financial Telecommunication (SWIFT[5]) was founded about 30 years ago with the goal to provide message exchange between financial institutions. It is a company owned by enterprises in the financial industry. SWIFT provides not only business document adventitiousness for the financial world but also provides the necessary network and software infrastructure for participants to exchange messages. SWIFT has developed a methodology for developing standards called SWIFT Standards Modeling and it has three layers.

[5] http://www.swift.com/

- The Business Layer describes the particular business independent of any technology support.

- The Logic Layer describes the necessary business data exchange and

- the Physical Layer provides the particular syntax necessary.

Since SWIFT started that long ago initially a non-XML syntax was used and deployed for quite some time in order to transfer financial business data. Recently SWIFT started acknowledging XML as an alternative syntax by proposing swiftml as XML-based definition of financial business data.

Additionally the Fix Protocol, Ltd. (FPL), a company, developed a public domain protocol called Financial Information Exchange Protocol (FIX[6]) targeted for the real-time exchange of securities transactions. SWIFT and FPL agreed to join efforts and to converge their protocols [B2BBus1].

2.2.9 EbXML

See 2.3.1.

2.2.10 RosettaNet

RosettaNet defines not only Partner Interface Processes (see 2.3.2), but also specific business document types for particular business data like purchase orders or invoices. The definitions are accomplished using XML. Two distinct business documents are defined for communication management. One is the Receipt Acknowledgment to acknowledge messages and one is the Exception to indicate error situations.

In addition, RosettaNet provides several dictionaries that define the valid content of the business data in the business documents. These are the Business Dictionary defining business data and entities, IT Dictionary defining IT products and properties and the EC Dictionary defining components and their properties. Validation rules are specified, too, that make use of the dictionary data in order to establish the correctness of a transmitted RosettaNet document. RosettaNet supports the D&B D-U-N-S Number[7], GTIN[8] (Global Trade Item Number) as well as UN/SPSC (see 2.1.2.2) codification standards. In addition to the business documents RosettaNet defines the message structure for sending business documents to trading partners. This structure contains three headers called Preamble, Delivery and Service Header and is followed by the payload and any number of attachments. The payload is the area where a business document is located [B2BBus1].

[6] http://www.fixprotocol.org/cgi-bin/Welcome.cgi

[7] http://www.dnb.com/us/

[8] http://www.uc-council.org/2005sunrise/global_trade_item_number.html

2.2.11 UBL

In the field of ongoing standardization efforts the following problem arises: A lot of XML business-to-business document standards are under development. The Universal Business Language (UBL) [UBL] has the goal to abolish this diversity.

Thus the question arises: why not use XML? XML is often described as the *lingua franca* of e-commerce. The implication is that by standardizing on XML, enterprises will be able to trade with anyone, any time, without the need for the costly custom integration work that has been necessary in the past. But this vision of XML-based “plug-and-play” commerce is overly simplistic. Of course XML can be used to create electronic catalogs, purchase orders, invoices, shipping notices, and the other documents needed to conduct business. But XML by itself doesn't guarantee that these documents can be understood by any business other than the one that creates them.

XML is only the foundation on which additional standards can be defined to achieve the goal of true interoperability. The UBL initiative is the next step in achieving this goal. The task of creating a universal XML business language is a challenging one. Most large enterprises have already invested significant time and money in an e-business infrastructure and are reluctant to change the way they conduct electronic business. Furthermore, every company has different requirements for the information exchanged in a specific business process, such as procurement or supply-chain optimization. A standard business language must strike a difficult balance, adapting to the specific needs of a given company while remaining general enough to let different companies in different industries communicate with each other.

The UBL effort addresses this problem by building on the work of the ebXML initiative. ebXML is a joint project of UN/CEFACT, the world body responsible for international Electronic Data Interchange (EDI), and the Organization for the Advancement of Structured Information Standards (OASIS), a nonprofit consortium dedicated to the open development of XML languages. UBL is organized as an OASIS Technical Committee to guarantee a open process for the standardization of the XML business language.

Thus UBL choose as a starting point an existing XML business document library as a basis for creating new Universal Business Language that will be a synthesis of existing XML business document libraries.

As currently envisioned, the UBL work will take place in two phases

1. Align the vocabulary and structures of UBL with the work of other initiatives (already existing business libraries such as RosettaNet and OAGIS[9])

2. Implement a mechanism for the generation of context-specific schemas through the application of transformation rules to a common XML source library

[9] Open Applications Group Interoperability Standard

2.2.12 Concluding remark

There are many standards for document and data exchange developed for special needs in e.g. the industrial or financial sector. The more recent ones are based on XML. They are mostly very specific to the area in which they have been developed and lack flexibility and semantic aspects of document description. Within SWWS we aim at capturing the semantics of content in order to allow dynamic discovery and invocation of web services. Here such standards are too tightly bound to their underlying domain to be used directly to solve the problems. We plan to achieve a more dynamic description mechanism by exploiting ontology technology to take semantics of arbitrary domains into account.

2.3 Collaboration

A collaboration describes how two concurrent executable business processes interact at the business level. EbXML (see 2.3.1) as well as RosettaNet (see 2.3.2) are two important specifications in this area of standardization efforts.

2.3.1 EbXML

One of the most important efforts to create a common framework for e-business integration is ebXML. The Electronic Business XML Initiative (sponsored by the UN/CEFACT[10] and OASIS [11]) is a worldwide project to standardize the exchange of electronic business data. EbXML is supported by hundreds of industry consortia, standards bodies and companies from around the world. [ebXML1]

The goal of EbXML is to establish a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet. Each specification is designed to be implementable independent of other specification, though appropriate mappings and hooks are provided to support efficient integration of components built using other ebXML specifications.

The goal is to build a **registry** that includes more detailed, business process-specific information about services. Using ebXML, companies now have a standard method to exchange business messages, conduct trading relationships, communicate data in common terms and define and register business processes. For instance if a service is part of a workflow application, ebXML lets you specify how other elements in the workflow should interact with it, through the use of Collaboration Protocol Profiles (CPP) and Collaboration Protocol Agreements (CPA).

The ebXML technical architecture makes use of existing standards wherever possible, building on the experience of EDI while taking advantage of the increased flexibility of XML and ubiquity of the Internet. Because the architecture is modular, industries or companies can choose to implement parts of the ebXML technology rather than trying to do everything all at once. [OAebXML]

[10] United Nations Centre for Trade Facilitation and Electronic Business

[11] Organization for the Advancement of Structured Information Standards

On the surface, using an ebXML system is straightforward (see Figure 9 [OAebXML]). It involves the following steps:

- Search for a Trading Partner (in the ebXML Registry).
- Create a CPA.
- Negotiate any issues regarding the CPA.
- Configure both *Business System Interfaces* using the CPA.
- Begin performing Business Processes.

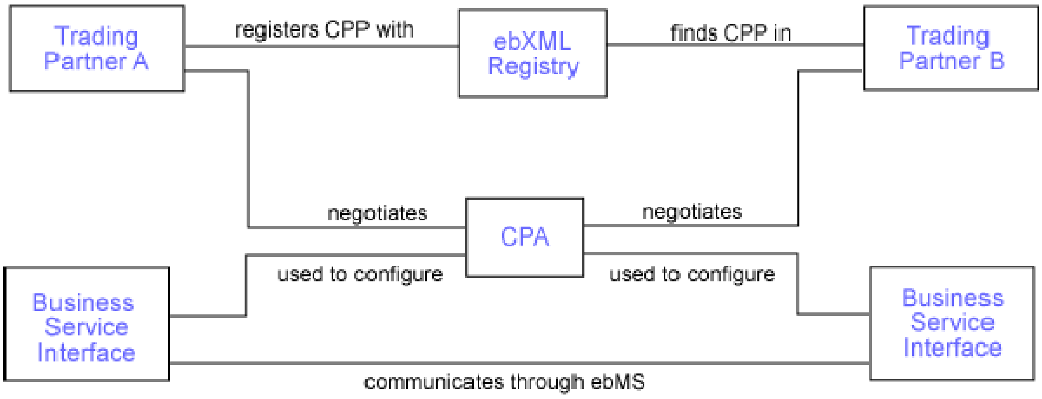


Figure 9 ebXML System

EbXML recognizes that integration is a complex problem that requires standardization in a number of distinct areas:

- **Messaging Services**
  Standard protocols like TCP/IP and HTTP are too low-level to serve the needs of electronic business. ebXML messaging addresses this problem by extending the SOAP protocol to add features needed for the exchange of business documents: security, authentication, and non-repudiation.

  Specification: Message based service invocation (ebMS)
  Link: http://www.oasis-open.org/committees/ebxml-msg/

  This specification focuses on defining a communications-protocol neutral method for exchanging electronic business messages. It defines specific enveloping constructs supporting reliable, secure delivery of business information. Furthermore, the specification defines a flexible enveloping technique, permitting messages to contain payloads of any format type. This versatility ensures legacy electronic business systems employing traditional syntaxes (i.e. UN/EDIFACT, ASC X12, or HL7) can leverage the advantages of the ebXML infrastructure along with users of emerging technologies.

- **Registry and Repository**
  ebXML also specifies a standard protocol for accessing central registries and repositories of business data. These data can include such things as trading partner profiles and business document formats.

  Specification: Registry Services Specification v2.0
  Link: http://www.oasis-open.org/committees/regrep/

  Role of ebXML Registry
  Registry is a registry ("catalog") as well as a repository ("warehouse"). Interfaces to manage the lifecycle of Registry entries and to support queries on Registry entries are provided. The *Registry* provides a stable store where information submitted by a *Submitting Organization* is made persistent. Such information is used to facilitate ebXML-based Business to Business (B2B) partnerships and transactions. Submitted content may be *XML* schema and documents, process descriptions, ebXML *Core Components*,context descriptions, *UML* models, information about parties and even software components.

- **Collaboration Partner Profile and Collaboration Partner Agreement**
  A CPP provides the information needed to do business with a specific trading partner, such as the business processes and document formats that it uses. When two parties trade for the first time, their CPPs are combined into a CPA that serves as the basis for their interaction.

  Specification: ebCPPA:
  Link: http://www.oasis-open.org/committees/ebxml-cppa/

  A CPPA specifies XML Schemas for CPP and CPA, and also guidelines to form a CPA from two CPPs. The CPP contains elements that specify Roles (e.g., Seller, Buyer), Services, Actions, and message attributes (e.g., number of retries, time out interval, and so on for reliable messaging, certificates for trust management).

  Trading-Partner representation with ebXML
  Standardizing on a specification for the electronic trading partner agreement (TPA) is essential to widespread e-commerce. TPAs capture critical information upon which organizations must agree in order for their applications and business processes to communicate. TPA will be a key element for interoperability among B2B server implementations.

  An electronic TPA can be defined as an XML document that records specific technology parameters for conducting electronic business. Partner identification, communications protocol, security for message exchanges (including encryption, authentication, and non-repudiation), definition of requests and responses are all part of a typical TPA.

  Much of the work is based on previously proposed OASIS technical work surrounding tpaML (the Trading Partners Agreement Markup Language). tpaML was originally developed by IBM.

A CPP (Collaboration-Protocol Profile) defines one business partner's technical capabilities to engage in electronic business collaborations with other partners by exchanging electronic messages. A CPA (Collaboration-Protocol Agreement) documents the technical agreement between two (or more) partners to engage in electronic business collaboration.

A CPP includes for example:

- Party’s information (contact name, contact info, etc.)
- Transport Protocol
- Transport Security Protocol
- Messaging Protocol
- Link to Process-Specification document
- Time out / Retry
- etc.

A CPA can be seen as an intersection of two or more CPPs.

Examples can be found at:

- CPP: http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-example-ompanyA-2_0b.xml
- CPA: http://www.oasis-open.org/committees/ebxml-cppa/schema/cpa-example-2_0b.xml

- **Business Processes and Core Components [ebCC]**
ebXML aims to create a generic metamodel for business processes with which any business process can be modeled in a machine-readable way. Eventually, this will enable companies to deploy software that automatically adapts to the specific business processes of its trading partners.

ebBPSS is used to specify the externally visible ("public") business process between Party A and Party B. It provides an XML Schema to specify Binary Collaboration between Party A and Party B. A Binary Collaboration may consist of multiple Business Transactions. Each Business Transaction is specified in terms of Business Envelopes, Business Documents, and Business Signals that are communicated between Party A and Party B.

An example can be found at:
http://www.oasis-open.org/committees/ebxml-cppa/schema/bpss-example-2_0a.xml

Finally, ebXML is compiling a set of common business document components for basic business information such as addresses, products, trading parties, and the like.

A **core component** used in a particular business context is called a *business information entity* (BIE). BIEs can be assembled into business document forms (purchase orders, invoices, etc.), and these forms, when populated with data, become interoperable business documents.

Both *Business Processes* and *Business Documents* are designed and documented prior to their use, and are usually composed from existing components and processes. For example, Business Processes may be composed from existing Core Processes documented in a business library or other registry. Business Documents are normally composed from existing Core Components in a registry. Both are documented using the *Business Process Specification Schema* (BPSS) and stored in an ebXML registry so that they can be referenced from CPPs, CPAs, and other structures.

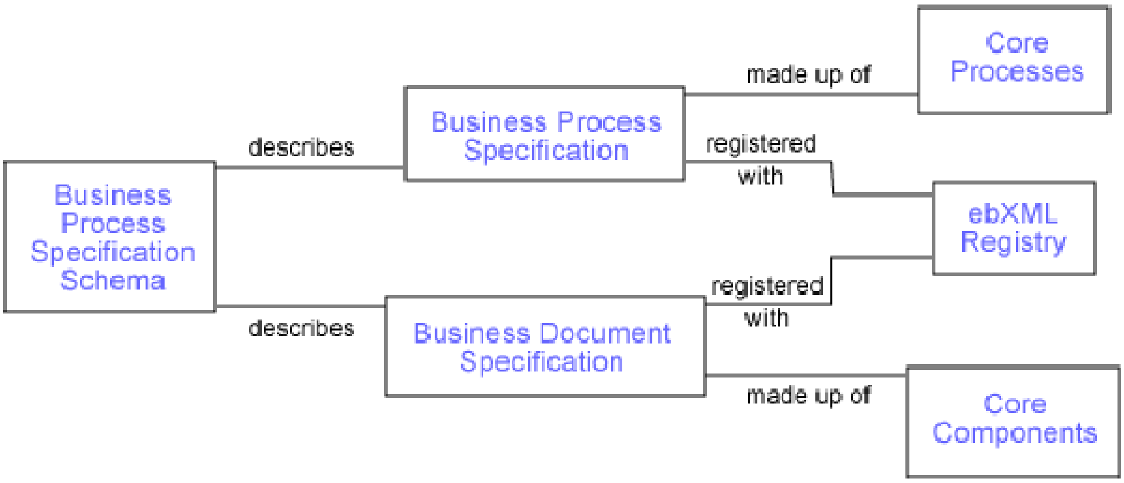


Figure 10: ebXML System II

The Business Process Specification Schema

Business models define how business processes are discovered, defined, and documented. In ebXML, you can accomplish this by using the *Unified Modeling Methodology* (UMM). UMM is not required. Just as an XML Schema or DTD provides a vocabulary for data within an XML document, the UMM provides a common language that can be used by those individuals who define business processes. The BPSS is a subset of the UMM. The BPSS is typically expressed in the *Unified Modeling Language* (UML) and translated to a XML Schema or DTD using production rules. In this way, the common language that the UMM uses to think about and discuss business processes becomes a common language through which you can describe processes using XML.

The BPSS is used to define both the *Business Processes* and the *Business Documents* they involve.

Architectural Decisions

- Modular specifications: each specification can be independent of another to facilitate easy adoption

- The operations described in the "Concept of Operation" earlier are divided into three phase: Implementation, Discovery, and Run-time. A CPA formed during the discovery phase is not changed during the execution of business transactions in run-time phase.

- Mappings among specifications: Whenever an ebXML specification can use a component built to another ebXML specification, the necessary mappings between the specifications are specified.

- Evolve the current state of the art instead of impose a new infrastructure - In B2B world EDI is still used heavily, and the best practices of such usage is used in the design of ebXML.

- Never reinvent the wheel - use other specifications (use of SOAP 1.1 and XMLDSIG in ebMS, for example) whenever available and appropriate.

The ebXML infrastructure specifications – Messaging services, Registry/Repository, and CPP/CPA – are now maintained by OASIS technical committees, while Core Component discovery and Business Process modeling continue under the aegis of UN/CEFACT.

Specifications are available at: http://www.oasis-open.org/

2.3.2 RosettaNet

RosettaNet [RNhp] is a self-funded, non-profit organization (since 1998). It is a consortium of major Information Technology, Electronic Components and Semiconductor Manufacturing companies working to create and implement industry- wide, open e-business process standards. These standards form a common e-business language, aligning processes between supply chain partners on a global basis.

RosettaNet aims to align the business processes of supply chain partners. This goal is achieved by the creation of Partner Interface Processes or PIPs [PIPTA]. Each PIP defines how two specific processes, running in two different partners organisations, will be standardised and interfaced across the entire supply chain. PIP includes all business logic, message flow, and message contents to enable alignment of the two processes. The standardization efforts concern the following aspects:

- PIPs
  RosettaNet Partner Interface Processes® (PIPs®) define business processes between trading partners.

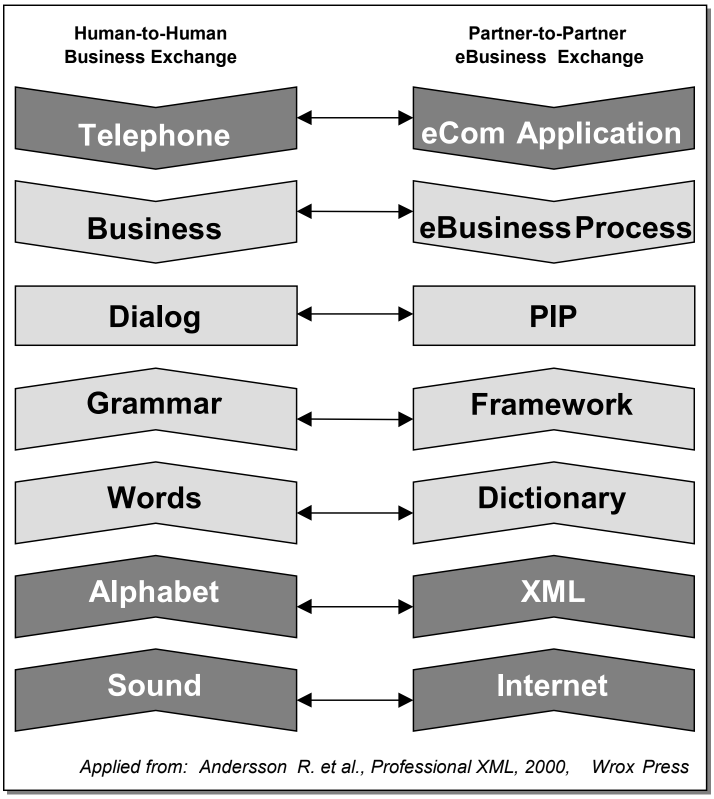
- PIP Directory
  The PIP® Directory provides faster access to the PIP information.

- Dictionaries
  RosettaNet dictionaries provide a common set of properties for PIPs®. The RosettaNet Business Dictionary designates the properties used in basic business activities. RosettaNet Technical Dictionaries provide properties for defining products.

- RosettaNet Implementation Framework
  The RosettaNet Implementation Framework (RNIF) Core Specification is the packaging, routing, and transport of all PIP® messages and business signals.

- Product & Partner Codes
  Product and partner codes in RosettaNet standards expedite the alignment of business processes between trading partners.

An analogy (see Figure 11) illustrates the purpose of the RosettaNet initiative. The fundamental system of exchanging sounds in a *human-to-human* business exchange can be compared to the Internet, which enables two servers to exchange information during a *server-to-server* electronic business exchange. HTML/XML functions as the 'alphabet' of this electronic exchange. And, presently ECOM applications serve as the instrument by which an electronic business process is transmitted.



Applied from: Andersson R. et al., Professional XML, 2000, Wrox Press

Figure 11 RosettaNet compared to a traditional dialog [RNhp]

What is further needed in order to scale eBusiness are the 'dictionaries,' the 'framework,' the 'Partner Interface Processes - PIPs' and the 'eBusiness processes.' RosettaNet fills this existing gap by focusing on building a master dictionary to define properties for products, partners, and business transactions. This master dictionary, coupled with an established implementation framework (exchange protocols), is used to support the eBusiness dialog known as the Partner Interface Process or PIP.

The purpose of each PIP is to provide common business/data models and documents enabling system developers to implement RosettaNet eBusiness interfaces.

Each includes

- Partner Role Descriptions (individuals / organizations),
- Business Data involved (and corresponding XML document(s) based on Implementation Framework DTDs, specifying PIP Service(s), Transactions, and Messages which include dictionary Properties)
- Business Process Activities (incl. Functional Process flow-chart)
- A validation tool and Implementation guide

RosettaNet has delineated the scope of supply chain processes for which it will design PIPs. This scope is divided into a total of 17 segments grouped in 6 clusters. The clusters and segments serve as a mechanism to group all supply chain processes into a manageable framework. When the PIPs are implemented, they should be selected from all segments to form a subset of PIPs required to address specific business interface scenarios.

The segments arranged in the corresponding clusters:

Cluster 1: Partner, Product and Service Review
Segment 1A: Partner Review
Segment 1B: Product & Service Review
Cluster 2: Product Introduction
Segment 2A: Preparation for Distribution
Segment 2B: Product Change Notification
Cluster 3: Order Management
Segment 3A: Quote and Order Entry
Segment 3B: Transportation and Distribution
Segment 3C: Returns and Finance
Segment 3D: Product Configuration
Cluster 4: Inventory Management
Segment 4A: Collaborative Forecasting
Segment 4B: Inventory Allocation and Replenishment
Segment 4C: Inventory Reporting
Segment 4D: Inventory Replenishment
Segment 4E: Sales Reporting

Segment 4F: Price Protection
Cluster 5: Marketing Information Management
Segment 5A: Lead/Opportunity Management
Segment 5B: Marketing Campaign Management
Segment 5C: Design Win Management (Electronic Components)
Segment 5D: Ship From Stock and Debit (Electronic Components)
Cluster 6: Service and Support
Segment 6A: Provide and Administer Warranties, Service Packages, and Contract Services
Segment 6B: Provide and Administer Asset Management
Segment 6C: Technical Support & Service Management

Process Model

RosettaNet does not provide a model for supply chain arrangements as a whole. What it does provide is a model for linking supply chain members' information flows in a uniform manner, within specific business processes.

The model describes several business activities that can be mapped to the RosettaNet XML-framework. The activities are collected inside processes, called Partner Information Processes, or short PIPs. The PIPs are based on RosettaNet Dictionaries and are enabled by the RosettaNet Implementation Framework (RNIF). These dependencies are illustrated in Figure 12 below.

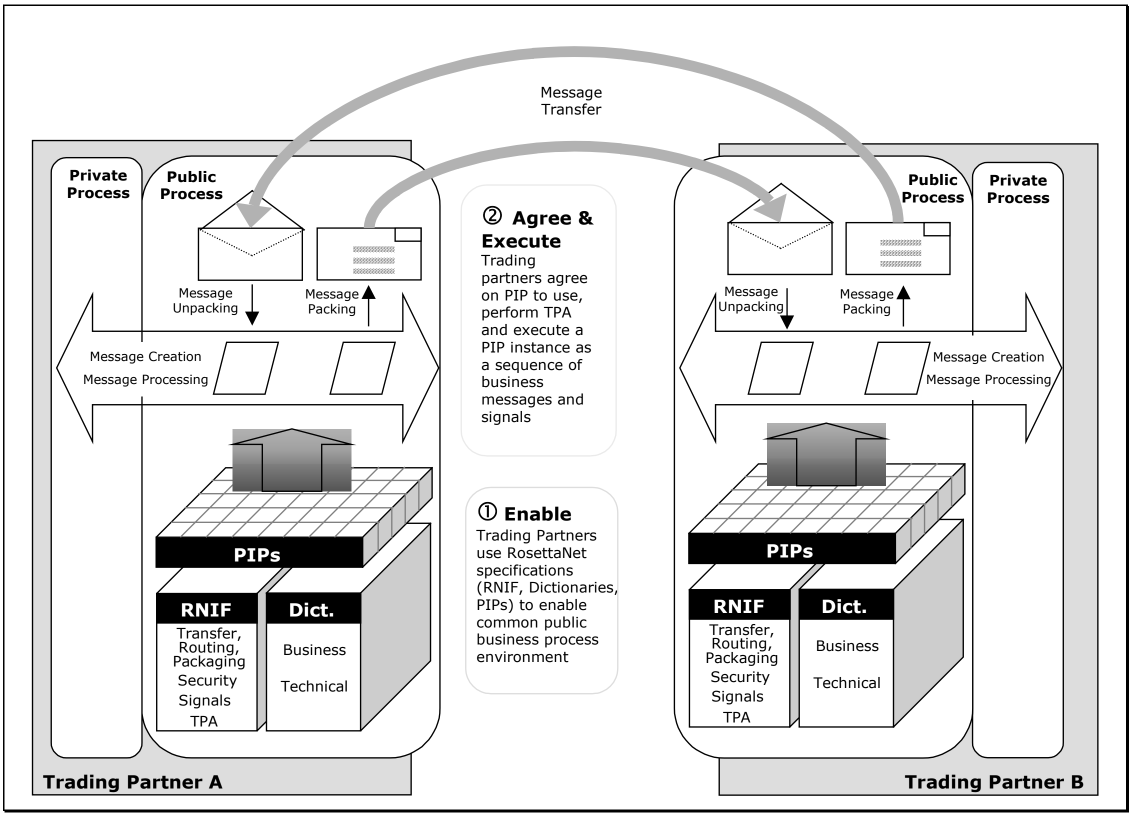


Figure 12 RosettaNet messaging [RNhp]

An organization’s business processes are divided into two categories in the RosettaNet model. The business processes that are internal to the organization are called “private processes,” while the business processes that involve interactions with trading partners are known as “public processes.”

The public processes are business processes through which partners conduct e-business. Within the context of RosettaNet, these are the partner interface processes that are visible between trading partners. Public processes implement the RosettaNet PIP specifications to exchange standard business documents over standard Internet transfer protocols, as specified by the RosettaNet Implementation Framework.

Within trading partner enterprises, private processes interface with public processes and with back-end business systems are needed to facilitate e-business exchanges between trading partner organizations, as illustrated Figure 13 below.

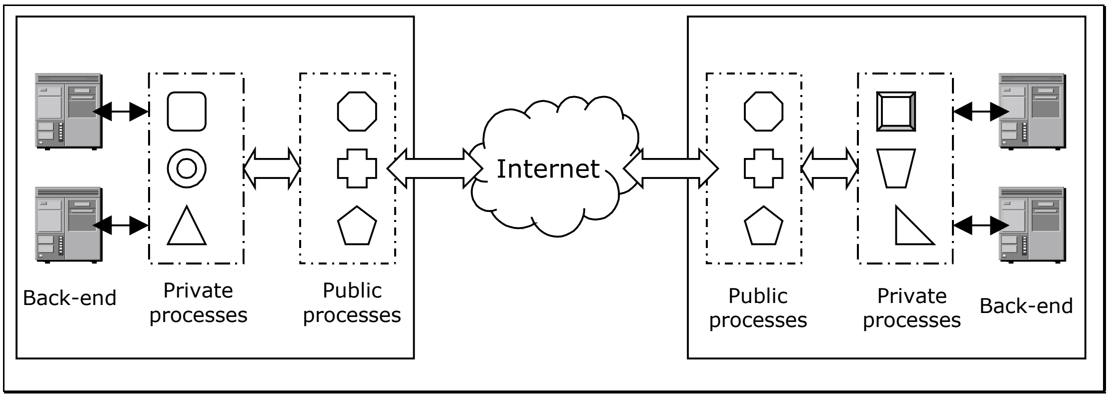


Figure 13 Process relations in RosettaNet

The elements needed on behalf of the public processes are declared in each of the PIP specifications. Source: [RNIF_R2].

The PIP Architecture

The RosettaNet PIP architecture comprises two fundamental parts.

1. **A business process model.**

   This model captures business roles and their interactive functional activities, the information that is exchanged when performing these interactive activities, and the sequence in which these interactions take place.

   The business process model specifies a generic "to-be" partner interface business process with the following purpose, viewpoint and context.

   - *Purpose.* The purpose of the model is to specify a generic channel-centric (not organization-centric) business process between roles performing interactive functional activities in the information technology distribution channel. This model is used to reach consensus on a common business practice. The model additionally acts as functional requirements for the distributed information system design.
   - *Viewpoint*. The model's viewpoint is that of an industry consortium specifying generic business process guidelines that can be adopted by partner companies that interface in the information technology distribution channel.
   - *Context*. The context of the model is determined by the scope of a PIP. The broad scope is that of business interface activities performed by partners in the information technology distribution channel.

   There are two presentation methods used to communicate the business process model.

- The first is a presentation that shows the upstream, downstream and intermediate "glue" activities that are preformed in a particular partner interface process. This provides a contextual understanding of the business process.

- The second is a presentation that shows just the role interactions that occur at the partner interfaces of the business process. These presentations do not show the "glue" activities in the business process. The presentation also serves to specify the interaction sequence between the roles.

An information model that specifies the information exchanged by the roles during each interaction accompanies the process model.

2. **A distributed information system design.**

This design specifies the agent and service software components, together with their information exchange and message protocols, that can either replace or support the roles in the business process model.

The distributed information system design specifies agents and service software components that inter-operate using an agreed message and transaction sequence protocol. An agent performs some unit of work on behalf of a role. Agents are often implemented as clients (e.g. a web browser is a user agent) but not necessarily so. Services are often implemented as servers or just behind servers (e.g. behind an HTTP server).

There are two parts to the protocol design.

- The message exchange specification. Information exchanged during a system transaction is specified as properties arranged into related containers and messages. One message is exchanged for each system interaction. Property specifications are obtained from RosettaNet's property dictionaries that are described in the following section.

- The service transaction specification. A transaction defines the boundaries (start, end) of a service commitment. A transaction starts with an initial message exchange and comprises any number of intermediate exchanges until the final message exchange. If an error occurs at any time during the transaction then the service must role back to its initial state. If no error occurs then the service can reliably commit the transaction's message actions. The transaction boundaries and the sequence of message exchange are specified for each of the role interactions in the business process.

Both the business process model and the information system design are expressed using formal visual languages.

2.4 Business processes

Everything that looks like a series of steps tends to be labeled a business process. Here is a simple **taxonomy of "business processes"** [BusPr1]. For the purpose of this deliverable we will distinguish five concepts, all of which are referenced in the literature as "business processes":

- Enterprise business processes
- Executable business processes
- ebXML business processes (a.k.a. collaborations)
- Business process activities
- Workflows

An **Enterprise Business Process i**s the description of steps needed to carry out a business activity regardless of the systems involved. They provide a high level view of the steps involved and can be used to model, benchmark and document existing or future designs. Enterprise business processes are actually free to span multiple corporations because of their nature which is not bounded to systems. An example would be describing all the steps that are required to happen for a pair of shoes to be manufactured in Asia and appear at your favourite store at the mall.

An **Executable Business Process** is a kind of Enterprise Business Process whose lifecycle is controlled by one or a combination of systems. We will call these systems: business process management systems (**BPMS**). It is limited to run within a single corporation. One of the important characteristics of an Enterprise Business Process or Executable Business Process is that it is long running. Its execution is not limited to minutes or hours like the session of a web-based application, it rather spans days, months, or years. An Executable Business Process relies on specific interactions between users, systems, and business partners which it ties together. This system provides all the facilities and services necessary for design and execution, and mediates the integration with its environment. As we will see in the later paragraphs of this section, a BPML, XLANG or WSFL business process is an Executable Business Process.

An **ebXML Business Process** (Collaboration) is a business collaboration specification which can be used to specify how two concurrent executable business processes interact at the business level.

A **Business Process Activity** (Task) represents a short-lived interaction between users or, in certain cases, systems. A Business Process Activity can be viewed as one step in an executable business process. A typical example is a user browsing a catalogue and filling a shopping cart. Once the user is finished, he or she pushes the checkout button, which in turn completes the activity. The proper information is passed to a business process management system as part of a completion message

We can often associate **workflow** to "automated document management" which requires reviews and approvals: for instance the review of a proposal or a contract by a large number of people. The engine in charge of this task does not know much about the documents themselves and is merely routing them through different people while keeping an audit trail. There is little or no integration with enterprise systems, let alone with other partners.

A **Workflow Management System (WFMS)** is a system that completely defines, manages and executes „workflows“ through the execution of software whose order of execution is driven by a computer representation of the workflow logic.

2.4.1 BPML / BPQL

The Business Process Management Initiative (BPMI.org) has announced the release of the final draft for the Business Process Modelling Language (BPML 1.0) and the first public working draft for the Business Process Modelling Notation (BPMN).

The Business Process Modelling Language BPML specification provides an abstract model and XML syntax for expressing executable business processes (including activities of varying complexity, transactions and their compensation, data management, concurrency, exception handling and operational semantics) and supporting entities, based on the concept of a transactional finite-state machine.

The Business Process Modelling Notation (BPMN) specification "provides a graphical notation for expressing business processes in a Business Process Diagram (BPD). The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes. Thus, BPMN creates a standardized bridge for the gap between the process analysis and process implementation. Additionally, the BPMN specification also provides a mapping between the graphics of the notation to underlying the constructs of execution languages, such as BPEL4WS and BPML."

Current version: Business Process Modeling Language (BPML 1.0)
Link: http://www.bpmi.org/

The **Business Process Query Language (BPQL)** [BPQL1] defines a standard interface to forthcoming Business Process Management Systems (BPMS). It allows system administrators to manage the BPMS and business analysts to query the instances of business processes it executes. **Analogy**: SQL established a standardized means of managing business data through DBMSXLANG

The XLANG (XML-based language) language is - similar to the WSFL language - an outdated specification. Hence, in this section only a brief description of XLANG is given because the standard together with the WSFL specification has merged in the new BPEL4WS standard.

XLANG is an XML-based language that supports transactions that may involve multiple Web Services. XLANG is the business process automation language utilized by Microsoft's BizTalk Server. As already mentioned within the introduction the automation of business processes based on Web Services requires a notation for the specification of message about exchange behaviour among participating Web Services. An XLANG service description extends a WSDL service description with an element describing the behavioural aspects of the service.

XLANG serves as the basis for automated protocol engines that can track the state of process instances and help enforce protocol correctness in message flows. XLANG makes a notation for expressing the compensatory actions for any request that needs to be undone. The Web Services infrastructure can leverage XLANG specifications to perform complex undo operations.

2.4.3 WSFL

The Web Services Flow Language (WSFL) is an XML language for the description of Web Services compositions. In this section only a brief description of WSFL is given because the standard together with the XLANG specification has merged in the new BPEL4WS standard.

WSFL considers two types of Web Services compositions:

- The first type specifies the appropriate usage pattern of a collection of Web Services, in such a way that the resulting composition describes how to achieve a particular business goal; typically, the result is a description of a business process.

- The second type specifies the interaction pattern of a collection of Web Services; in this case, the result is a description of the overall partner interactions.

Flow Models

There exists one flow model for each service provider. It defines the invocation sequence of operations of port types. A composition is created by describing how to use the functionality provided by the collection of composed Web Services. This is also known as flow composition, orchestration, or choreography of Web Services. WSFL models these compositions as specifications of the execution sequence of the functionality provided by the composed Web Services. Execution orders are specified by defining the flow of control and data between Web Services. For this reason, in this document, we will also use the term flow model to refer to the first type of Web Services compositions. Flow models can especially be used to model business processes or workflows based on Web Services.

Global Models

A global model relates to operations of all service providers. No specification of an execution sequence is provided. Instead, the composition provides a description of how the composed Web Services interact with each other. The interactions are modeled as links between endpoints of the Web Services' interfaces, each link corresponding to the interaction of one Web Service with an operation of another Web Service's interface. Because of the decentralized or distributed nature of these interactions, we will use the term global model in this document to refer to this type of Web Services composition.

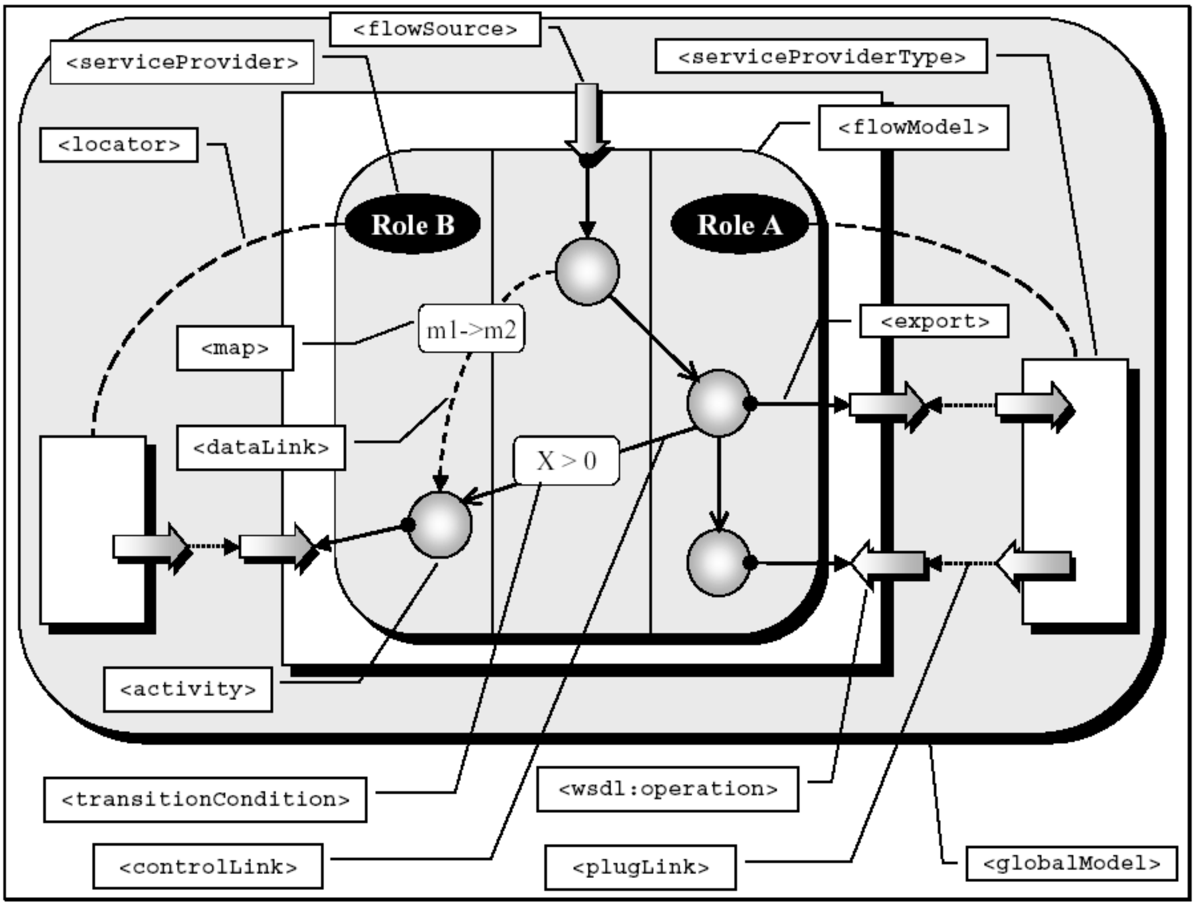


Figure 14 WSFL overview [WSFL]

A WSFL flow model defines the structure of the business process: WSFL activities (circles in the figure above) describe the processing steps, and WSFL data and control links represent the sequencing rules and information flows (eventually performing necessary data mapping) between these activities. For each activity, they would identify the WSFL service provider responsible for the execution of the process step (for example, services offered by shipping company A or by goods-supplier company B) and define the association between activities in the flow model and operations offered by the service provider using WSFL export and plug link elements. The resulting flow model is shown in the center of the figure above with ”swim lanes” representing the association of activities with service provider roles.

Recursive Composition
WSFL provides extensive support for the recursive composition of services: In WSFL, every Web Service composition (a flow model as well as a global model) can itself become a new Web Service, and can thus be used as a component of new compositions. The ability to do recursive composition of Web Services provides scalability to the language and support for top-down progressive refinement design as well as for bottom-up aggregation. For these reasons, recursive composition has been a central requirement in the design of the WSFL language.

Hierarchical and Peer-to-Peer Interaction
WSFL compositions support a broad spectrum of interaction patterns between the partners participating in a business process. In particular, both hierarchical interactions and peer-to-peer interactions between partners are supported. Hierarchical interactions are often found in more stable, long-term relationships between partners, while peer-to-peer interactions reflect relationships that are often established dynamically on a per-instance basis.

Relation to Web Services Stack
The guiding principle behind WSFL is to fit naturally into the Web Services computing stack. It is layered on top of the Web Services Description Language (WSDL – see SWWS deliverable D1.2). WSDL describes the service endpoints where individual business operations can be accessed. WSFL uses WSDL for the description of service interfaces and their protocol bindings. WSFL also relies on an envisioned “endpoint description language” to describe non-operational characteristics of service endpoints, such as quality-of-service properties. Here, we will refer to this language as the “Web Services Endpoint Language (WSEL)” WSEL layers on top of the WSDL to describe the service. Multiple service providers may provide the same service with different features. These are described using WSEL. It defines the characteristics of the web service that are impacted by its implementation environment. This protocol has only been hinted at publicly and needs a LOT of work. ebXML Collaboration-Protocol Profile and Agreement Specification (CPP) is another example of such a description.

2.4.4 BPEL4WS

BPEL4WS (Business Process Execution Language for Web Services) provides a language for the formal specification of business processes and business interaction protocols. As already mentioned within the introduction, Business processes can be described in two ways. Executable business processes model actual behavior of a participant in a business interaction. Business protocols, in contrast, use process descriptions that specify the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal behavior. The process descriptions for business protocols are called abstract processes.
BPEL4WS is meant to be used to model the behavior of **both executable and abstract processes**. By doing so, it extends the Web services interaction model and enables it to support business transactions. BPEL4WS defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and the business-to-business spaces.

As an executable process implementation language, the role of BPEL4WS is to define a new Web service by composing a set of existing services. Thus, BPEL4WS is basically a language to implement such a composition. The interface of the composite service is described as a collection of WSDL portTypes, just like any other Web service. The composition (called the process) indicates how the service interface fits into the overall execution of the composition. Figure 15 illustrates this outer view of a BPEL4WS process.

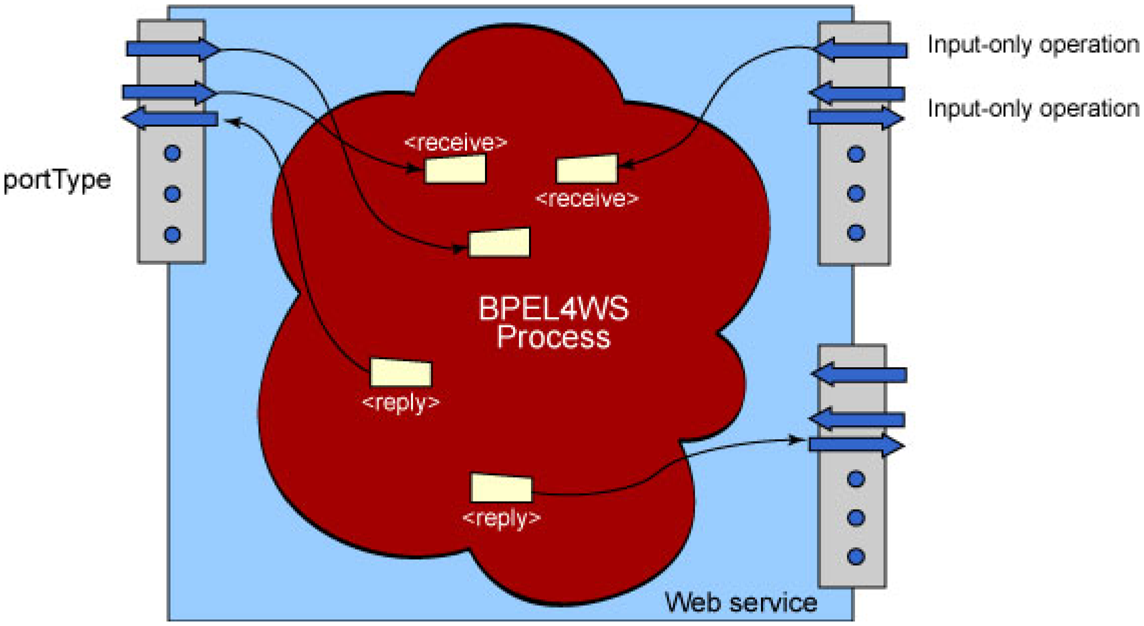


Figure 15 View of a Web service implemented as a BPEL4WS process [BPEL4ws]

Example

In order to demonstrate how activities may be created and aggregated with BPEL4WS, Figure 16 depicts a simple example that processes a credit transfer during a shopping. The graphic shows the external view of a BPEL4WS process to the customer. He only sees the web service of the shop he is buying his goods. He types in his PIN number (<receive>) and gets a receipt (<reply>).

The middle step will involve sending the PIN and further information to a Web services enabled financial institution. From the customer's point of view, the process will consume his request and then send him an answer. As mentioned Figure 16 [BPEL4ws] shows this external view of the request process using the cloud diagram introduced in the BPEL4WS overview (see Figure 15).

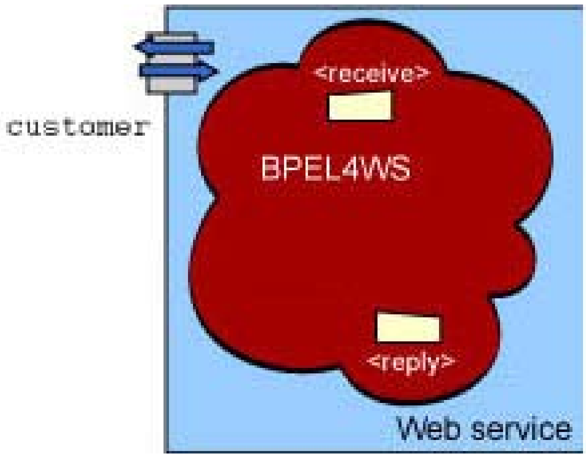


Figure 16 External View BPEL4WS Process

The behavior above consists of getting a message, then invoking the financial institution's Web service, and finally replying to the customer. These three actions are defined in BPEL4WS using the <receive>, <invoke>, and <reply> activities. However, the process needs to define the relation of such simple activities to each other in order to know how and when to run them.

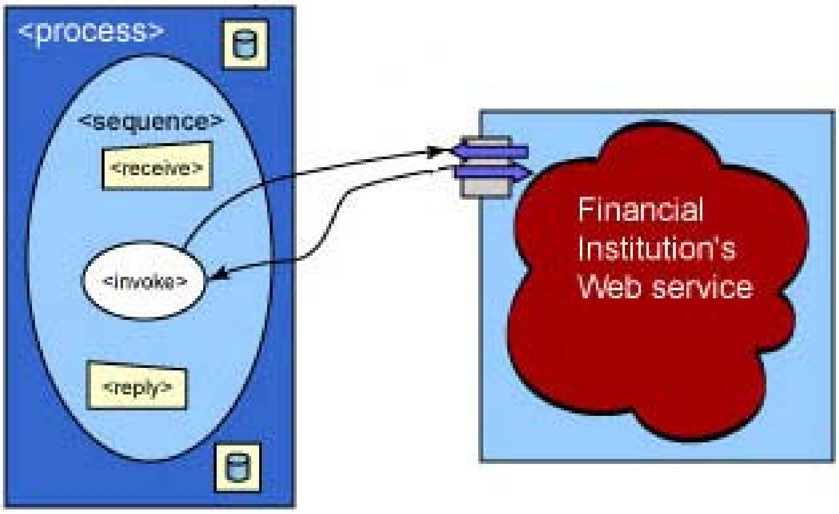


Figure 17 Internal view BPEL4WS Process

Such relations are defined in BPEL4WS by using structured activities that define restrictions on how to run the activities they enclose. In this example, you want the three to occur one after the other. This ordering may be achieved in BPEL4WS using a <sequence> activity, that would contain first the <receive> to consume the message, followed by an <invoke> to talk to the financial institution, and ending with a <reply> to send the answer to the customer. Therefore, the cloud above will contain a process that has a sequence of these three activities, and can invoke the financial institution

BPEL4WS extensions

BPEL4WS was released along with two others specifications

- **WS-Coordination**: describes how services can make use of pre-defined coordination contexts to subscribe to a particular role in a collaborative activity.
- **WS-Transaction**: provides a framework for incorporating transactional semantics into coordinated activities. In essence, WS-Transaction uses WS-Coordination to extend BPEL4WS to provide a context for transactional agreements between services. Different agreements may be described in an attempt to achieve consistent, desirable behavior while respecting service autonomy.Specification available at [WSTRANS].

Specification: Business Process Execution Language for Web Services, Version 1.0 [BWSIBM]

2.4.5 MDA / EDOC

The Enterprise Distributed Object Computing (EDOC) profile of UML was adopted by the OMG in November of 2001 as the standard for modeling enterprise systems. It is a modeling standard for Internet computing - providing for model driven development of enterprise systems based on the "Model Driven Architecture" (OMG-MDA).

The Model Driven Architecture defines an approach to enterprise distributed system development that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform. The UML Profile for Enterprise Distributed Object Computing (EDOC) represents a first attempt to define a PIM (Platform-Independent Model) along with several non-normative sketches of mappings to PSMs. (Platform-Specific Models) The potential benefits of such an approach are obvious: support for system evolution, high-level models that truly represent and document the implemented system, support for integration and interoperability, and the ability to migrate to new platforms and technologies as they become available. While technologies such as the Meta Object Facility (MOF) and the UML are well-established foundations on which to build PIMs and PSMs, there is as yet no well-established foundation on which to rely in describing how we take an instance of a PIM and transform it to produce an instance of a PSM.

The EDOC **Goals** can be stated as:

- Simplify the development of component based Enterprise (EDOC) systems by means of a modelling framework, based on UML 1.4 and conforming to the OMG Model Driven Architecture.

- Provide a platform independent, recursive collaboration based modelling approach that can be used at different levels of granularity and different degrees of coupling, for both business and systems modelling.

The **Component Collaboration Architecture (CCA)**[12] forms the architectural and modelling foundation for EDOC. CCA provides the base modelling concepts and notation that are required to support enterprise computing using XML, Web Services, ebXML, .NET, EJB,

[12] http://www.enterprise-component.com/products/edoc_rfp.htm

CORBA and other middleware technologies. CCA is the "normal form" for open enterprise computing, which includes B2B, B2C, EAI, Supply chain and customer support applications. Using the MDA concepts, CCA tools can generate WSDL, ebXML-Business Process Specifications and other technology artifacts. EDOC is aligned with ebXML and the OMG-EAI specification. Not yet standard mappings have also been produced for WSDL, J2EE and other technologies.

Additional information at: http://www.omg.org/cgi-bin/doc?ad/2001-06-09

2.4.6 XPDL

The WfMC[13] has identified five functional interfaces to a workflow service as part of its standardization program. The XML Processing Description Language (XPDL) [XPDL] specification forms part of the documentation relating to "Interface one" - **supporting Process Definition Import and Export**. This interface includes a common meta-model for describing the process definition (this specification) and also an XML schema for the interchange of process definitions.

A variety of different tools may be used to analyse, model, describe and document a business process. The workflow process definition interface defines a common interchange format, which supports the transfer of workflow process definitions between separate products.

The specification uses XML as the mechanism for process definition interchange. XPDL forms a common interchange standard that enables products to continue to support arbitrary internal representations of process definitions with an import/export function to map to/from the standard at the product boundary. The interface also defines a formal separation between the development and run-time environments, enabling a process definition, generated by one modelling tool, to be used as input to a number of different workflow run-time products. A workflow process definition, generated by a build-time tool, is capable of interpretation in different workflow runtime products. Process definitions transferred between these products or stored in a separate repository are accessible via that common interchange format.

To provide a common method to access and describe workflow definitions, a workflow process definition meta-data model has been established. This meta-data model identifies commonly used entities within a process definition. A variety of attributes describe the characteristics of this limited set of entities. Based on this model, vendor specific tools can transfer models via a common exchange format.

One of the key elements of the XPDL is its extensibility to handle information used by a variety of different tools. XPDL may never be capable of supporting all additional information requirements in all tools. Based upon a limited number of entities that describe a workflow

[13] Workflow Management Coalition

process definition (the "Minimum Meta Model"), the XPDL supports a number of differing approaches.

One of the most important elements of XPDL is a generic construct that supports vendor specific attributes for use within the common representation. We recommend that any missing attributes be proposed to the WfMC interface one workgroup for inclusion in future releases.

The specification describes a meta-model, which is used to define the objects and attributes contained within a process definition. The XPDL grammar is directly related to these objects and attributes. This approach needs two operations to be provided by a vendor:

- Import a workflow definition from XPDL.
- Export a workflow definition from the vendor's internal representation to XPDL.

A vendor can use a XSL style sheet to comply with those two operations.

Current version at: http://www.wfmc.org/

2.4.7 UML

UML activity diagrams [UML] are intended to model both computational and organizational processes (i.e. workflows). However, if activity diagrams are to succeed as a standard in the area of organizational process modeling, they should compare favorably to the languages currently used for this purpose, that is, those supported by existing Workflow Management Systems (WFMS).

UML activity diagrams are special cases of UML state diagrams, which in turn are graphical representations of state machines. State machines are transition systems whose arcs are labeled by ECA (Event-Condition-Action) rules. The occurrence of an event fires a transition if the machine is in the source state of the transition,

- the type of the event occurrence matches the event description of the transition, and
- the condition of the transition holds.
- The event (also called trigger), condition (also called guard), and action parts of a transition are all optional. A transition without an event is said to be triggerless. Triggerless transitions are enabled when the action or activity attached to their source state is completed.

A state can contain an entire state machine within it, leading to the concept of compound state. Compound states come in two flavours: OR and AND. An OR-state contains a single state-chart, while an AND-state contains several state-charts (separated by dashed lines) which are intended to be executed concurrently. Each of these state-charts is called a concurrent region. When a compound state is entered, its initial transition(s) are taken. The execution of a compound state is considered to be complete when it reaches (all) its final state(s). Initial states are denoted by filled circles, while final states are denoted by two concentric circles: one filled and one unfulfilled (see Figure 18).

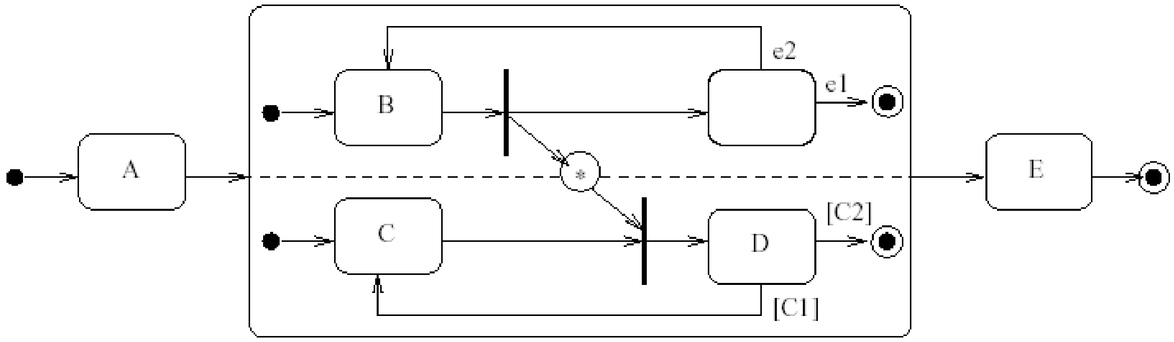


Figure 18 An example of an activity diagram [UML]

Actions or sequences of actions can be attached to basic (i.e. non-compound) states. In this respect, one can distinguish the following kinds of basic states:

- Wait state: no action or activity is performed. A state of this kind is exited when one of its outgoing transitions due to an event occur.

- Action state: a single action is attached to a state. The execution of an action is non-interruptible, so that the transitions emanating from such a state cannot fire until the action is completed.

- Activity-in-state: an activity (expressed as a sequence of actions) is attached to the state. The execution of this activity can be aborted prior to its completion if one of the state's outgoing transitions fires. We found no definition of the term “activity abortion" in the standard, so it is not clear if an activity abortion means that no more actions in the sequence are executed (interruption semantics), or if it means that the system's state before the activity's commencement is restored.

2.4.8 PSL – Process Specification Language

The Process Specification Language (**PSL**) is an interchange format designed to help exchange process information automatically among a wide variety of manufacturing applications such as process modeling, process planning, scheduling, simulation, workflow, project management, and business process re-engineering tools ([Schlenoff]). Tools can interoperate by translating between their native format and PSL. Then, any system is able to automatically exchange process information with any other system via PSL. PSL can be used to define formal semantics for process specification in WSMF [WSMF].
This representation would facilitate communication among the various applications because they would all have a common understanding of concepts to be shared.
Analogy: PSL is for discrete process data as STEP (Standard for the Exchange of Product Model Data) is for product data.

2.4.9 Comparison

In [WoFlo] a set of workflow patterns is proposed, which address comprehensive workflow functionality. These patterns provide the basis for comparison of process modeling and workflow systems.

basic control

1. **Sequence**
   An activity in a workflow process is enabled after the completion of another activity in the same process.
2. **Parallel split**
   A point in the workflow process where a single thread of control splits into multiple threads of control which can be executed in parallel, thus allowing activities to be executed simultaneously or in any order.
3. **Synchronisation**
   A point in the workflow process where multiple parallel subprocesses/activities converge into one single thread of control, thus synchronizing multiple threads. It is an assumption of this pattern that each incoming branch of a synchronizer is executed only once.
4. **Exclusive choice**
   A point in the workflow process where, based on a decision or workflow control data, one of several branches is chosen.
5. **Simple Merge**
   A point in the workflow process where two or more alternative branches come together without synchronization. It is an assumption of this pattern that none of the alternative branches is ever executed in parallel.

advanced branching and synchronisation

6. **Multiple choice**
   A point in the workflow process where, based on a decision or workflow control data, a number of branches are chosen.
7. **Synchronizing merge**
   A point in the workflow process where multiple paths converge into one single thread. If more than one path is taken, synchronization of the active threads needs to take place. If only one path is taken, the alternative branches should reconverge without synchronization. It is an assumption of this pattern that a branch that has already been activated, cannot be activated again while the merge is still waiting for other branches to complete.
8. **Multiple merge**
   A point in a workflow process where two or more branches reconverge without synchronization. If more than one branch gets activated, possibly concurrently, the activity following the merge is started for every activation of every incoming branch.
9. **Descriminator**
   The discriminator is a point in a workflow process that waits for one of the incoming branches to complete before activating the subsequent activity. From that moment on it waits for all remaining branches to complete and "ignores" them. Once all incoming branches have been triggered, it resets itself so that it

can be triggered again (which is important otherwise it could not really be used in the context of a loop).

structure

10. **Arbitrary cycles**
    A point in a workflow process where one or more activities can be done repeatedly.
11. **Implicit termination**
    A given subprocess should be terminated when there is nothing else to be done. In other words, there are no active activities in the workflow and no other activity can be made active (and at the same time the workflow is not in deadlock).

multiple instances

12. **Multiple Instances without synchronization**
    Within the context of a single case (i.e., workflow instance) multiple instances of an activity can be created, i.e., there is a facility to spawn off new threads of control. Each of these threads of control is independent of other threads. Moreover, there is no need to synchronize these threads.
13. **Multiple Instances with a priori design time knowledge**
    For one process instance an activity is enabled multiple times. The number of instances of a given activity for a given process instance is known at design time. Once all instances are completed some other activity needs to be started.
14. **Multiple Instances with a priori runtime knowledge**
    For one case an activity is enabled multiple times. The number of instances of a given activity for a given case varies and may depend on characteristics of the case or availability of resources, but is known at some stage during runtime, before the instances of that activity have to be created. Once all instances are completed some other activity needs to be started.
15. **Multiple Instances without a priori runtime knowledge**
    For one case an activity is enabled multiple times. The number of instances of a given activity for a given case is not known during design time, nor is it known at any stage during runtime, before the instances of that activity have to be created. Once all instances are completed some other activity needs to be started. The difference with Pattern 14 is that even while some of the instances are being executed or already completed, new ones can be created.

state

16. **Deferred choice**
    A point in the workflow process where one of several branches is chosen. In contrast to the XOR-split, the choice is not made explicitly (e.g. based on data or a decision) but several alternatives are offered to the environment. However, in contrast to the AND-split, only one of the alternatives is executed. This means that once the environment activates one of the branches the other alternative branches are withdrawn. It is important to note that the choice is delayed until the processing in one of the alternative branches is actually started, i.e. the moment of choice is as late as possible.
17. **Interleaved parallel routing**
    A set of activities is executed in an arbitrary order: Each activity in the set is

executed, the order is decided at run-time, and no two activities are executed at the same moment (i.e. no two activities are active for the same workflow instance at the same time).

18. **Milestone**

    The enabling of an activity depends on the case being in a specified state, i.e. the activity is only enabled if a certain milestone has been reached which did not expire yet. Consider three activities named A, B, and C. Activity A is only enabled if activity B has been executed and C has not been executed yet, i.e. A is not enabled before the execution of B and A is not enabled after the execution of C. Figure 16 illustrates the pattern. The state in between B and C is modeled by place m. This place is a milestone for A. Note that A does not remove the token from M: It only tests the presence of a token.

cancellation

19. **Cancel activity**

    An enabled activity is disabled, i.e. a thread waiting for the execution of an activity is removed.

20. **Cancel case**

    A case, i.e. workflow instance, is removed completely (i.e., even if parts of the process are instantiated multiple times, all descendants are removed).

For each standard-pattern combination, a group at the Eindhoven University [EindUni] checked whether it is possible to realize the workflow pattern with the language. If a standard directly supports the pattern through one of its constructs, it is rated +. If the pattern is not directly supported, it is rated +/-. Any solution which results in incomprehensible diagrams or coding, is considered as giving no direct support and is rated -. Note that a pattern is only supported directly if there is a feature provided by the language which supports the construct without resorting to any of solutions mentioned in the implementation part of the pattern.

Pattern	Standard						
	BPEL4WS	XLANG	WSFL	XPDL	UML	WSCI	BPML
1 Sequence	+	+	+	+	+	+	+
2 Parallel split	+	+	+	+	+	+	+
3 Synchronization	+	+	+	+	+	+	+
4 Exclusive choice	+	+	+	+	+	+	+
5 Simple merge	+	+	+	+	+	+	+
6 Multichoice	+	-	+	+	-	-	-
7 Synchron. merge	+	-	+	-	-	-	-
8 Multimerge	-	-	-	-	-	+/-	+/-
9 Discriminator	-	-	-	-	-	-	-
10 Arbitrary cycles	-	-	-	+	-	-	-
11 Impl. termination	+	-	+	+	-	+	+
12 Multiple instances without synchron.	+	+	+	-	-	+	+
13 Multiple instances with a priori design time knowledge	+	+	+	+	+	+	+

Table 1 Standard-pattern / standards relationship

Pattern	Standard						
	BPEL4WS	XLANG	WSFL	XPDL	UML	WSCI	BPML
14 Multiple instances with a priori runtime knowledge	-	-	-	-	+	-	-
15 Multiple instances without a priori runtime knowledge	-	-	-	-	-	-	-
16 Deferred choice	+	+	-	-	+	+	+
17 Interleaved paral. routing	+/-	-	-	-	-	-	-
18 Milestone	-	-	-	-	-	-	-
19 Cancel activity	+	+	+	-	+	+	+
20 Cancel case	+	+	+	-	+	+	+

Table 2 Standard-pattern / standards relationship (continued)

Business process control flow standards and paradigms are becoming important for semantic web services. In current web service description languages services are described in terms of processes with input and output parameters. The description of conditions and effects aims at the realization of automatic invocation, composition and monitoring of web services. Here patterns and ideas from workflow and description of process execution can be directly adopted to model execution semantics. Table 2 shows that all the reviewed languages for process description offer the basic control patterns and that some of the more advanced patterns are only sparsely supported. For semantic web services these basic patterns should form a good basis to start with and to implement elementary scenarios upon.

2.5 Business Transactions

Business transaction protocols (BTPs) [BTP] are an integral part of business process management systems and more widely, B2B, since they deal with managing long running transactions.

2.5.1 Business Transaction Protocol (BTP)

The *Business Transaction Protocol* (BTP) is an XML-based protocol for representing and seamlessly managing complex, multi-step business-to-business (B2B) transactions over the Internet. The protocol allows complex XML message exchanges to be tracked and managed as loosely coupled 'conversations' between and among businesses. BTP goes beyond the problem domain currently being addressed by ebXML and is independent of transport protocols and messaging frameworks [BTP].

Purpose
The purpose of BTP is to put loosely coupled software services (e.g. Web services) into a single business transaction. There are two kinds of business transactions: cohesive and atomic. The initial version of the standardized protocol focuses on atomic business transactions, but within a scope where they are components of cohesive business transactions. The transaction model and the actors that are described in this document are applicable to the atomic transactions. Atomic business transactions are made up of services that all agree to enforce a common outcome of the transaction: In case of a failure all services un-do (compensate, roll-back) their operations that were invoked during the transaction, in case of a success all services make the results of their operation permanent. There is no assumption as to the mechanisms used by the services to achieve the un-do of the operations. Cohesive business transactions are made up of several atomic transactions. The atoms forming a particular cohesion do not necessarily have a common outcome. Some may be performed (confirmed), others may fail (cancelled i.e. their operations are undone) [BTP].

History and Development
BEA Systems presented a draft for its BTP to a newly formed OASIS technical committee, that works on business transactions, in January 2001. BEA is a member of OASIS. The OASIS technical committee has not approved the draft at the time this report is written. In May 2001 the OASIS Technical Committee for Business Transactions published the Scope and Requirements document [BTP] as part of the ongoing work in the area of business transactions.
Technical Aspects

The BTP protocol is used for communication between a transaction coordinator and the participants of a transaction. A transaction coordinator manages different services from different parties that take part in a business case. The transaction coordinator decides whether the services in the business case a committed or rolled back. Figure 8 depicts the relation between a business coordinator and the services included in a business case. BTP defines the content and sequence of messages that are sent between actors, and the contracts that determine their reactions.

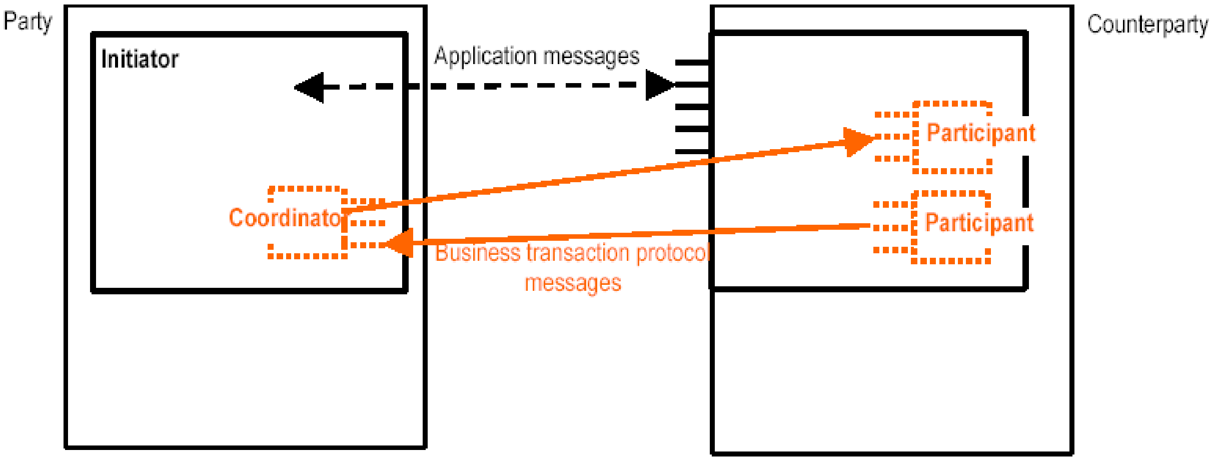


Figure 19 BTP Architecture [BTP]

Service providers have decided to make their services “BTP capable” and have advertised this fact (outside the scope of BTP). An initiator decides to create an atomic business transaction (not shown in the diagram), which means that a transaction coordinator is created to coordinate any participants that get involved in that atomic business transaction.

An atomic transaction has an id (called Atom Identifier) that is attached to an application message and is sent to an operation (e.g., service method). If the operation consists of actions that are capable of being undone by the transaction coordinator, then the operation enrolls a participant. A message is sent back to the coordinator, telling it about the participant (which is identified by a Participant Identifier). In the process of these exchanges both the coordinator and the participant get each other’s Address. Any work that a service does and that is related to an Atomic Business Transaction will be tagged with the participant id. (In fact, it may be convenient to group units of work into separate, multiple participants, which are used by the service and each of which is enlisted with the coordinator).

At some point the initiator decides to terminate the Atomic Business Transaction, which causes prepared messages to be sent to all enrolled participants. The participant, on receiving this message, should log the information required to either confirm or cancel the work done for this transaction, so that it can either complete the work of the transaction, or undo it. If the participant can complete the work, it sends a “VOTE/Ready” message to the coordinator; if the participant is not able to do so, it sends a “VOTE/Cancel” message back to the transaction coordinator. The messages between the coordinator and the participant are Business Transaction Protocol messages. If the coordinator receives any VOTE/Cancel messages then it sends a CANCEL message to all registered participants of the Atomic Business Transaction. Otherwise it waits to be told by the initiator whether to send a “CANCEL” or a” CONFIRM” message to all participants. The participants do whatever makes sense to them, in either case. A cancel might reverse database changes, or do some other compensatory work that makes sense for the Web service provider. The initiator is not aware of the details, but it may know that the contract it has with the service implies certain things about a cancellation (like the Web service will not go ahead with a credit card transaction) [BTP].

2.5.2 Transaction Authority Markup Language (XAML)

Transaction Authority Markup Language (XAML) is a vendor-neutral standard that enables the coordination and processing of online transactions in the rapidly emerging world of XML Web services, the revolutionary new model of Internet-based computing that is now being adopted by all major systems and software vendors. XAML is intended to be a completely open standard for web-based business transactions.

Purpose

The XAML standard defines a set of XML message formats and interaction models that Web services can use in order to provide business-level transactions that span multiple parties across the Internet. The following scenario demonstrates a business-level transaction involving a set of web services that would utilize XAML. Consider an industrial company that purchases benzene from a chemical manufacturer on the Web. In order for the buyer to purchase the benzene, she requires additional value-added services provided by third parties, such as shipping with specific delivery terms, payment financing, casualty insurance, and government compliance for safe transport. The buyer will not agree to the purchase of benzene until all of these services are available, and all meet her requirements. She will purchase all of them or none of them. In other words, all of these inter-related requirements need to be satisfied in order for the business transaction to be completed.

This scenario requires that the industrial company initiate a set of calls to Web services that are owned by the various product and service providers mentioned above. Today, Web services provide the low-level means of supporting basic requests to distributed systems, using protocols such as HTTP, XML, SOAP, and other industry-specific data formats. However, in this scenario, the industrial company needs to be able to engage these web services to form a single business transaction.

In order to do this, the requestor must be able to coordinate the calling of the individual Web services, and must have a means of asking for commitment from all of them, prior to actually committing. The process involves multiple interactions with each of the Web services, and multiple stages of progress towards each Web service’s transactional completion, with the ability to commit and cancel operations, and, in some cases, initiate compensating Web services that undo or reverse the work of previous transactions [XAML]. XAML fits into the overview picture of Web Services as add-on to the communication between client and Web Services to monitor and allow transactions using Web services.

Technical Aspects

XAML is an XML markup language used with Web services, along with a set of interaction models that establishes a means by which Web services can perform transactional units of work, and participate in business-level transactions that span multiple Web services. XAML enables a Web service to provide transactional capabilities, including commit, cancel, and compensatory actions. XAML also enables software systems to coordinate the calling of multiple Web services, so as to ensure that all of the work, or none of the work, is completed.

Consider a transaction that has to span two distributed Web services at the sites One.com and Two.com. Through the use of XAML-encoded messages, the calling system issues a Web service request to One.com. One.com replies, indicating that it can perform the service. Next, the calling system obtains the same level of promise from the Web service at Two.com. At this point, the calling system can request activity from both One.com and Two.com. When the activity is complete, the calling system can obtain status on the activity from each Web service, and can request to have that work committed, canceled, or compensated for, at each site. Depending on the success or failure of the work provided by each Web service, the software handling the business transaction could coordinate the desired overall outcome.

XAML extends transactional capabilities to Web services, enabling businesses to offer fine-grained interaction with their Web services. This is critical to the software that provides business transactions and must ensure integrity of the high-level process [XAML].

3 Workflow Systems

Before comparing the products (some based on workflow patterns as well as some older products – but for the sake of completeness they are listed in this section), we briefly introduce each product and supply some background information (see .3.1 to 3.15).

3.1 Staffware

Staffware[14] is one of the leading workflow management systems. Staffware is authored and distributed by Staffware PLC. In 1998, it was estimated by the Gartner Group that Staffware has 25 percent of the global market. The routing elements used by Staffware are the Start, Step, Wait, Condition, and Stop. The Step corresponds to an activity which has an OR-join/AND-split semantics. The Wait step is used to synchronize flows (i.e. an AND- join) and conditions are used for conditional routing (i.e. XOR-split). Arbitrary loops are supported. There is no direct provision for multiple instances nor for the advanced synchronization constructs. There is no need to define explicit termination points, i.e. termination is implicit. Staffware does not offer a state concept. The so-called “withdraw" transition allows the Cancel Activity pattern to be supported. No support is available for Cancel Case.

3.2 COSA

COSA[15] is a Petri-net-based workflow management system developed by Ley GmbH (formerly operating under the names Software Ley, COSA Solutions, and Baan). Ley GmbH is a German company based in Pullheim (Germany) and is part of Thiel Logistik AG. COSA is one of the leading workflow management systems in Europe and can be used as a stand-alone workflow system or as the workflow module of the Baan IV ERP system. This evaluation is based on version 3.0. The modeling language of COSA consists of two types of building blocks: activities (i.e., Petri net transitions) and conditions (i.e. Petri net places). COSA extends the classical Petri net model with control data to allow for explicit choices based on information and decisions. Unfortunately, only safe Petri nets are allowed, i.e., it is not allowed to have multiple tokens in one place. Therefore, COSA is unable to support multiple instances directly.

The only way to deal with multiple instances is to use workflow triggers. Every sub-process in COSA has a unique start activity and a unique end activity. As a result, only highly structured sub-processes are possible and termination is always explicit. The main feature of the workflow language of COSA is that it allows for the explicit representation of states. As a result, state-based patterns such as the Deferred Choice, and Interleaved Parallel Routing are supported in a direct and graphical manner. Tokens can be removed from places, providing support for Cancel Activity, however COSA does not have an explicit provision for Cancel Case other than through its API.

[14] http://www.staffware.com/

[15] http://www.cosa.nl/uk.asp

3.3 InConcert

InConcert[16] has been established in 1996 as a Xerox fully-owned subsidiary. In 1999 it has been bought by TIBCO Software. This evaluation is based on InConcert 2000 (version 5.1). An InConcert workflow definition is called a job". A job can contain none, one or many activities. An activity is either simple or compound. An activity can be connected to an arbitrary number of other activities but circular dependencies are not allowed. Each activity has a perform condition attached to it. The default setting of the perform condition is true such that activities can be executed in general. If the perform condition evaluates to false, the activity is skipped. If an activity is skipped, then the subsequent activities are not skipped automatically. Conditional branching or case branching can be achieved by parallel activities with different perform conditions. Arbitrary cycles are not supported. An explicit termination point is not required. There is no direct provision for multiple instances nor for direct implementation of the state-based patterns. The cancellation patterns are not supported.

3.4 Eastman Software

Eastman Software offers a variety of imaging products. Their software is used to electronically capture, share, display, fax, print, and store vital document-based information. On top of their imaging products, Eastman Software also offers a workflow management system. Enterprise Workflow 4.0, a component of the Eastman Software Enterprise Work Manager Series, provides a so-called RouteBuilder tool to design workflow processes consisting of different types of work steps. The following types of work steps (i.e., activity types) are supported: custom, system, archive, print, OCR, fax, transfer, program, rendezvous, split, and join. The standard semantics of a work step is an XOR-join/XOR-split semantics. The rendezvous, split, and join steps have been added to allow for parallel routing. For each join step, the user can indicate how many threads need to be synchronized. Moreover, using techniques based on the number of active parallel threads, join steps are bypassed if synchronization is not possible. This leads to constructs similar to the false-token propagation in MQSeries.

3.5 FLOWer

FLOWer[17] is Pallas Athena's case handling product. This evaluation is based on version 2.05. FLOWer can be used for flexibly structured processes, but also supports traditional production workflow functionality. The case handling mechanisms of FLOWer solve many of the flexibility problems of traditional workflow management systems. Flexibility is guaranteed through data-driven workflows, redo and skip capabilities, and activity independent forms. FLOWer consists of a number of components: FLOWer Studio, FLOWer Case Guide, FLOWer CFM, FLOWer Queues/Queries, FLOWer Integration Facility, and FLOWer Management Information and Case History Logging. FLOWer Studio is the graphical design environment. It is used to define processes, activities, precedences, data objects, and forms. FLOWer Case Guide is the client application which is used to handle individual cases. FLOWer queue corresponds to the worktray, worklist or in-basket of traditional WFM systems. The FLOWer queue provides a refined mechanism to look for cases satisfying

[16] http://www.tibco.com/solutions/products/active_enterprise/in_concert/default.jsp

[17] http://www.pallas-athena.com/

specified search criteria. FLOWer CFM (ConFiguration Management) is used to define users (i.e. actors), work profiles, and authorization profiles. The profiles are used to map users onto roles. FLOWer CFM is also available at the operational level to allow for run-time flexibility. FLOWer Management Information and Case History Logging can be used to store and retrieve management information at various levels of detail. FLOWer Integration Facility provides the functionality to interface with other applications. The modeling language of FLOWer is block-structured. Blocks are named plans can be nested and there are five types of plans: static, dynamic, sequential, user decision and system decision. The static plan is used to specify subprocesses. The dynamic subplan is used to model multiple instances. The sequential subplan is used to model iteration. The user decision corresponds to the deferred choice and the system decision corresponds to the explicit choice.

3.6 DominoWorkflow

Domino Workflow[18] is the workflow extension of the widely used groupware product Lotus Domino/Notes (Lotus/IBM). Clearly, the tight integration with the groupware product is one of the attractive features of this product. The marriage between groupware (Lotus Domino/Notes) and workflow (Domino Workflow) allows for partly structured workflows.

There are various types of resource classes, e.g., person (singleton), workgroup (including inheritance and many-to-many relationships), department (only one-to-many relationships, however with inheritance), and roles. Each routing relation is of one of the following types: (1) always (for AND-split) (2) exclusive choice (for XOR-split made by the user at the end of the activity), (3) multiple choice (for OR-split made by the user after completing the activity), (4) condition (automatically evaluated on the basis of data elements), and (5) else (only taken if none of the other routing relations is activated). Each activity can serve as a join. The type of join is determined implicitly. Joins are either enabled or disabled. If a join is disabled, it serves as an XOR-join, i.e., the activity is enabled the moment one of the preceding activities completes. If the join is enabled, it continuously checks whether potentially it can receive more inputs in the future without activating itself. This way it is possible to make AND-joins or use more advanced synchronization mechanisms.

3.7 Meteor

Meteor[19] has been tested by several industry partners and is in the process of being commercialized by Infocosm Inc. A workflow in Meteor is defined as a collection of activities and dependencies. An activity can be any combination of AND/XOR-joins and AND/XOR-splits and there are two types of dependencies: control dependencies and data dependencies. The focus of Meteor is on transactional features and distribution aspects. The workflow modeling language supports few of the more advanced constructs. For example, it is not possible to handle any of the state-based patterns, multiple instances are not supported explicitly, termination is always explicit, and the Synchronization merge, Discriminator and cancellation are not supported. The Multi-merge and Arbitrary cycles patterns are supported.

[18] http://www.lotus.com/

[19] http://lsdis.cs.uga.edu/proj/meteor/meteor.html

3.8 Mobile

Mobile is a workflow management system developed by members of the Database Systems group at the University of Erlangen/Nuernberg (Germany). It is a research prototype with several interesting features, e.g. the system is based on the observation that a workflow comprises many perspectives and one can reuse each perspective separately. This evaluation is based on the 1999 version of Mobile. The control-flow perspective of Mobile offers various routing constructs to link so-called workflow types. A workflow type is either an elementary activity or the composition of other workflow types. A powerful feature of the Mobile language is that the set of control-flow constructs is not fixed, i.e. the language is extensible. It is possible to add any of the design patterns identified in this paper as a construct.

To add a construct, one can use the Mobile editor MoMo to add the graphical representation of the construct. The semantics is expressed in terms of Java. Since the Java code has direct access to the state of the workflow instance, all routing constructs can be supported. The fact that the language is extensible makes the workflow language of Mobile hard to compare with the other languages. The standard constructs of Mobile include, in addition to the basic patterns, the N-out-of-M join and Interleaved Parallel Routing.

3.9 MQSeries/Workflow

MQSeries/Workflow[20] is the successor of IBM's workflow offering, FlowMark. Flow-Mark was one of the first workflow products that was independent from document management and imaging services. It has been renamed to MQSeries/Workflow after a move from the proprietary middleware to middleware based on the MQSeries product. This evaluation is based on version 3.1 of the product. The workflow model consists of activities linked by transitions. Other than a decomposition block, few other special modeling constructs are available. The workflow engine of MQSeries/Workflow has a unique execution semantics in that it propagates a False Token for every transition with a condition evaluating to False. This allows for every activity that has more than one incoming transition to act as a synchronizing merge.

Other than the synchronizing merge, which is a natural construct for MQSeries/Workflow, there is no way to directly implement any of the other advanced synchronization patterns. Support for multiple instances is provided through the Bundle construct although it is not suitable if the number of instances is not known at any point prior to generating the instances involved. Arbitrary loops are not supported. An explicit termination point is not required and the workflow process will terminate when there is nothing else to be executed. There is no direct way to model the state-based and cancellation patterns.

[20] http://www-3.ibm.com/software/ts/mqseries/workflow/

3.10 Sun / Forte

Forte Conductor is a workflow engine that is an add-on to Forte's development environment, Forte 4GL (formerly Forte Application Environment). Conductor's engine is based on experimental work performed at Digital Research and its modeling language is powerful and flexible. Forte Software has been acquired by Sun Microsystems and subsequently became part of iPlanet E-Commerce Solutions. In late 2000 version 3.0 of the product became an integral part of iPlanet Integration Server.This evaluation is based on version 1.0 of the product. The workflow model in Conductor comprises a set of activities connected with transitions (called Routers). Each transition has associated transition conditions.

Each activity has a trigger that determines the semantics of that activity if it has more than one incoming transition. The triggers are flexible enough for easy specification of OR-join, AND-join and N-out-of-M join although the semantics of such a specification is implicit and not visible to the end-user. Arbitrary cycles are supported, but explicit termination points are required. Forte supports creation of multiple instances directly (through the use of a multi-merge join) but does not support any direct means of their subsequent synchronization. State-based patterns cannot be realized. Forte does not have a construct for Cancel Activity but Cancel Case is available through its termination semantics - when an activity is executed which has no other triggers, it will terminate that workflow decomposition.

3.11 Verve

Verve[21] is debuted in 1998. In late 2000 it was acquired by Versata and renamed Versata Integration Server (VIS). This evaluation is based on version 2.1 of the product that was released just before the acquisition by Versata. What makes Verve Workflow Engine an interesting workflow product is that it has been designed from the ground up as an embeddable workflow engine. The workflow engine of Verve is very powerful and amongst other features allows for multiple instances and dynamic modification of running instances. The Verve workflow model consists of activities connected by transitions. Each transition has an associated transition condition. Extra routing constructs such as synchronizer and discriminator are supported. Arbitrary loops are supported. An explicit termination point is required. Multiple instances are directly supported (through the use of the multi-merge) as long as they do not require subsequent synchronization. There is no direct way to implement state-based patterns. Of the cancellation patterns, Cancel Case is supported through the forced termination by the first of the last activities which terminates.

3.12 Visual WorkFlo

Visual WorkFlo is one of the market leaders in the workflow industry. It is part of the FileNet's Panagon suite (Panagon WorkFlo Services) that includes also document management and imaging servers. Visual WorkFlo is one of the oldest and best established products on the market. Since its introduction in 1994 it managed to gain a respectable share of all worldwide workflow applications. FileNet as a corporation ranks amongst the top 60 software companies in the world. Value Added Resellers building solutions on top of Panagon's suite. This evaluation is based on version 3.0 of the product. The workflow modeling language of

[21] http://www.versata.com/versata.vjsp?pageid=240

Visual WorkFlo is highly structured and is a collection of activities and routing elements such as Branch (XOR-split), Web methods integration platform While (structured loop), Static Split (AND-split), Rendezvous (AND-join), and Release. Visual WorkFlo does not directly support any of the advanced synchronization patterns. It requires the model to have structured loops only and one, explicit, termination node thus limiting the suitability of the resulting specifications. Direct support for Multiple Instances is possible through the Release construct as long as there is no further synchronization required. There is no direct way to implement any of the state-based patterns. There is no explicit support for the cancellation patterns.

3.13 HP / Changengine

Changengine[22] is a workflow offering from HP, the second largest computer supplier in the world. The first major version of the product, 3.0, was introduced in 1998 and it focused on high performance and support for dynamic modifications. In late 2000 the product changed its name to HP Process Manager to better convey the purpose of the product to the customers. This evaluation is based on version 4.0, introduced in early 2000. Workflow models in Changengine consist of a set of work nodes and routers linked by arcs. A work node can have only one incoming and one outgoing arc. If more transitions are required, they have to be created explicitly through the router node. Router node semantics is determined by the set of route rules. Arbitrary loops are allowed. Changengine does not provide any support for multiple instances. The termination policy is rather unusual: the process will terminate once all process nodes without outgoing activities (End Points) are reached. There is no direct way to implement the state-based patterns. A routing rule associated with an activity can be set to cause termination of a decomposition, thus supporting Cancel Case. The Cancel Activity pattern is not supported.

3.14 I-Flow

I-Flow[23] is a workflow offering from Fujitsu that can be seen as a successor of the workflow engine from the same company, TeamWare. I-Flow is web-centric and has a Java/CORBA based engine built specifically for Independent Software Vendors and System Integrators. This evaluation is based on version 3.5 of the product, introduced in early 2000. As of the beginning of 2002 the latest version of the product is 4.1. The workflow model in I-Flow consists of activities and a set of routing constructs connected by transitions (called Arrows). Routing constructs include Conditional Node (XOR-split), OR-NODE (Merge), and AND-NODE (synchronizer). The AND-split can be modeled implicitly by providing an activity with more than one outgoing transition. Multiple instances can be implemented using the Chained Process Node which allows for asynchronous subprocess invocation. Arbitrary loops are allowed but the process requires an explicit termination point. There is no direct way to implement state-based patterns. Cancel Case but not Cancel Activity is supported.

[22] http://www.hp.com

[23] http://www.i-flow.com/

3.15 SAP R/3 Workflow

SAP[24] is the main player in the market of ERP systems. Its R/3 software suite includes an integrated workflow component that have been evaluated independently of the rest of R/3. This evaluation is based on release 3.1 of the product. Note that SAP workflow should not be confused with EPCs (Event-driven Process Chains) found in ARIS and in other parts of the SAP system. EPCs there are used entirely for business process modeling purposes and not for modeling executable workflows in the SAP R/3 runtime environment. SAP R/3 Workflow imposes a number of restrictions on the use of EPCs. EPCs that are used for workflow modeling consist of a set of functions (activities), events and connectors (AND, XOR, OR). However, in SAP R/3 Workflow not the full expressive power of EPCs can be used, as there are a number of syntactic restrictions similar in vein to the restrictions imposed by Filenet Visual Workflo (e.g. every workflow needs to have a unique starting and a unique ending point, and-splits are always followed by and-joins, or-splits by or-joins etc). As such, there is no direct provision for the advanced synchronization constructs (with one exception: it is possible to specify for the join operator how many parallel branches it has to wait for, hence its semantics corresponds to the N-out-of-M join), multiple instances, arbitrary loops, state-based or cancellation patterns.

3.16 Comparison

The following two tables summarize the results of the comparison of the different workflow management systems in terms of the patterns introduced in section 2.4.9 [WoFlo]. These comparisons were provided on the web sites at University Eindhoven (see http://tmitwww.tm.tue.nl/staff/wvdaalst/ - Wil van der Aalst). Additionally at lot of information related to the field of business processes, workflows and Workflow Management Systems is available there.

For each product-pattern combination the tables show whether it is possible to realize the workflow pattern with the tool. If a product directly supports the pattern through one of its constructs, it is rated +. If the pattern is not directly supported, it is rated +/-. Any solution which results in incomprehensible diagrams or coding, is considered as giving no direct support and is rated -. Note that a pattern is only supported directly if there is a feature provided by the graphical interface of the tool (i.e., not in some scripting language) which supports the construct without resorting to any of solutions mentioned in the implementation part of the pattern.

[24] http://www.sap.com/

	Product							
pattern	*Staffware*	*COSA*	*InConcert*	*Eastman*	*FLOWer*	*Domino*	*Meteor*	*Mobile*
1 Sequence	+	+	+	+	+	+	+	+
2 Parallel split	+	+	+	+	+	+	+	+
3 Synchronization	+	+	+	+	+	+	+	+
4 Exclusive choice	+	+	+/-	+	+	+	+	+
5 Simple merge	+	+	+/-	+	+	+	+	+
6 Multichoice	-	+	+/-	+/-	-	+	+	+
7 Synchron. merge	-	+/-	+	+	-	+	-	-
8 Multimerge	-	-	-	+	+/-	+/-	+	-
9 Discriminator	-	-	-	+	+/-	-	+/-	+
10 Arbitrary cycles	+	+	-	+	-	+	+	-
11 Impl. termination	+	-	+	+	-	+	-	-
12 Multiple instances without synchron.	-	+/-	-	+	+	+/-	+	-
13 Multiple instances with a priori design time knowledge	+	+	+	+	+	+	+	+
14 Multiple instances with a priori runtime knowledge	-	-	-	-	+	-	-	-
15 Multiple instances without a priori runtime knowledge	-	-	-	-	+	-	-	-
16 Deferred choice	-	+	-	-	+/-	-	-	-
17 Interleaved paral. routing	-	+	-	-	+/-	-	-	+
18 Milestone	-	+	-	-	+/-	-	-	-
19 Cancel activity	+	+	-	-	+/-	-	-	-
20 Cancel case	-	-	-	-	+/-	+	-	-

Table 3 Product comparison overview

	Product						
pattern	MQSeries	Forte	Verve	Vis. WF	Changeng.	I-Flow	SAP/R3
1 Sequence	+	+	+	+	+	+	+
2 Parallel split	+	+	+	+	+	+	+
3 Synchronization	+	+	+	+	+	+	+
4 Exclusive choice	+	+	+	+	+	+	+
5 Simple merge	+	+	+	+	+	+	+
6 Multichoice	+	+	+	+	+	+	+
7 Synchron. merge	+	-	-	-	-	-	-
8 Multimerge	-	+	+	-	-	-	-
9 Discriminator	-	+	+	-	+	-	+
10 Arbitrary cycles	-	+	+	+/-	+	+	-
11 Impl. termination	+	-	-	-	-	-	-
12 Multiple instances without synchron.	-	+	+	+	-	+	-
13 Multiple instances with a priori design time knowledge	+	+	+	+	+	+	+
14 Multiple instances with a priori runtime knowledge	+/-	-	-	-	-	-	+/-
15 Multiple instances without a priori runtime knowledge	-	-	-	-	-	-	-
16 Deferred choice	-	-	-	-	-	-	-
17 Interleaved paral. routing	-	-	-	-	-	-	-
18 Milestone	-	-	-	-	-	-	-
19 Cancel activity	-	-	-	-	-	-	+
20 Cancel case	-	+	+	-	+	-	+

Table 4 Product comparison overview (continuation)

4 Conclusion

In the previous sections we have reviewed the current standards and technologies in the area of business-to-business eCommerce and workflow. The are centered around classification, document exchange, collaboration and workflow together with processes and transactions.

As in product catalog systems, classification is also important for Semantic Web Services. Services can be classified according to Taxonomies building a backbone for semantic resource description. However, current eCommerce classification standards lack the support for formal semantics and ontological structures that go beyond simple categorization trees. Their taxonomic relationship is not semantically well-founded.

Standards for document exchange are widely accepted and used in the industry and therefore cannot be ignored by Semantic Web Service solutions for the eCommerce area. For SWWS we intend to investigate within the project case studies which of the specifications can be applied in the corresponding scenario. In principle there are two ways to integrate document exchange standards into a semantically enabled service-based B2B scenario. First, the details of document description, as proposed by the standard, can be seen as black box just being put through from one partner to the other. Second, the internal structure of document description can be ontologically modeled and incorporated in the semantic service description mechanisms in order to use it for reasoning.

Successful collaboration of two enterprises by integrating their business processes via automated Web Service communication necessitates declarative semantics of the dynamics of the business processes involved. Otherwise the knowledge about this dynamics has to be hard-coded in the software agents and applications involved, which does not scale up. This makes standardization efforts around business interaction and protocol specification an important issue and valuable input for the work on Web Service description in SWWS.

Workflow, business processes and transactions are closely related. In terms of Semantic Web Services they describe the execution semantics of service invocation. Also here a declarative approach is needed to support reasoning about control flow for semantically enabled agents. Standardization efforts that are related to business processes are concerned with descriptions of control flow structures as known from programming languages. As described in section 2.4.9, five basic control flow patterns could be identified. They are common to all the specifications and, thus, seem to be a good basis for modeling execution semantics. For SWWS we have to see in concrete case study work which particular standardization effort can be either directly applied or be valuable input for our description of execution semantics.

5 Appendix: B2B Integration Systems

B2B integration is the automated exchange of information between different organizations. Business Integration software gives you the ability to integrate the diverse data and information sources both within and outside your enterprise into a single coherent framework. An integrated information infrastructure can then be shared by mission-critical applications such as CRM, executive information portals, and automated supply chain systems.

The three main tasks of B2B integration systems are Data Integration, Application Integration and Business Process Integration, as shown in Figure 20.

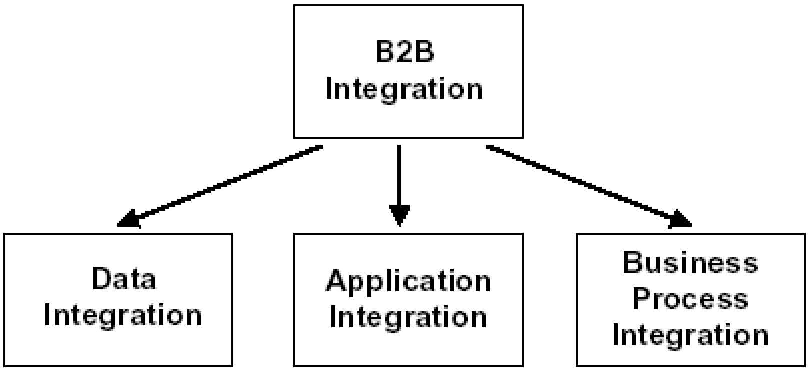


Figure 20 B2B Integration [ESBB]

The following B2B Integration Systems descriptions were taken from the book *B2B Integration - Concepts and Architecture* - by Christoph Bussler. [B2BBus1]

5.1 Oracle Oracle9iAS Integration

Oracle's offering for integration is 9iAS Integration 9.0.4[25] and 9iAS InterConnect 9.0.2. The latter is the preceding version of the former. Therefore, only 9iAS Integration is discussed in more detail.

9iAS Integration provides an integrated set of concepts for processes and data. The handling of data is implemented through the concept of events. An incoming message from a back end application system or trading partner (through a B2B protocol) is represented as a native event that contains the data format as received. A native event is converted into an application event through translation. The application event is in the common syntax of 9iAS Integration, but the values are still the same as in the native event. The application event is transformed into business events through transformation. Business events are in the common syntax and common terminology of 9iAS Integration. Business events are user defined and provide a common view or common representation of all application events of all B2B protocols or back end application systems (both subsumed under the concept party).

[25] http://www.oracle.com/ip/deploy/ias/integration/index.html?content.html

For example, if three different parties can send purchase orders then the business event purchase order would be a common representation into which every party's purchase order can be transformed into. Business events that are sent to parties are going through the same conversions. First, they are transformed into application events and then translated into native events.

Native events are created and consumed by the adapter framework of 9iAS Integration. The adapter framework allows to connect to adapters that are J2EE Connector Architecture compliant. Native events are converted into the representation as required by the adapter interface (i.e. records according to the connector standard) and sent out as messages. Message received by adapters are given to the adapter framework according to the adapter's interface. 9iAS Integration provides back end application system adapters as well as B2B protocol adapters. Native events can be correlated to determine related events. Various security mechanisms are provided according to B2B protocol requirements so that 9iAS Integration can participate in secure B2B interactions.

Process management is provided through several concepts. For each class of event a corresponding process has to be defined. This is called a role (the term role indicates behavior like seller or buyer). Native events are processed in native roles, application events are processed in application roles and business events are processed in business roles. For example, a RosettaNet Partner Interface Process (PIP) is modeled as a native role. This constitutes the public process. After translation the application events are processed in application roles in order to define the behavior of application events. Since native roles are different from application roles it is possible to build an abstraction. For example, it is possible to not pass acknowledgments on to application roles from native roles. This means that application roles do not have to deal with acknowledgments any more. The same applies to duplicate checks or time-out behavior. All that can be abstracted from. After transformation of the application events the resulting business events are given to business roles. These define the enterprise's behavior like seller, buyer, shipper and so on. Business logic is implemented in an additional concept called business process. Business roles are connected to business processes and therefore connect to parties through this approach. For example, a request for quotation-based buying would connect to two different business roles. One for handing the request for quotation exchange and one for handling the purchase order exchange.

Events are passed back and forth between roles through role parameters called ports. Ports are like input and output parameters and allow to bind event instances at runtime. Within roles and business processes process steps are available for process modeling like conditional steps and other constructs. Roles and business processes implement long-running transactions since intermediate processing states are externalized in the database. A party management component manages back end application systems as well as trading partners. Agreements are managed to define which events are accepted from which party and which event can be sent to a party. 9iAS Integration is a modeled environment where all aspects of integration are modeled. All definition data is stored in a database as values. At runtime the execution of events, roles and processes is interpreted based on one holistic database schema. The following table[26] shows the relationship of this product with the web service technology relevant to the SWWS project.

Vendor	Oracle	
Web Services	**Product Name**	Oracle 9iAS Web Services
	Version	
	Technology	J2EE
	Platforms	Windows NT/ 2000, Solaris, HP- UX, AIX, Tru64, Linux
	Solution type	Server
	Comment	Within the Oracle 9i Application Server, Release 2 suite, the OC4J (Oracle Containers For J2EE) J2EE server features a container for Web services, based on the Apache SOAP 2.2 framework. Oracle 9iAS Web Services support SOAP 1.1, WSDL 1.0 and UDDI 2.0. A UDDI 2.0-compliant registry is bundled, with import/export functions that allow for synchronizing with a third-party, public or private UDDI registry. The framework allows for exposing plain Java classes, EJBs (stateless or stateful session beans, or message- driven beans), and PL/SQL stored procedures as Web services. Highlights of Oracle9iAS Web Services include automatic, on-demand generation of WSDL for hosted services, advanced clustering, load-balancing, and security features, as well as support for both dynamic or static binding for invocation of services.
	Product Name	Oracle 9i Developer Suite (JDeveloper)
	Version	
	Technology	J2EE
	Platforms	Windows NT/ 2000, Linux
	Solution type	IDE
	Comment	Oracle JDeveloper includes comprehensive facilities for developing, assembling, and deploying Web services to Oracle 9iAS. The IDE also caters for automatic generation of a Java Web service client

[26] http://esc.dl.ac.uk/TechReports/WebServices/webServices_doc/node1.html

		JAR archive from the WSDL definition of a service.

5.2 Microsoft Biztalk

Microsoft's Biztalk Server 2002[27] consists of two parts. One part is the business process execution engine, called Biztalk Orchestration Engine. Its user interface is called Biztalk Orchestration Designer. The other part provides the base functionality for integration that does not require business process management. It is called Biztalk Messaging Services and is discussed first in the following.

The base messaging concepts provided by the Biztalk Server are receive functions, channels, and messaging ports. In addition, documents can be validated as well as correlated in order to detect acknowledgements corresponding to already processed documents. Receive functions are the entry point for documents submitted to the Biztalk Server. There are several ways through which documents can be submitted to receive functions. These are the Internet, e-mail attachments, message busses, message queues, adapters. Receive functions receive documents and deliver them to the different available channels. A Component Object Model object called Interchange is also supplied by the Biztalk Server to submit documents directly to a channel without going through receive functions.

Channels modify the document structure if necessary through transformation. For example, if the incoming document structure does not conform to the one expected by the target then transformation is used to transform it. In addition, channels provide functionality for encryption/decryption, digital signatures as well as logging. Incoming documents can be logged in their entirety or only parts of it.

Messaging ports connect the Biztalk Server to schedules (see below), trading partners or applications (through adapters). Messaging ports represent the target or recipient of documents processed by the Biztalk Server. Messaging ports are the outbound interface of the system and they deliver the documents. Different channels can send documents to the same messaging port. This allows a target to receive document from many different sources.

Schedules are the processes implemented by the Biztalk Orchestration Engine and are connected through the messaging ports to channels. A document delivered to a schedule through a messaging port becomes available in this schedule. It processes the document according to the process definition as built by the Biztalk Orchestration Designer. A schedule can receive documents from different messaging ports and can deliver documents to multiple channels. This allows to route message from different sources to different targets.

Schedules consist of process steps that are connected by decisions, loops, actions (like receiving or sending documents), parallel branching and synchronization as well as other process execution elements. A schedule in turn can deliver documents to channels. This means that a schedule can deliver documents to targets. In addition it is possible to deliver

[27] http://www.microsoft.com/biztalk/default.asp

documents through channels to other schedules allowing a decomposition and reuse of schedules.

Schedules separate the interface of a business process from its implementation. Fundamentally, sources and target of documents like message queues or COM objects are separated from the business process through ports. Ports are like input and output parameters of schedules that are bound to the sources and targets of documents (and are different from messaging ports which are connected to channels). This means that a designer specifying a schedule has to specify ports, however, without necessarily knowing how the documents are delivered or provided to ports. Later on, once the schedule with its ports is defined, a developer can link the ports to the various technologies like message queues or COM objects to receive document from or to deliver documents to the schedule. Within a schedule ports can be accessed by actions (specific process steps). An action that is part of the processing flow can either read or write to a port.

Schedules can implement different types of transactions. One type are long-running transactions that externalize intermediate execution states in order to support long life times of schedules like weeks or years. In addition, compensation handling is possible by specifying compensation transactions in case a schedule requires compensation in error cases. In addition to long-running transactions, schedules can implement short-lived transactions, too, when specified as those by a user. In this case the complete schedule is executed as one (database) transaction. Furthermore, sets of process steps within a long-lived schedule can be specified as being short-lived. This allows to group process steps within schedules together as short, atomic transactions. For those compensation transactions can be specified that are executed in case of failures. And finally, timed transactions are available that are running up to a specified time. Once this time is reached and the transaction is not committed successfully compensation is initiated automatically.

In addition, Biztalk Adapters as well as Biztalk Accelerators are available that connect the Biztalk Server to B2B protocols and to back end application systems. The Biztalk Server provides native adapters as well as adapters provided by third parties to the Biztalk Server.

Biztalk Server provides a large array of tools. These are Biztalk Orchestration designer, Biztalk Editor, Biztalk Mapper, Biztalk Messaging Manager, Biztalk Server Administration, Biztalk Document Tracking and Biztalk SEED Wizard.

The following table[28] shows the relationship of this product with the web service technology relevant to the SWWS project.

Vendor	Microsoft	
Web Services	**Product Name**	Biztalk Server 2000
	Version	
	Technology	COM
	Platforms	Windows 2000

[28] http://esc.dl.ac.uk/TechReports/WebServices/webServices_doc/node1.html

	Solution type	Server and IDE
	Comment	Biztalk Server 2000 is not dedicated to Web Services, it's current support of SOAP is limited. However, this product provides a great solution for automating B2B exchanges, using XML and various other formats
	Product Name	Visual Studio .NET
	Version	1
	Technology	Microsoft .NET
	Platforms	Windows 2000
	Solution type	IDE
	Comment	All editions of Visual Studio.NET provide an IDE for creating Web services that can be written in Visual Basic.NET, Visual C++.NET, or Visual C#.NET. The IDE also provides for deploying the service to a Web server and testing it through a Web interface. The WSDL description for the service is automatically generated and accessible through a special URL. Client applications that use the service only need to add a reference to the WSDL description in order to access the service transparently.

5.3 IBM CrossWorlds

IBM acquired CrossWorlds[29] in 2002. With the acquisition IBM inherited CrossWorlds' product with the same name. Technical information is readily available on IBM's web site for CrossWorlds.

IBM CrossWorlds V4.1.1 implements collaborations and business objects as the core concepts. Business objects are the equivalent to the integration concept of events (not business objects). Two classes of business objects are implemented. Generic business objects that are equivalent to the integration concept of business events. And Application specific business objects that are equivalent to the integration concept of clear text events. Business objects implement not only the structure of the data exchanged but also verbs that correspond to the integration concept action. Example verbs are create, retrieve or delete.

Collaborations are used to describe business processes and are equivalent to the integration concept of business processes. Collaborations have the equivalent to process parameters. This is called ports in CrossWorlds. Through ports business objects are given to collaborations and returned from collaborations. Ports are accessible from the internal of collaborations in order to implement the business object flow implementing the business logic in collaborations.

[29] http://www-3.ibm.com/software/integration/cw/

Collaborations can be aggregated into collaboration groups. Collaborations that are part of a collaboration group can pass business objects to each other. This accomplishes not only the reuse of collaborations but also the building of complex processes based on simpler building blocks.

Collaborations have an internal structure if the integration modeler chooses to define it. A collaboration can consist of scenarios. Each scenario implements a different process. For example, instead of one collaboration implementing the processing for create, update and delete a customer a collaboration can be built with three scenarios, one for create, one for update and one for delete customer. This allows to separate the processing within a collaboration depending on the business object and its verb coming in. At runtime, when a create customer business object is received, the appropriate scenario is executed.

Collaborations can be transactional. In the transactional case collaborations implement long-running processes that provide compensation functionality. For each transactional substep of a collaboration that requires compensation a compensation step has to be defined by the integration modeler. At runtime, when an error occurs, the system automatically initiates compensation and the compensation steps are executed in the reverse order of the original collaboration steps. Various data isolation levels are implemented that assure that collaborations execute in various degrees of isolation in order to avoid data inconsistencies through concurrent update.

Maps are provided that are the equivalent to transformation. Maps are used to map generic business objects to application specific business objects and vice versa.

CrossWorlds is a meta data driven system that stores all modeling data in a relational repository. A large set of adapters is provided (in CrossWorlds lingo called connectors) that allows connectivity to a whole range of back end application systems. A connector consists of two parts, a controller and an agent. The controller is the part of a connector that connects to the CrossWorlds hub part of the product. The agent is responsible for connecting to the back end application system. The agent produces the application specific business object in the inbound case or receives the application specific business object in the outbound case. Translation is not an explicit modeling concept in CrossWorlds, however, the functionality is implemented through program language code by so-called data handlers.

In addition to accessing data of back end application systems through connectors CrossWorlds allows the access of itself through a synchronous server access interface. A component like an application server can access CrossWorlds directly without going through a connector. The server access interface allows the external component to send data to CrossWorlds and receive data from CrossWorlds. Through this additional interface homogeneous access can be implemented where the external component does not have to have a connector associated with it.

CrossWorlds provides a range of tools for the integration modeler in order to set up, define and monitor the execution of integration.

The following table[30] shows the relationship of this product with the web service technology relevant to the SWWS project.

Vendor	IBM	
Web Services	**Product Name**	Websphere Application Server
	Version	4.0
	Technology	J2EE
	Platforms	Windows NT/ 2000, AIX (all Java-enabled platforms)
	Solution type	Server
	Comment	WebSphere Application Server supports SOAP 1.1 (through Apache SOAP 2.2), WSDL 1.0, and provides an API for UDDI publishing and searching (UDDI4J). Being actively involved in Web Service- related standards (SOAP, WSDL, UDDI), IBM is rolling out a complete suite for building e-business applications, and embedding and using Web Services.
	Product Name	WebSphere Studio Application Developer
	Version	4.0
	Technology	J2EE
	Platforms	Windows 98/Me/ NT/2000/XP, Linux
	Solution type	IDE
	Comment	WebSphere Studio Application Developer replaces Visual Age for Java 4.0 for J2EE application development in IBM's product portfolio. Based on the Eclipse open-source development platform.The Wizards for building, testing, and deploying Web services Application Server: Web services can be created out of EJBs, DB2 XML Extender calls and stored procedures, SQL queries, and it is also possible to integrate existing services from their WSDL description. The Web Services Client Wizard provides features for automatically generating proxy client code to access Web services. Testing and deployment features allow for testing services running either locally or remotely, and for deploying them into the WebSphere Application Server or Tomcat test environments. Finally, it is possible to publish Web services to a UDDI registry, and to browse through

[30] http://esc.dl.ac.uk/TechReports/WebServices/webServices_doc/node1.html

		UDDI to locate existing Web services for integration.

5.4 BEA WebLogic Integration 7.0

BEA's WebLogic Integration 7.0[31] is a workflow-based integration environment. Core of the architecture is a workflow system that allows to define processes (also called workflows). Three areas of functionality, namely B2B integration functionality, application integration functionality and data integration functionality are added to the workflow system through a plug-in framework that allows the workflow system to communicate with the implementation of these three areas of functionality by means of workflow steps. The different components are discussed in turn in the following.

The workflow management system provides user interface tools for the design, monitoring and runtime interactions (like user interactions) with the system. In addition, application programming interfaces are provided for configuration clients, design clients, runtime management clients and monitoring clients. These APIs support the access outside the provided user interface tools. The workflow system supports the definition of flowcharts (also called process flow) that use predefined modeling elements like start, decision or join in order to define sequences of workflow steps. Defined flowcharts are stored as templates ready for instantiation at runtime. Instantiation of workflow instances can happen through several means: explicit call by an application or another workflow, manual invocation, triggered by the arrival of a XML instance message or started automatically through a timer. At runtime data are represented as XML instances. Interaction with workflow external components during workflow execution is achieved through either EJB methods or XML instances as messages on JMS queues. The external communication is achieved through actions that are implemented within workflow steps.

The distinction of private and public processes can be made in WebLogic Integration. Private processes are implemented in the workflow system. Public processes (also called collaborative processes) are implemented in the B2B integration plug-in.

The B2B integration component supports several concepts for B2B integration. These are conversations, trading partner configurations, business protocols, collaboration agreements, and security. Collaborations are exchanges of XML and non-XML messages. The message transmission is secured and the sequence of business messages is supervised by conversations. Conversations are implemented through collaborative or public processes. A conversation describes a processes for each interacting trading partner. Each partner has a conversation role associated (like buyer or shipper) with it and this relates the partner to a specific process of the conversation. Through this it is defined which trading partner executes which public process and therefore exhibits a specific message exchange behavior.

Trading partner configurations define individual trading partners. For example, a unique name is provided as well as the connectivity information like network addresses. A trading partner configuration also refers to the B2B protocols a trading partner supports for integration.

[31] http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/integrate

Some business protocols (aka B2B protocols) are supported by WebLogic Integration out of the box. Amongst them are RosettaNet, ebXML and EDI.

Collaboration agreements are necessary to relate all concepts necessary to conduct message exchanges. Conversations, roles, collaboration processes, trading partners and network connectivity are related by collaboration agreements. Only then WebLogic Integration has a complete configuration to allow message exchanges.

Security functionality is provided as required by B2B protocols. This includes a SSL-based platform for conversations, certificate verification, digital signatures, non-repudiation of origin and receipt and data encryption.

The application integration component provides a framework to integrate adapters to back end application systems through the J2EE Connector Architecture standard. Each back end application that needs to be integrated must have an associated adapter installed in WebLogic itself. WebLogic Integration defines the concept of an application view as an additional abstraction on top of adapters. An application view of an adapter represents the data going into an adapter and coming from an adapter as XML instances independent of the particular representation the adapter needs. From an integration viewpoint all adapters take and produce XML instances through this approach.

In addition, an adapter development kit is provided that supports the building of custom adapters that are not provided out of the box by any adapter provider.

The final component is the data integration component. It consists of translation and transformation functionality. Translation translates any format into XML and vice versa thereby allowing legacy data formats to be represented in XML for processing in WebLogic Integration. It is meta data driven and the particular transformation rules are modeled through a user interface. As soon as data are represented in XML format WebLogic Integration can interpret its contents.

Data transformation is provided in form of a transformation tool that transforms XML instances to XML instances. It is based on XSL style sheets that are defined through a graphical tool.

The following table[32] shows the relationship of this product with the web service technology relevant to the SWWS project.

Vendor	BEA	
Web Services	**Product Name**	WebLogic Server
	Version	6.1
	Technology	J2EE
	Platforms	Windows NT/ 2000, Unix (many Java-enabled

[32] http://esc.dl.ac.uk/TechReports/WebServices/webServices_doc/node1.html

		platforms)
	Solution type	Server
	Comment	BEA WebLogic Server features a framework for accessing and exposing Web services, and supports SOAP 1.1 and WSDL 1.0. Note that the framework does not allow for handling SOAP attachments in messages. It's interesting to note that BEA has teamed up with Bowstreet to better support Web Services. Having been slow to move on Web Services, BEA is now catching up, and selecting Bowstreet's product to assemble components built with Weblogic may help things along. However, the two vendors suffer from some feature- overlap, and still have some work to do to achieve smooth integration. BEA recently announced the forthcoming availability of a visual IDE for developing and composing Web services. WebLogic Workshop (codenamed "Cajun") will allow rapid modeling of Web services interacting with EJBs, databases, legacy applications (through JCA adapters), and asynchronous messaging systems.

5.5 Further products

Of course, there are more integration products offered for A2A and B2B integration. Some of them are listed in the following Table 5 [B2BBus1]with a reference to the company's web site for more information. Since technical information like product manuals is not readily available the products are not discussed in detail here.

Integration Product	**Reference**
Cyclone Commerce	http://www.cyclonecommerce.com
EXcelon	http://www.exceloncorp.com/
IONA	http://www.iona.com/
Modulant	http://www.modulant.com/
SeeBeyond	http://www.seebeyond.com/
SUN	http://www.sun.com
TIBCO	http://www.tibco.com/
Vitria	http://www.vitria.com/
WebMethods	http://www.webmethods.com/

Table 5 Additional Integration Products

6 References

B2BBus1	B2B Integration - Concepts and Architecture -, Christoph Bussler
B2BOnt	Borys Omelayenko, Integration of product ontologies for B2B marketplaces: a preview, ACM SIGecom Exchanges, v.2 n.1, p.19-25, Winter, 2001
BMEcat	http://www.bmecat.org/
BPEL4ws	http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/
BPQL1	http://www.bpmi.org/faq.esp
BTP	http://www-106.ibm.com/developerworks/webservices/library/ws-wstx1/?dwzone=webservices
BusPr1	http://www.ebpml.org/
BWSIBM	
Chimaera	
cXMLUG	cXML User's Guide, VERSION 1.2.008, NOVEMBER, 2002, www.cxml.org
ECCom	Evaluation and Characterization of Business-to-Business Integration Systems, Christian Kurz, Ewald Hotop, Günter Haring, Department for Distributed Systems, Institute for Computer Science and Business Informatics, University of Vienna, Austria, Lenaugasse 2/8, A-1080 Wien
ECLASS	http://www.eclass.de/
eCLSp	Electronic Catalog XML (eCX XML), 3.0 Specifications, XML for Catalog Interoperability, November 27, 2002, Document Version 1.0
ECX	http://www.ecx-xml.org/?eBusiness-Standard.com
EindUni	http://tmitwww.tm.tue.nl/research/patterns/
ESBB	Evaluation and Characterization of Business-to-Business Integration Systems, Christian Kurz, Ewald Hotop, Günter Haring, Department for Distributed Systems, Institute for Computer Science and Business Informatics, University of Vienna, Austria, Lenaugasse 2/8, A-1080 Wien
OAebXML	http://www.oasis-open.org/committees/ebxml-cppa/
PIPTA	Partner Interface Process Technical Architecture, Arthur Greef, RosettaNet Chief Architect, 04/30/03
PRMT	Noy, N., Musen, M.: PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: Proceedings of the AAAI-00 Conference, Austin, TX 2000
RNIF	RosettaNet Implementation Framework (RNIF) 2.0
RNIF_R2	RosettaNet Implementation Framework, Release 2.00.00
RNhp	http://www.rosettanet.org/rosettanet/Rooms/DisplayPages/LayoutInitial

Schlenoff	C. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell, J. Lee: The Process Specification Language (PSL): Overview and Version 1.0 Specification, NISTIR 6459, National Institute of Standards and Technology, Gaithersburg, MD (2000). http://www.mel.nist.gov/psl/.
UBL	UBL: The Next, Step for Global, E-Commerce**,** UBL Marketing Subcommittee, Last updated 2 April 2002
UML	UML Activity Diagrams as a Workflow Specification Language, Marlon Dumas and Arthur H.M. ter Hofstede, Cooperative Information Systems Research Centre, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001, Australia fm.dumas, a.terhofstedeg@qut.edu.au
UNSPSCwp	Granada Research, United Nations Standard Products and Services Code, White Paper, Why Coding and Classifying Products is Critical to Success in Electronic Commerce, September 1998, updated October 2001
WoFlo	W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. (PDF, 718 Kb) Distributed and Parallel Databases, 14(3), pages 5-51, July 2003
WSCOOR	http://www-106.ibm.com/developerworks/library/ws-coor/
WSFL	Web Services Flow Language (WSFL 1.0), Prof. Dr. Frank Leymann, Distinguished Engineer Member IBM Academy of Technology, IBM Software Group, May 2001
WSMF	The Web Service Modeling Framework WSMF, Dieter Fensel, Christoph Bussler, report number IR-493, February 2002, vrije Universiteit amsterdam
WSTRANS	http://www-106.ibm.com/developerworks/webservices/library/ws-transpect/?dwzone=webservices
XPDL	Workflow Management Coalition, Workflow Standard, Workflow Process Definition Interface, - XML Process Definition Language, Document Number WFMC-TC-1025, Document Status –1.0 Final Draft, October 25, 2002, Version 1.0, Workflow Management Coalition, 2436 N. Federal Highway #374, Lighthouse Point, Fl 33064, USA