Electronic Commerce of Services

Ph.D. Thesis

Silvia Llorente Viejo

Director: Jaime Delgado Mercé

May 2004

Universitat Pompeu Fabra

A Manuel, por su cariño y comprensión

DL B.26400-2007
ISBN:978-84-690-8313-0

Table of Contents

List of Figures

List of tables

Introduction

1 Introduction

The research work for this "Electronic commerce of services" Ph.D. Thesis has been done in several fields around two main concepts: electronic commerce and services, with the final objective of specifying electronic commerce of services, (e-services, in short). To do so, we have developed and validated a methodology for describing them.

The starting point of this research was the implementation of a system for the provision of legal and administrative services inside the European project *TRials in the Domain of Electronic commerce* (TRADE). To do so, closed and proprietary tools were used, giving as a result a final closed solution, which made difficult to take advantage of the experience for applying it to new services or systems. Nevertheless, we realised that the lessons learnt in the TRADE project were applicable to other fields and with this objective in mind we started further research.

The research work inside this Ph.D. Thesis has been structured in the following parts:

- State of the art
- Contribution
- Conclusion and future lines
- References

The first part of the work presents the state of the art on process description initiatives and metadata schemas initiatives, which make use of the XML language. The selection of these two themes was motivated because we found that we needed to describe the structure of electronic commerce of services and the information associated. An additional requirement we imposed was the use of XML. In order to solve the problem of describing the structure or workflow of electronic commerce of services using XML, we studied several process description initiatives that used XML-derived languages to decide if any of them was suitable for our purposes. We also studied metadata schema initiatives to find out if it was possible to describe information associated to electronic commerce of services by means of any of the schemas studied. Moreover, metadata schemas were also studied in the context of the representation of information associated to documents.

The second part of the work describes the contributions of this work related to the description, definition and implementation of electronic commerce of services, based on the research done and making use of some elements of the initiatives studied and described in the state of the art.

The third part contains the conclusions to the research done and the future research lines that this work opens.

The State of the Art part has the following sections:

- Section 2: Process description initiatives
- Section 3: Metadata schemas initiatives

Section 2 presents several process description initiatives, with the common characteristic of the use of the XML language for providing such descriptions. These descriptions were studied in order to decide if any existing XML-derived language was suitable for the description of the structure (workflow) of a service that had to be provided by electronic means. To be precise, the initiatives studied are:

- Workflow Management Coalition: WfMC
- Web services
- Business process execution languages for web services
- Cross-organizational workflow support in virtual enterprises (Crossflow)

- Electronic business using extensible markup language (ebXML)
- Darpa agent mark-up language – Services (DAML-S) and OWL Web ontology language

Section 3 presents several metadata schemas. They also have the common characteristic that have been described using XML or XML-derived languages. The study of these initiatives was motivated by the presence of metadata inside services that are offered by electronic means. It was also noted that some documents resulting from services offered by electronic means could also have associated metadata. The initiatives presented are:

- IEEE Standard for learning object metadata (LOM)
- Dublin Core metadata initiative
- CEN/ISSS workshop on Dublin Core metadata

The Contribution part has the following sections:

- Section 4: Outline of the contribution
- Section 5: Electronic commerce and e-services
- Section 6: Classification of e-services
- Section 7: Workflow inside electronic commerce
- Section 8: Methodology for the definition of e-services
- Section 9: Validation of the model: E-services implementation

Section 4 briefly presents the contribution of this research work. Section 5 introduces the concept of electronic commerce, focusing on electronic commerce of services, e-services, and comparing it with electronic commerce of products and web services. It also provides a definition of e-service. Section 6 defines which kinds of e-services we have, depending on the different features studied. Section 7 introduces the concept of workflow inside e-services and, in general, in electronic commerce. Section 8 presents the methodology for the definition of e-services, describing each component identified inside the methodology together with the different alternatives we can have for each of them. Finally, section 9 provides a validation of the methodology by describing the implementation of two different kinds of e-services, legal and administrative ones and collaborative editing. These e-services are further developed in Appendix A and B, respectively.

The Conclusions and future lines part has the following sections:

- Section 10: Conclusions
- Section 11: Future lines

Section 10 presents some conclusions extracted from the research work done in this Ph.D. thesis. Section 11 shows some of the future research lines opened from the research done.

The References part has bibliographic references present in this thesis work including the research work publications.

State of the art

2 Process description initiatives

2.1 Workflow Management Coalition: *WfMC*

The Workflow Management Coalition (*WfMC*) [WFMC04a], founded in August 1993, is a non-profit, international organization of workflow vendors, users, analysts and university/research groups.

Its mission is to promote and develop the use of workflow through the establishment of standards for software terminology, interoperability and connectivity between workflow products. Consisting of over 285 members, spread throughout the world, it has quickly become established as the primary standards body for this rapidly expanding software market.

The mission statement of *WfMC* can be summarised as follows:

- Increase the value of customers' investment with workflow technology
- Decrease the risk of using workflow products
- Expand the workflow market through increasing awareness for workflow

2.1.1 *WfMC* Structure

WfMC is divided into three major committees, the Technical Committee, the External Relations Committee and the Steering Committee. Small working groups exist within each committee for the purpose of defining workflow terminology, interoperability and connectivity standards, conformance requirements and for assisting in the communication of this information to the workflow user community.

WfMC membership is open to all interested parties involved in the creation, analysis or deployment of workflow software systems. Membership is governed by a document of understanding, which outlines meeting regulations, voting rights and so on.

2.1.2 *WfMC* Working Groups

The *WfMC* has established a number of working groups, each of them working on a particular area of specification. The working groups are loosely structured around the *Workflow Reference Model* which provides the framework for the *WfMC*'s standards program. The Reference Model identifies the common characteristics of workflow systems and defines 5 discrete functional interfaces through which a workflow management system interacts with its environment: users, computer tools and applications, other software services, etc. Working groups meet individually and also under the umbrella of the Technical Committee, which is responsible for overall technical direction and co-ordination.

Table 1 briefly describes the WfMC working groups, summarising their objectives.

Table 1. *WfMC* working groups, together with their objectives

	WfMC Working Groups	Objectives
Currently grouped	1 - Process Definition Interchange Model & APIs	Definition of a standard interface between process definition and modelling tools and the workflow engine(s).
Currently grouped	2 - Client Application APIs	Definition of APIs for client applications to request services from the workflow engine to control the progression of processes, activities and work-items.
Currently grouped	3 - Application Invocation Interface	A standard interface definition of APIs to allow the workflow engine to invoke a variety of applications, through common agent software.
	4 - Workflow Interoperability	Definition of workflow interoperability models and the corresponding standards to support interworking.
	5 - Administration & Monitoring	The definition of monitoring and control functions.
	6 - OMG	Development of a Workflow Management Facility IDL specification endorsed by both the *WfMC* and the OMG.
	7 - Conformance	To develop the *WfMC*'s policy on product conformance against its specifications and agree an approach to vendor certification.
	8 - Reference Model	Specify a framework for workflow systems, identifying their characteristics, functions and interfaces. Development of standard terminology for workflow systems.

The *WfMC* identified 5 functional interfaces to a workflow service as part of its standardisation programme. There is a relationship among these interfaces, as described in the workflow reference model shown in Figure 1. These interfaces were described by the different working groups inside *WfMC*. The rest of interfaces, from 6 to 8, do not appear in the figure, as they are not part of the workflow reference model. They are related to CORBA compliant workflow engines, how conformance of systems can be checked and terminology that has to be used in *WfMC* documents and discussions, respectively.

In the following subsections these interfaces are briefly explained.

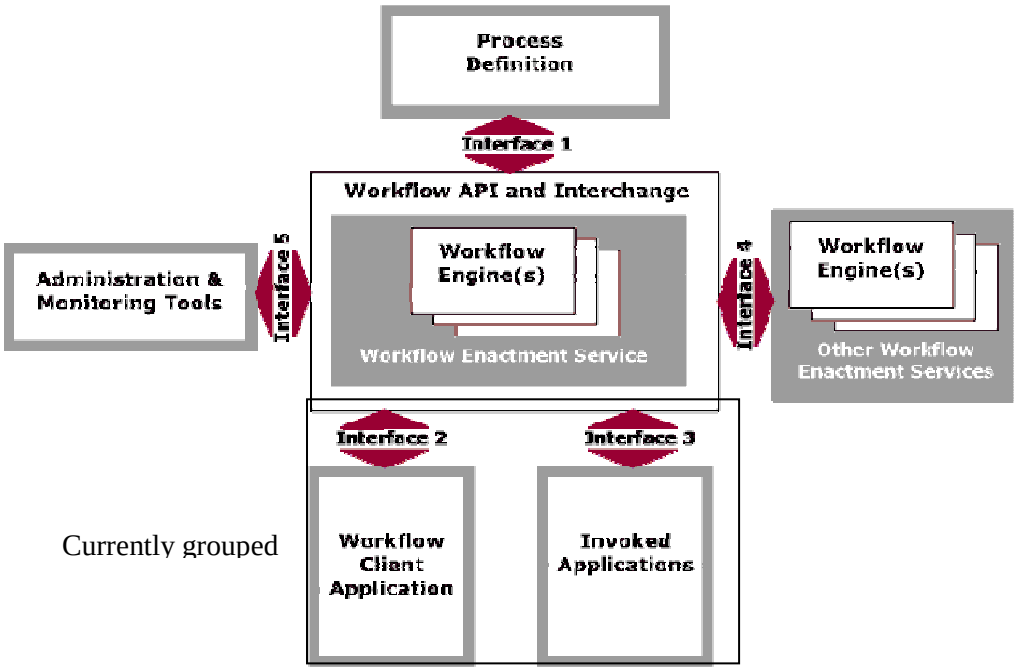


Figure 1. *WfMC* workflow reference model

2.1.3 *WfMC* Interface 1: Process Definition Interchange

Inside the different functional interfaces identified by *WfMC*, interface 1 [WFMC99b] defines process definition import and export among different workflow engines. In this interface it is included a common meta-model for describing the process definition and also a textual grammar for the interchange of process definitions, Workflow Process Definition Language (WPDL) and application programming interfaces (API's) for the manipulation of process definition data.

WPDL was later defined in XML, becoming the XML Process Definition Language (XPDL), described in [WFMC02b]. The XML Schema corresponding to this language can be found in [WFMC02c].

In the next subsections, some meta-models of this language are presented.

2.1.3.1 Meta-Model

The Meta-Model describes the top-level entities contained within a Process Definition, their relationships and attributes (including some which may be defined for simulation or monitoring purposes rather than for enactment). It also defines various conventions for grouping process definitions into related process models and the use of common definition data across a number of different process definitions or models.

The top-level meta-model entities are shown in Figure 2, including their relationship:

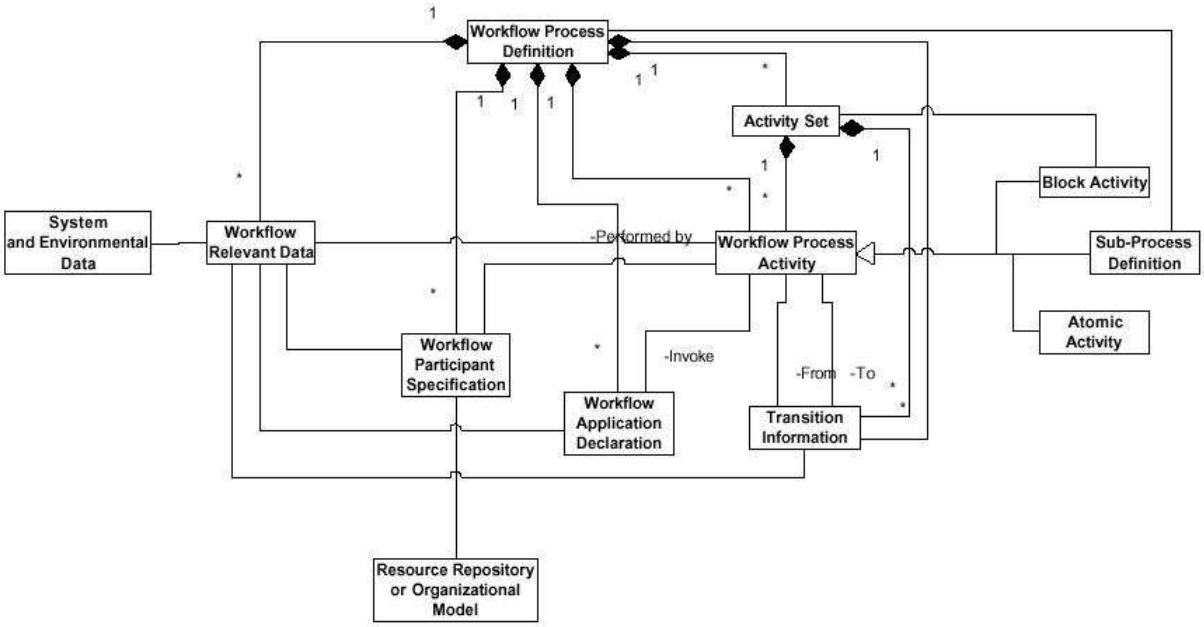


Figure 2. Meta-Model top-level entities

For each of the above entities, there is an associated set of properties, which describe the characteristics of the entity. The complete description of these properties can be found in [WFMC02b].

2.1.3.2 Process and Package Meta-Models

The process model includes various entities whose scope may be wider than a single process definition. In particular the definitions of participants, applications and workflow relevant data may be referenced from a number of process definitions. The meta-model assumes the use of a common process definition repository, associated with the workflow management system, to hold the various entity types comprising the process definition. Within the repository itself and to support the efficient transfer of process definition data to/from the repository, the concept of package is introduced, which acts as a container for the grouping of common data entities from a number of different process definitions, to avoid redefinition within each individual process definition.

The package provides a container to hold a number of common attributes from the workflow process definition entity (author, version, status, etc.). Each process definition contained within the package will automatically inherit any common attributes from the package, unless they are separately re-specified locally within the process definition.

Within a package, the scope of the definitions of some entities is global and these entities can be referenced from all workflow process definitions (and associated activities and transitions) contained within the package. Those entities are:

- Workflow participant specification
- Workflow application declaration
- Workflow relevant data

The package reference allows the use within the package or its contained objects of references to top-level entities in the referenced external package:

- Process identifiers for subflow reference

- Workflow participant specifications
- Workflow application declarations

Workflow relevant data naming must be unique within a package; where such data is passed between processes as parameters. Responsibility rests with process designers / administrators to ensure consistent name / data type usage within process definitions / models to support subflow operations (including any required remote process interoperability).

Figure 3 shows the process meta-model identifying the basic set of entities and attributes for the exchange of process definitions. For a Process Definition the following entities must be defined, either explicitly at the level of the process definition, or by inheritance directly or via cross reference from a surrounding package:

- Workflow Process Activity
- Transition Information
- Workflow Participant Specification
- Workflow Application Declaration
- Workflow Relevant Data

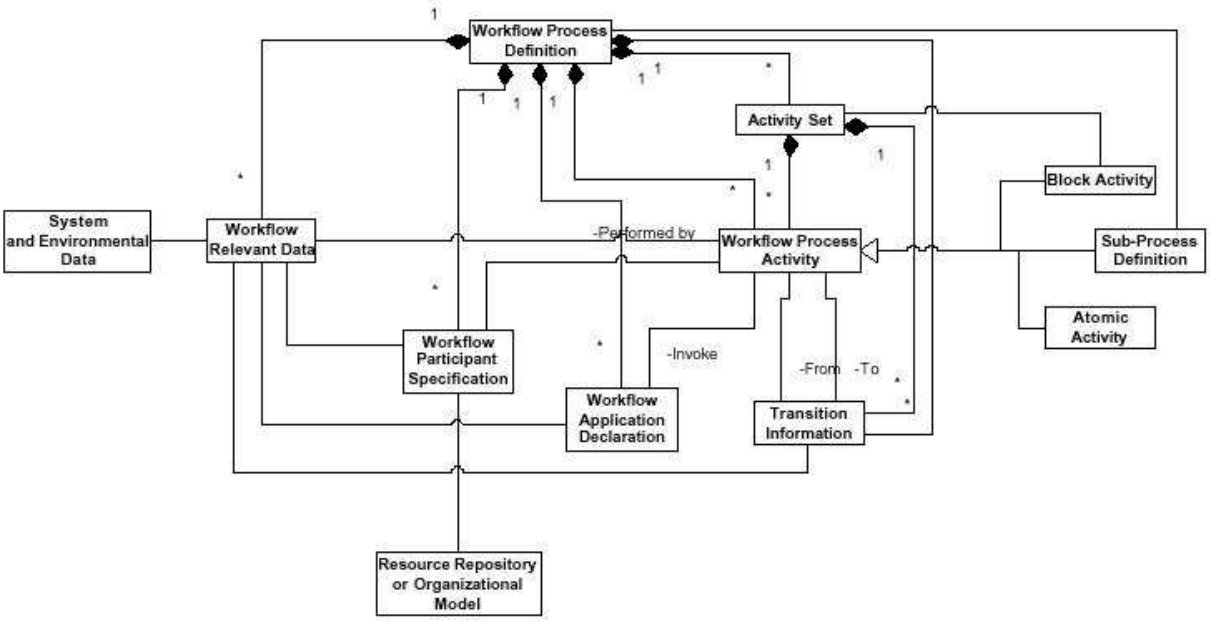


Figure 3. Workflow process definition meta-model

For each of the above entities, there is an associated set of properties, which describe the characteristics of the entity. The complete description of these properties can be found in [WFMC02b].

Multiple process definitions are bound together in a model definition. The Package acts as a container for grouping together a number of individual process definitions and associated entity data, which is applicable to all the contained process definitions (and hence requires definition only once). Figure 4 shows the Package meta-model, which contains the following entity types:

- Workflow Process Definition
- Workflow Participant Specification
- Workflow Application Declaration
- Workflow Relevant Data

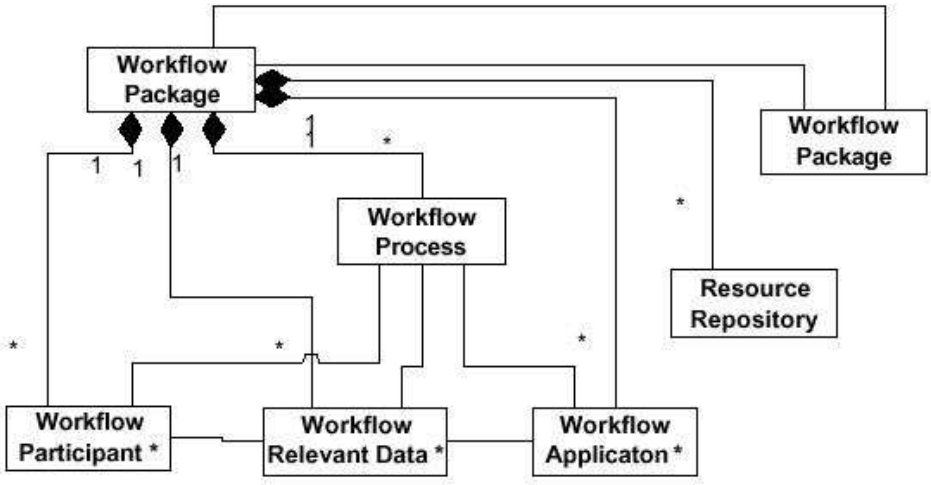


Figure 4. Package definition meta-model

The meta-model for the Package identifies the entities and attributes for the exchange, or storage, of process models. It defines various rules of inheritance to associate an individual process definition with entity definitions for participant specification, application declaration and workflow relevant data, which may be defined at the package level rather than at the level of individual process definitions.

The Package Definition allows the specification of a number of common process definition attributes, which will then apply to all individual process definitions contained within the package. Such attributes may then be omitted from the individual process definitions.

2.1.3.3 XPDL Elements overview

Table 2 gives an overview of major elements defined within XPDL.

- The first row contains attributes and elements common to all major elements. All major elements have the attributes id and name and may contain a Description and Extended Attributes.
- The second row contains specific properties of the respective major element.
- The third group consists of elements that may contain references to other elements.
- The fourth group consists of Documentation and Icon elements. They contain presentation information to be used by the executing engine.
- The fifth group contains information relevant for simulation and process optimisation.

Further elements and predefined attributes may be added to the model to create future conformance levels. The complete description and semantics of all elements can be found in [WFMC02b].

Table 2. Overview of elements

Package	Workflow Process	Activity	Transition	Application	Data Field	Participant
Id	Id	Id	Id	Id	Id	Id
Name	Name	Name	Name	Name	Name	Name
Description	Description	Description	Description	Description	Description	Description
Extended attributes	Extended attributes	Extended attributes	Extended attributes	Extended attributes	Extended attributes	Extended attributes
XPDL version	Creation date	Automation mode			Data type	Participant type
Source vendor id	Version	Split				
Creation date	Author	Join				
Version	Codepage	Priority				
Author	Country key	Limit				
Codepage	Publication status	Start mode				
Country key	Priority	Finish mode				
Publication status	Limit	Deadline				
Conformance class	Valid from date					
Priority unit	Valid to date					
Responsible	Parameters	Performer	Condition			
External package	Responsible	Tool	From	Parameters	Initial value	
		Subflow	To			
		ActivitySet				
		Actual parameter				
Documentation	Documentation	Documentation				
Icon	Icon	Icon				
Cost unit	Duration unit	Cost				
	Duration	Duration				
	Waiting time	Waiting time				
	Working time	Working time				

2.1.4 *WfMC* Interfaces 2 & 3: Workflow Management Application Programming Interface Specification

The purpose of this specification is the definition of standard workflow management Application Programming Interfaces (API), which can be supported by workflow management (WFM) products. These API calls provide for a consistent method of access to WFM functions in cross-product WFM Engines. The API set is named Workflow Application Programming Interfaces (WAPI).

In the API specifications there are guidelines of the *WfMC* for building workflow-enabled applications (Interfaces 1, 2 and 3 in the Workflow Reference Model).

The support of these interfaces in WFM products allows the implementation of front-end applications, which need to access WFM Engine functions (Workflow services). Implementation of these API calls are also intended to allow the workflow applications to be adjusted to operate with different WFM Engines using this common API interface.

These API calls should allow a WFM exploiter to have a single end user interface and functions set regardless of the number of WFM products existing in an installation. WAPI calls may be implemented in a number of languages. The WAPI calls are for use at run-time. That is, when processes are executing or are to be executed. They would normally be used by workflow applications (e.g. worklist handlers, cooperating applications) but may also be used by a WFM Engine when it wishes to interact with another WFM product within the context of the API functions.

Through its set of functions, the WAPI provides a set of workflow services that a workflow enactment service provides. The WAPI does not assume any specific user interface, but rather it specifically assumes that the user interface of the workflow enabled application, that uses these services, provides its own user interface, that depends solely on the application development environment facilities where it is implemented.

The WFM Engine functions can broadly be classified in the following areas:

- WAPI Connection Functions
- WAPI Workflow Definition Functions
- WAPI Process Control Functions
- WAPI Activity Control Functions
- WAPI Process Status Functions
- WAPI Activity Status Functions
- WAPI Worklist Functions
- WAPI Administration Functions

More information about this specification can be found in [WFMC98b].

2.1.5 *WfMC* Interface 4: Interoperability

Interoperability issue was first defined by means of an abstract specification [WFMC99c]. In this document it is defined the functionality required to support interoperability between different workflow engines. Workflow product vendors should use this abstract specification to understand the principles of how interoperability between workflow engines are effected using the WfMC Standards. In the abstract specification it is not presented how the standard has to be implemented,

but a series of principles have been defined to indicate workflow engine vendors how to implement interoperable products.

Apart from this abstract specification, several specific transport bindings have been defined. They give the details of how conformant implementations of the abstract specification must work. Interoperability implementations can claim compliance with a specific binding of the abstract specification. At the current moment, the following bindings have been defined:

- Wf-XML 2.0, XML Based Protocol for Run-Time Integration of Process Engines [WFMC03a]
- AWSP, Asynchronous Web Services Protocol [WFMC02a]
- Internet e-mail MIME Binding [WFMC00a]
- Asynchronous HTTP binding of Wf-XML [WFMC00b]

2.1.6 *WfMC* Interface 5: Audit Data Specification

The audit data specification deals with the information that needs to be captured and recorded from the various events occurring during a workflow enactment. In [WFMC98d] it is defined what information is to be gathered and made available for analysis, but not how it is stored. This information is called Common Workflow Audit Data (CWAD).

Again, there is an abstract specification from which it is intended that concrete bindings will be derived to different programming languages, such as SQL, C, or C++. By defining the semantics for this data, a cohesive analysis is possible when working with heterogeneous workflow products.

For the moment, there are no concrete bindings defined in this interface, only the abstract specification [WFMC98d] is available from *WfMC*.

2.1.7 *WfMC* Interface 6: OMG

This interface only consists on a submission made by OMG [WFMC98c] with the objective of defining the requirements for interoperability between different workflow object implementations in a CORBA environment.

2.1.8 *WfMC* Interface 7: Conformance

This interface deals with the way of measuring conformance of implementations brought to the market by workflow product vendors against the intended semantics laid out in the standards and interface/binding specifications published by the WfMC.

The requirements for conformance of workflow products are defined in the following document [WFMC98a].

2.1.9 *WfMC* Interface 8: Reference model

Inside this interface it is defined the terminology and glossary defined by *WfMC* in order to be used in their specifications and discussions. The complete document can be found in [WFMC99a].

2.2 Web Services

Web Services refer to loosely coupled software applications distributed across the Internet and World Wide Web. They are completely self-contained and self-described, unlike the rest of distributed software applications, which are dependent on the technology used for implementing them. They are intended for being used by companies that offer their services over the Internet. Other companies will use these services through the web in a loosely coupled way.

A Web Service is a fully encapsulated, modular unit of application logic that can be found and used by other applications without requiring an intimate knowledge of the inner working of the service. It is based on well-known standards like eXtensible Markup Language (XML) [XML04a] and Hypertext Transfer Protocol (HTTP) [HTTP99a] and can be mixed and matched with other Web Services when needed.

Simple Object Access Protocol (SOAP) [SOAP03a, SOAP03b, SOAP03c], Universal Description, Discovery and Integration (UDDI) [UDDI04a] and Web Services Description Language (WSDL) [WSDL03a, WSDL03b, WSDL03c] are key in the Web Services paradigm. They are independent but used together provide a complete infrastructure for Web Services implementation. We are going to describe them in next sections, but we want to briefly explain their relationship before.

UDDI provides an open framework for describing, discovering and integrating services over the Internet. It also provides a registry that works as a *yellow pages* service for Web Services. UDDI is accessed from SOAP, a lightweight XML-based protocol used to exchange information in a distributed way, usually following a client-server model. WSDL is an XML-based description language that complements UDDI as it provides a uniform mechanism for describing services and network protocol bindings.

In the next sections we give a more detailed explanation on each of these technologies.

2.2.1 Simple Object Access Protocol (SOAP)

Simple Object Access Protocol (SOAP) [SOAP03a, SOAP03b, SOAP03c] Version 1.2 provides the definition of the XML-based information which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment.

SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information. SOAP is silent on the semantics of any application-specific data it conveys, as it is on issues such as the routing of SOAP messages, reliable data transfer, firewall traversal, etc. However, SOAP provides the framework by which application-specific information may be conveyed in an extensible manner. Also, SOAP provides a full description of the required actions taken by a SOAP node on receiving a SOAP message.

In the next sections, some of the main features of SOAP are described.

2.2.1.1 SOAP message structure

Figure 5 shows an example of a SOAP message. It is structured into a SOAP Envelope (marked with orange text), which contains a SOAP Header (red text) and a SOAP Body (green text).

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
   xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

Figure 5. Example of SOAP message

The SOAP header element is optional. It is an extension mechanism that provides a way to pass information in SOAP messages that is not application payload. Such *control* information includes, for example, passing directives or contextual information related to the processing of the message. This allows a SOAP message to be extended in an application-specific manner. The immediate child elements of the `env:Header` element are called header blocks, and represent a logical grouping of data which can individually be targeted at SOAP nodes that might be encountered in the path of a message from a sender to an ultimate receiver. The SOAP header in Figure 5 contains two elements, a reservation and a passenger. These header blocks have to be processed in the next SOAP intermediary or by the final recipient of the message. This is indicated by the presence of the attribute `env:role` that represents who has to process the message. The other attribute `env:mustUnderstand` means that the header block is mandatory. If a node does not understand it, the processing of the SOAP message must fail.

The SOAP body is the mandatory element within the SOAP `env:Envelope`, which implies that this is where the main end-to-end information conveyed in a SOAP message must be carried. The

`env:Body` element and its associated child elements, itinerary and lodging, are intended for exchange of information between the initial SOAP sender and the SOAP node which assumes the role of the ultimate SOAP receiver in the message path. Therefore, the `env:Body` and its contents are implicitly targeted and are expected to be understood by the ultimate receiver. The means by which a SOAP node assumes such a role is not defined by the SOAP specification and is determined as a part of the overall application semantics and associated message flow.

The SOAP Message shown in Figure 5 can be expressed as a set of blocks, as shown in Figure 6.

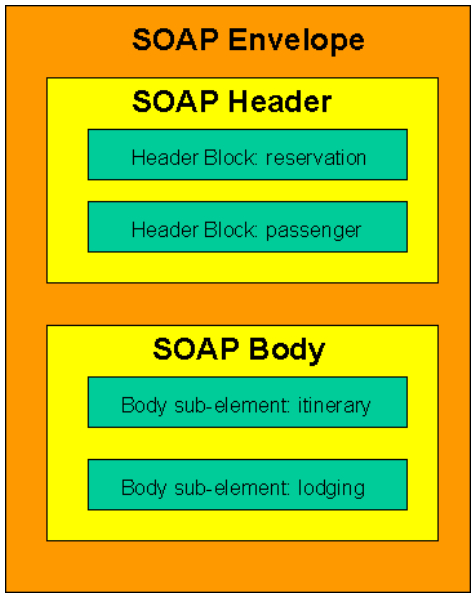


Figure 6. SOAP message blocks

2.2.1.2 SOAP message exchange

SOAP Version 1.2 is a simple messaging framework for transferring information specified in the form of an XML document between an initial SOAP sender and an ultimate SOAP receiver. The more interesting scenarios typically involve multiple message exchanges between these two nodes. The simplest such exchange is a request-response pattern. Some early uses emphasized the use of this pattern as means for conveying remote procedure calls (RPC), but it is important to note that not all SOAP request-response exchanges can or need to be modelled as RPCs. The latter is used when there is a need to model a certain programmatic behaviour, with the exchanged messages conforming to a pre-defined description of the remote call and its return.

A much larger set of usage scenarios than that covered by the request-response pattern can be modelled simply as XML-based content exchanged in SOAP messages to form a back-and-forth *conversation*, where the semantics are at the level of the sending and receiving applications.

Nevertheless, one of the most relevant message exchanges in SOAP is the one that models RPC calls. One of the design goals of SOAP Version 1.2 is to encapsulate remote procedure call functionality using the extensibility and flexibility of XML.

To invoke a SOAP RPC, the following information is needed:

- The address of the target SOAP node: It contains or supports the target of the RPC. It assumes the role of the ultimate SOAP receiver. The way that this node identifies the target method or procedure depends on the underlying protocol binding.
- The procedure or method name.

- The identities and values of any arguments to be passed to the procedure or method together with any output parameters and return value. A clear separation of the arguments used to identify the Web resource which is the actual target for the RPC, as contrasted with those that convey data or control information used for processing the call by the target resource.
- The message exchange pattern, which will be employed to convey the RPC.
- Optionally, data, which may be carried as a part of SOAP header blocks.

2.2.1.3 SOAP processing model

The SOAP processing model describes the actions taken by a SOAP node on receiving a SOAP message. There is a requirement for the node to analyse those parts of a message that are SOAP-specific, namely those elements in the SOAP `env` namespace. Such elements are the envelope itself, the header element and the body element. A first step is, of course, the overall check that the SOAP message is syntactically correct. The processing of a SOAP message depends on the header attributes present and their values.

A SOAP node is required to process a header block if it assumes the role identified by the value of the `env:role` attribute. There are three standardised values for the role attribute: none (no SOAP node has to process the message), next (action to be taken by the next node in the message path) and ultimateRecipient (action to be taken by the ultimate recipient of the message). An application can define their own values for role attribute, indicating the action that has to take place when a SOAP node receives the message.

After a SOAP node has correctly identified the header blocks (and possibly the body) targeted at itself using the `env:role` attribute, the additional attribute, `env:mustUnderstand`, in the header elements determines further processing actions that have to be taken. In order to ensure that SOAP nodes do not ignore header blocks which are important to the overall purpose of the application, SOAP header blocks also provide for the additional optional attribute, `env:mustUnderstand`, which, if `true`, means that the targeted SOAP node must process the block according to the specification of that block. Such a block is colloquially referred to as a mandatory header block. In fact, processing of the SOAP message must not even start until the node has identified all the mandatory header blocks targeted to him and understood them. Understanding a header means that the node must be prepared to do whatever is described in the specification of that block. If the node is not capable of processing the header block, a SOAP fault must be generated.

SOAP Version 1.2 defines another optional attribute for header blocks, `env:relay`, which indicates if a header block targeted at a SOAP intermediary must be relayed if it is not processed.

If a header block is processed, the SOAP processing rules requires that it be removed from the outbound message. The default behaviour for an unprocessed header block targeted at a role played by a SOAP intermediary is that it must be removed before the message is relayed.

The reason for this choice of default is to lean on the side of safety by ensuring that a SOAP intermediary make no assumptions about the survivability past itself of a header block targeted at a role it assumes, and representing some value-added feature, particularly if it chooses not to process the header block, very likely because it does not understand it. That is because certain header blocks represent hop-by-hop features, and it may not make sense to unknowingly propagate it end-to-end. As an intermediary may not be in a position to make this determination, it was thought that it would be safer if unprocessed header blocks were removed before the message was relayed.

Targeting the header block at the role `next` together with the `env:relay` attribute set to `true` can always serve to ensure that each intermediary has a chance to examine the header, because one of the anticipated uses of the `next` role is which header blocks that carry information are expected to persist along a SOAP message path. Therefore, there is no restriction on the use of the `env:relay` attribute with any role except of course the roles of `none` and `ultimateReceiver`, for which it is meaningless.

2.2.1.4 SOAP message transport

SOAP messages may be exchanged using a variety of underlying protocols, including other application layer protocols. The specification of how SOAP messages may be passed from one SOAP node to another using an underlying protocol is called a SOAP binding. The best-known SOAP binding is the SOAP HTTP binding that we describe next.

HTTP has a well-known connection model and a message exchange pattern. The client identifies the server via a URI, connects to it using the underlying TCP/IP network, issues a HTTP request message and receives a HTTP response message over the same TCP connection. HTTP implicitly correlates its request message with its response message; therefore, an application using this binding can chose to infer a correlation between a SOAP message sent in the body of a HTTP request message and a SOAP message returned in the HTTP response. Similarly, HTTP identifies the server endpoint via a URI, the Request-URI, which can also serve as the identification of a SOAP node at the server.

HTTP allows for multiple intermediaries between the initial client and the origin server identified by the Request-URI, in which case the request/response model is a series of such pairs. Note, however, that HTTP intermediaries are distinct from SOAP intermediaries.

The HTTP binding makes use of the SOAP Web Method feature to allow applications to choose the so-called Web method, restricting it to one of GET or POST, to use over the HTTP message exchange. In addition, it makes use of two message exchange patterns that offer applications two ways of exchanging SOAP messages via HTTP: the use of the HTTP POST method for conveying SOAP messages in the bodies of HTTP request and response messages, and the use of the HTTP GET method in a HTTP request to return a SOAP message in the body of a HTTP response.

The purpose of providing these two types of usages is to accommodate the two interaction paradigms, which are well established on the World Wide Web. The first type of interaction allows for the use of data within the body of a HTTP POST to create or modify the state of a resource identified by the URI to which the HTTP request is destined. The second type of interaction pattern offers the ability to use a HTTP GET request to obtain a representation of a resource without altering its state in any way. In the first case, the SOAP-specific aspect of concern is that the body of the HTTP POST request is a SOAP message, which has to be processed (per the SOAP processing model) as a part of the application-specific processing required for conforming POST semantics. In the second case, the typical usage that is foreseen is the case where the representation of the resource that is being requested is returned not as a HTML, or indeed a generic XML document, but as a SOAP message. That is, the HTTP content type header of the response message identifies it as being of media type `application/soap+xml`. Presumably, there will be publishers of resources on the Web who determine that such resources are best retrieved and made available in the form of SOAP messages.

There are non-normative binding specifications for other protocol bindings, such as email [SOAP02a].

2.2.2 Universal Description, Discovery and Integration

Universal Description Discovery and Integration specification [UDDI04a, UDDI00b, UDDI01a] comes from the joint effort of a group of technology and business leaders to cope with the problem of describing business services offered by them. Their solution is the creation of a service registry architecture that presents a standard way for businesses to build a registry, query other business and enable to registered businesses to interoperate and share information globally in a distributed manner.

With UDDI businesses can:

- Discover each other
- Define how they interact over the Internet
- Share information in a global registry

To ensure that the partners involved in the creation of UDDI use it, they have created an UDDI Business Registry on the web. It is an implementation of the UDDI specification and it enables any company to announce their electronic commerce activities. Each company receives a global unique identifier when it registers and it gives a better knowledge of the company.

Figure 7 shows graphically how version 1 of UDDI works [UDDI00a].

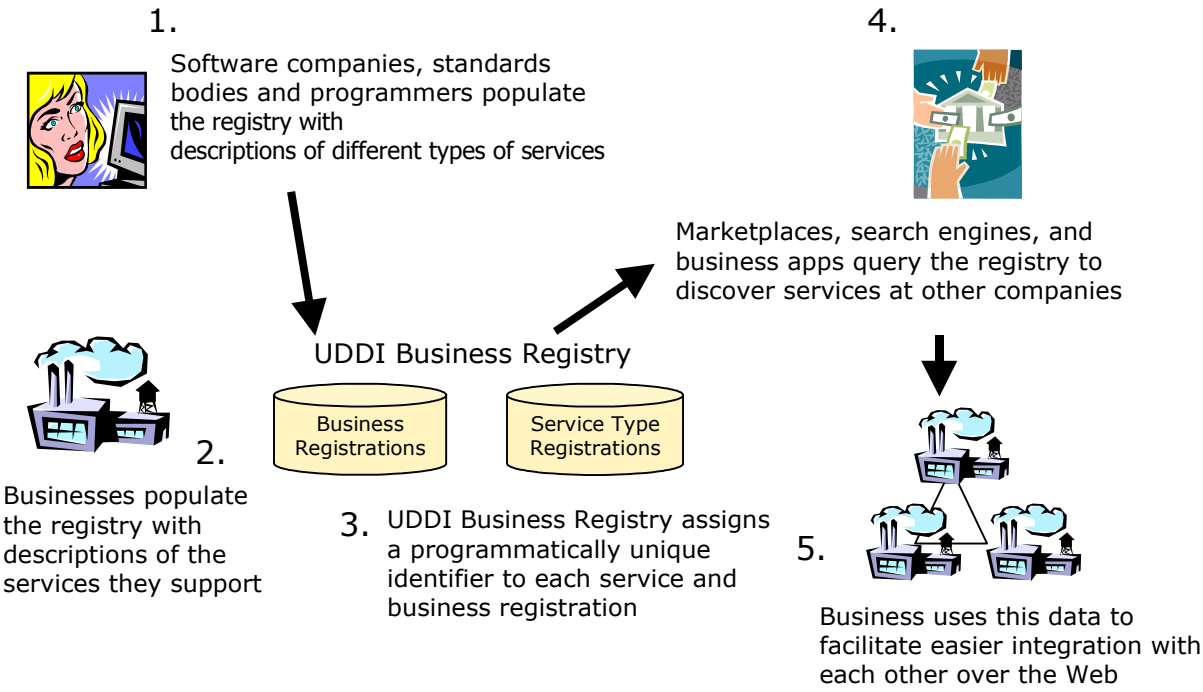


Figure 7. UDDI version 1 way of working

UDDI is currently in version 3 [UDDI02a, UDDI02b] and we briefly describe in next section its main features.

2.2.2.1 UDDI version 3.0

UDDI Version 3 [UDDI02b] builds on the vision of UDDI as a *meta service* for locating web services by enabling robust queries against rich metadata. Expanding on the foundation of Versions 1 and 2, Version 3 offers a specification for building flexible, interoperable XML Web services registries useful in private as well as public deployments.

The information that makes up a UDDI registry consists of instances of four core data structure types, the businessEntity, the businessService, the bindingTemplate and the tModel, together with instances of additional data structure types defined in the UDDI API Schema.

The four core types and their relationships are shown in a simplified diagram in Figure 8 and we briefly explain them next.

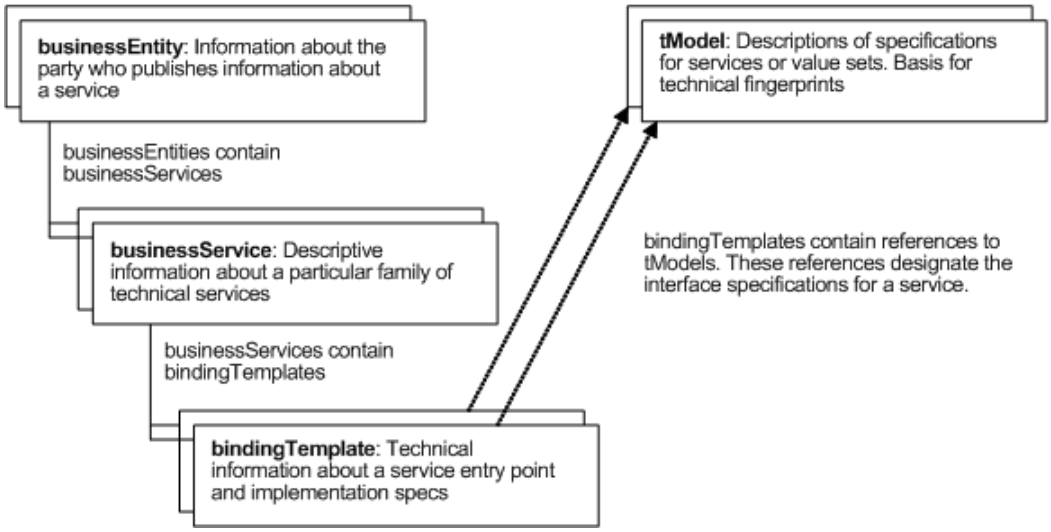


Figure 8. UDDI core data structures

The *businessEntity* structure represents businesses and providers within UDDI. It contains descriptive information about the business or provider and about the services it offers. This would include information such as names and descriptions in multiple languages, contact information and classification information. Service descriptions and technical information are expressed within a businessEntity by contained businessService and bindingTemplate structures.

The *businessService* structure represents a logical grouping of Web services. At the service level, there is still no technical information provided about those services; rather, this structure allows the ability to assemble a set of services under a common rubric. Each businessService is the logical child of a single businessEntity. Each businessService contains descriptive information outlining the purpose of the individual Web services found within it. For example, a businessService structure could contain a set of Purchase Order Web services (submission, confirmation and notification) that are provided by a business.

The *bindingTemplate* structure represents an individual Web service. In contrast with the businessService and businessEntity structures, which are oriented toward auxiliary information about providers and services, a bindingTemplate provides the technical information needed by applications to bind and interact with the Web service being described. It must contain either the access point for a given service or an indirection mechanism that will lead one to the access point.

Each binding Template is the child of a single businessService. A bindingTemplate can be decorated with metadata that enable the discovery of that bindingTemplate, given a set of parameters and criteria.

Technical Models, or *tModels* for short, are used in UDDI to represent unique concepts or constructs. They provide a structure that allows re-use and, thus, standardization within a software framework. The UDDI information model is based on this notion of shared specifications and uses tModels to engender this behaviour. For this reason, tModels exist outside the parent-child containment relationships between the businessEntity, businessService and bindingTemplate structures.

Each distinct specification, transport, protocol or namespace is represented by a tModel. Examples of tModels that enable the interoperability of Web services include those based on Web Service Description Language [WSDL03b] (WSDL), XML Schema Definition [XMLS01a] (XSD), and other documents that outline and specify the contract and behaviour that a Web Service may choose to comply with. To describe a Web service that conforms to a particular set of specifications, transports and protocols, references to the tModels that represent these concepts are placed in the bindingTemplate. In such a way, tModels can be re-used by multiple bindingTemplates. The bindingTemplates that refer to precisely the same set of tModels are said to have the same *technical fingerprint* and are of the same type. In this way, tModels can be used to promote the interoperability between software systems.

The use of tModels is essential to how UDDI represents data and metadata. The UDDI specification defines a set of common tModels that can be used canonically to model information within UDDI. If a concept that is required to model a particular scenario does not exist in a registry, a user should introduce that concept by saving a tModel containing the URL of the relevant overview documents.

One of the most important objectives of UDDI version 3.0 is the provision of mechanisms to support a multi-registry environment, where root and affiliate registries coexist in the aim of making easier the sharing of data among UDDI registries. The need for a root registry comes from the prevention of identifier key collisions among different registries. In this way, the root registry will establish safe key spaces for each registry. By relying on a common root registry as an arbitrator of key spaces, affiliate registries can share data with both the root registry and among one another with the knowledge that a given partition is unique. Note that it is still the responsibility of each registry, both a root registry and its affiliates, to insure the data integrity and uniqueness of the keys within its custody.

An important example of a root registry is the UDDI Business Registry (UBR), which has a set of policies in place to generate unique uuidKeys as well as to validate domainKeys through signatures that correlate with Domain Name Service (DNS) records. These policies insure the uniqueness of both domainKeys and uuidKeys within the UDDI Business Registry, and thus the UBR serves as a reasonable root registry for many purposes. In fact, establishing alternate root registries is not recommended, as this would ultimately defeat the goal of publishers being able to share data between multiple registries with an assurance of avoiding a key collision. By acknowledging the UDDI Business Registry as a root, an affiliate registry can establish inter-registry communication policies and procedures with both the UDDI Business Registry and any other registry, which is an affiliate of the UDDI Business Registry.

2.2.3 Web Services Description Language (WSDL)

Web Services Description Language (WSDL) [WSDL03b] provides a model and an XML format for describing Web services. WSDL enables the separation of the description of the abstract functionality offered by a service from concrete details of a service description such as *how* and *where* that functionality is offered.

This specification defines a language for describing the abstract functionality of a service as well as a framework for describing the concrete details of a service description. The WSDL Version 2.0 Part 2: Message Exchange Patterns specification [WSDL03c] defines the sequence and cardinality of abstract messages sent or received by an operation. The WSDL Version 2.0 Part 3: Bindings specification [WSDL03a] defines a language for describing such concrete details for SOAP 1.2 [SOAP03b], HTTP [HTTP99a] and MIME [MIME96a].

In the next sections, the different parts of WSDL are described in more detail.

2.2.3.1 Core Language

WSDL describes a Web service in two fundamental stages: one abstract and one concrete. Within each stage, the description uses a number of constructs to promote reusability of the description and separate independent design concerns.

At an abstract level, WSDL describes a Web service in terms of the messages it sends and receives; messages are described independent of a specific wire format using a type system, typically XML Schema [XMLS01a].

An operation associates a message exchange pattern with one or more messages. A message exchange pattern identifies the sequence and cardinality of messages sent and/or received as well as who they are logically sent to and/or received from. An interface groups together operations without any commitment to transport or wire format.

At a concrete level, a binding specifies transport and wire format details for one or more interfaces. An endpoint associates a network address with a binding. And finally, a service groups together endpoints that implement a common interface.

In [WSDL03b] the conceptual model for WSDL is described as a set of components with properties. Each aspect of a Web service that WSDL can describe has its own property set. For each component it is also described its XML representation.

The components defined inside WSDL are the following:

- Definitions: Container for WSDL components (interfaces, bindings and services) and type system components. It has several properties such as, *interfaces*, *bindings*, *services*, *type definitions* and *element declarations*.
- Interface: Describes sets of messages that a service sends and/or receives, by grouping related messages into operations. An operation is a set of input and output messages and an interface is a set of operations. It has the following properties, *name*, *target namespace*, *extended interfaces*, *style default*, *operations*, *features* and *properties*.
- Interface operation: Describes an operation that a given interface supports. An operation is an interaction with the service consisting of a set of messages exchanged between the service and the other roles involved in the interaction, in particular the service requester. It has the

following properties, *name*, *target namespace*, *message exchange pattern*, *message references*, *fault references*, *style*, *features* and *properties*.

- Message reference: Associates XML element declarations that define the message content for one of the messages participating in an operation. It has the following properties, *message reference*, *direction* and *message*.
- Fault reference: Associates an XML element declaration that defines the fault message contents for a fault that occurs related to a message participating in an operation. It has the following properties, *name*, *message reference*, *direction* and *message*.
- Feature: Describes an abstract piece of functionality typically associated with the exchange of messages between communicating parties. It has the following properties, *name* and *required*.
- Property: Describes the set of possible values for a particular property. A property is typically used to control a feature's behaviour. Properties, and hence property values, can be shared amongst features. It has the following properties, *name*, *required* and *value constraint*.
- Binding: Describes a concrete message format and transmission protocol, which may be used to define an endpoint. Binding components can be used to describe such information in a re-usable manner for any interface or specifically for a given interface. It has the following properties, *name*, *target namespace*, *interface*, *operations*, *features* and *properties*.
- Binding operation: Describes a concrete binding for a particular operation of an interface to a particular concrete message format. A particular operation of an interface is uniquely identified by the target namespace of the interface and the name of the operation within that interface. It has the following properties, *name*, *target namespace*, *message references* and *fault references*.
- Binding message reference: Describes a concrete binding for a particular message participating in an operation to a particular concrete message format. It has the following properties, *message reference* and *direction.*
- Binding fault reference: Describes a concrete binding of a particular fault message of an operation to a particular concrete message format. It has the following properties, *name*, *message reference* and *direction*.
- Service: Describes a set of endpoints at which the single interface of the service is provided. The endpoints thus are in effect alternate places at which the service is provided. It has the following properties, *name*, *target namespace*, *interface* and *endpoints*.
- Endpoint: Defines the particulars of a specific endpoint at which a given service is available. It has the following properties, *name* and *binding*.

2.2.3.2 Message exchange patterns

Web Services Description Language (WSDL) message patterns [WSDL03c] define the sequence and cardinality of abstract messages listed in an operation. Message patterns also define which other nodes send messages to, and receive messages from, the service implementing the operation.

By design, WSDL message patterns abstract out specific message types. Patterns identify placeholders for messages, and placeholders are associated with specific message types by the operation using the pattern.

Unless explicitly stated otherwise, WSDL message patterns also abstract out binding-specific information like timing between messages, whether the pattern is synchronous or asynchronous, and whether the message are sent over a single or multiple channels.
Like interfaces and operations, WSDL message patterns do not exhaustively describe the set of messages exchanged between a service and other nodes; by some prior agreement, another node and/or the service may send other messages that are not described by the pattern. For instance, even though a pattern may define a single message sent from a service to one other node, the Web Service may multicast that message to other nodes.

To maximize reuse, WSDL message patterns identify a minimal contract between other parties and Web Services, and contain only information that is relevant to both the Web Service and another party.

WSDL patterns are described in terms of the WSDL component model, specifically the Message Reference and Fault Reference components.

The message patterns described are the following:

- In-only and Robust in-only: Consist on one message, which has a value of *in* for the direction property. The message is received from some node called *N*. The difference between these two patterns is the fault generation rule they use.
- In-out: Consists on two messages, one with value of *in* for the direction property and another with value of *out* for the direction property. The first message is received from some node called *N* and the second one is sent to node *N*.
- Out-only and Robust out-only: Consist on one message, which has a value of *out* for the direction property. The message is sent to some node called *N*. The difference between these two patterns is the fault generation rule they use.
- Out-in: Consists on two messages, one with value of *out* for the direction property and another with value of *in* for the direction property. The first message is sent to some node called *N* and the second one is received from node *N*.
- Asynchronous out-in: Consists on two messages, one with value of *out* for the direction property and another with value of *in* for the direction property. The second message is optional. The first message is sent to some node called *N* and the second one is received from node *N*.

2.2.3.3 Binding specifications

WSDL Bindings [WSDL03a] defines binding extensions for the following protocols and message formats:

- SOAP Version 1.2 [SOAP03b].
- HTTP/1.1 GET/POST [HTTP99a].
- MIME [MIME96a].

Nevertheless, WSDL bindings continuously evolve as the protocols it uses evolve. In [WSDL03a] many of the sections are marked to be revised even deleted, so we do not want to extend us in explaining something that may change in a few time.

2.3 Business Process Execution Language for Web Services

The specification document [BPEL03b] of the Business Process Execution Language for Web Services (BPEL4WS) defines a notation for specifying business process behaviour based on Web Services.

The Business Process Execution Language for Web Services (BPEL4WS) is currently in its version 1.1, published on 5th May 2003. It is a joint effort of BEA Systems, International Business Machines Corporation (IBM), Microsoft Corporation, SAP AG and Siebel Systems.

Processes in BPEL4WS export and import functionality by using Web Service interfaces exclusively. Business processes can be described in two ways:

- Executable business processes. They model actual behaviour of a participant in a business interaction.
- Abstract processes. Business protocols use process descriptions that specify the mutually visible message exchange behaviour of each of the parties involved in the protocol, without revealing their internal behaviour. The process descriptions for business protocols are called abstract processes.

BPEL4WS provides a language for the formal specification of business processes and business interaction protocols. 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.

2.3.1 Relationship between BPEL4WS and WSDL

BPEL4WS depends on the following XML-based specifications: Web Services Description Language (WSDL) 1.1 [WSDL01a], XML Schema 1.0 [XMLS01a], XPath 1.0 [XPAT99a] and WS-Addressing [BPEL03a].

Among these, WSDL has the most influence on the BPEL4WS language. The BPEL4WS process model is layered on top of the service model defined by WSDL 1.1. At the core of the BPEL4WS process model is the notion of peer-to-peer interaction between services described in WSDL; both the process and its partners are modelled as WSDL services. A business process defines how to coordinate the interactions between a process instance and its partners. In this sense, a BPEL4WS process definition provides and/or uses one or more WSDL services, and provides the description of the behaviour and interactions of a process instance relative to its partners and resources through Web Service interfaces. That is, BPEL4WS defines the message exchange protocols followed by the business process of a specific role in the interaction.

The definition of a BPEL4WS business process also follows the WSDL model of separation between the abstract message contents used by the business process and deployment information (messages and portType versus binding and address information). In particular, a BPEL4WS process represents all partners and interactions with these partners in terms of abstract WSDL interfaces (portTypes and operations); no references are made to the actual services used by a process instance.

However, the abstract part of WSDL does not define the constraints imposed on the communication patterns supported by the concrete bindings. Therefore a BPEL4WS process may

define behaviour relative to a partner service that is not supported by all possible bindings, and it may happen that some bindings are invalid for a BPEL4WS process definition.

A BPEL4WS process is a reusable definition that can be deployed in different ways and in different scenarios, while maintaining uniform application-level behaviour across all of them. The dependency on WS-Addressing is meant to avoid inventing a private BPEL4WS mechanism for web service endpoint references - such references are obviously a very general requirement in the usage of web services.

2.3.2 Core concepts and usage patterns

Business protocol description and executable business process description usage patterns require a common core of process description concepts. In the BPEL4WS specification the core concepts are clearly separated from the extensions required specifically for the two usage patterns: business protocols and executable business processes. The BPEL4WS specification is focused on defining the common core, and adds only the essential extensions required for each usage pattern.

2.3.3 Definition of a business process

This section provides a quick summary of the BPEL4WS syntax. It is only a brief overview; the details of each language construct are described in [BPEL03b].

The basic structure of the language is as follows:

```
<process>

  <partnerLinks/>

  <partners/>

  <variables/>

  <correlationSets/>

  <faultHandlers>
    <catch>
      activity
    </catch>
    <catchAll>
      activity
    </catchAll>
  </faultHandlers>

  <compensationHandler>
    activity
  </compensationHandler>

  <eventHandlers>
    <onMessage>
      activity
    </onMessage>
    <onAlarm>
      activity
    </onAlarm>
  </eventHandlers>

  activity

</process>
```

The top-level attributes are as follows:

- queryLanguage. This attribute specifies the XML query language used for selection of nodes in assignment, property definition, and other uses. The default for this attribute is XPath 1.0, represented by the URI of the XPath 1.0 specification: http://www.w3.org/TR/1999/REC-xpath-19991116.
- expressionLanguage. This attribute specifies the expression language used in the process. The default for this attribute is XPath 1.0, represented by the URI of the XPath 1.0 specification: http://www.w3.org/TR/1999/REC-xpath-19991116.
- suppressJoinFailure. This attribute determines whether the joinFailure fault will be suppressed for all activities in the process. The effect of the attribute at the process level can be overridden by an activity using a different value for the attribute.
- enableInstanceCompensation. This attribute determines whether the process instance as a whole can be compensated by platform-specific means.
- abstractProcess. This attribute specifies whether the process being defined is abstract (rather than executable).

The token *activity* can be any of the following:

- <receive> allows the business process to do a blocking wait for a matching message to arrive.
- <reply> allows the business process to send a message in reply to a message that was received through a <receive>. The combination of a <receive> and a <reply> forms a request-response operation on the WSDL portType for the process.
- <invoke> allows the business process to invoke a one-way or request-response operation on a portType offered by a partner.
- <assign> can be used to update the values of variables with new data. An <assign> construct can contain any number of elementary assignments.
- <throw> generates a fault from inside the business process.
- <terminate> is only available in executable processes. It can be used to immediately terminate the behavior of a business process instance within which the terminate activity is performed. All currently running activities must be terminated as soon as possible without any fault handling or compensation behavior.
- <wait> allows you to wait for a given time period or until a certain time has passed.
- <empty> allows you to insert a *no-op* instruction into a business process.
- <sequence> allows you to define a collection of activities to be performed sequentially in lexical order.
- <switch> allows you to select exactly one branch of activity from a set of choices.
- <while> allows you to indicate that an activity is to be repeated until a certain success criteria has been met.
- <pick> allows you to block and wait for a suitable message to arrive or for a time-out alarm to go off. When one of these triggers occurs, the associated activity is performed and the pick completes.
- <flow> allows you to specify one or more activities to be performed concurrently. Links can be used within concurrent activities to define arbitrary control structures.
- <scope> allows you to define a nested activity with its own associated variables, fault handlers, and compensation handler.

- <compensate> is used to invoke compensation on an inner scope that has already completed normally. This construct can be invoked only from within a fault handler or another compensation handler.

2.3.4 The lifecycle of a business process

The interaction model that is directly supported by WSDL is essentially a stateless client-server model of synchronous or uncorrelated asynchronous interactions. BPEL4WS, builds on WSDL by assuming that all external interactions of the business process occur through Web Service operations. However, BPEL4WS business processes represent stateful long-running interactions in which each interaction has a beginning, defined behaviour during its lifetime and an end.

The creation of a process instance in BPEL4WS is always implicit; activities that receive messages (that is, <receive> and <pick> activities) can be annotated to indicate that the occurrence of that activity causes a new instance of the business process to be created.

To be instantiated, each business process must contain at least one such *start activity*. This must be an initial activity in the sense that there is no basic activity that logically precedes it in the behaviour of the process.

If more than one start activity is enabled concurrently, then all such activities must use at least one correlation set and must use the same correlation sets.

If exactly one start activity is expected to instantiate the process, the use of correlation sets is unconstrained. This includes a pick with multiple onMessage branches; each such branch can use different correlation sets or no correlation sets.

A business process instance is terminated in one of the following ways:

- When the activity that defines the behavior of the process as a whole completes. In this case the termination is normal.
- When a fault reaches the process scope, and is either handled or not handled. In this case the termination is considered abnormal even if the fault is handled and the fault handler does not rethrow any fault. A compensation handler is never installed for a scope that terminates abnormally.
- When a process instance is explicitly terminated by a terminate activity. In this case the termination is abnormal.
- If a compensation handler is specified for the business process as a whole, a business process instance can be compensated after normal completion by platform-specific means.

2.4 Cross-Organizational Workflow Support in Virtual Enterprises (Crossflow)

CrossFlow (Cross-Organizational Workflow Support in Virtual Enterprises) [CROSS04a] was a project in the 4th framework of the European Community, contract number 28635. The project started in September 1998 and terminated in September 2000.

2.4.1 Project summary

The approach of the project was to describe a number of scenarios that were analysed as background to the development of a contract framework. A match-making facility was provided to match contract offers with contract requirements. This helped agreeing on a contract between the requester and provider of a service in a dynamic manner. This contract was then used in setting-up the inter-organisational link. The necessary resources and mechanisms were then put in place for both organisations, on a per-contract basis. In parallel to the above, additional cooperative support services were developed to facilitate the operational management of cross-organisational workflow.

The results of the project are the following:

- A framework for describing contracts. The framework includes a conceptual model for electronic contracts, an XML-based contract specification language, an XSL style sheet for converting XML-based contracts to HTML representation, and contract templates for the application domains used in the CrossFlow project. Contract templates make the process of match-making contract offers and contract requests easier. This framework also enables the match-making of contract offers and contract requests at run-time.
- Tools for setting up the link between the workflow management systems (WfMSs) of the two organisations according to the contract. This entails the allocation of necessary resources in each workflow and the creation of gateways, which control and monitor the interactions, and translate the passing information from an internal to an external form and vice versa.
- Extensions of the current facilities in WfMSs to provide Co-operative Support Services (CSS) such as:
    - Monitoring of outsourced tasks: to provide information about task progress, resources consumed and quality of service.
    - Change control of outsourced tasks: to allow the modification of tasks after their dispatching.
    - Level of control: to allow a service requester control over a task being executed remotely, for example, to facilitate abort, roll back and compensation where necessary.
    - The conclusions were proposed as open standards to standardisation bodies, thus making the CrossFlow standards also available to WfMS vendors.
    - A publicly available demonstration of the deployment of the solution in two specific application areas, namely insurance and logistics.

2.4.2 Project objectives

The project aimed to enable the use of WfMSs in situations where business processes span organisational boundaries. This will allow businesses to take full advantage of the considerable benefits WfMSs bring to many business situations involving the co-operation between different organisations.

A service-providing organisation (provider) offers a service to carry out a task on behalf of the service-requesting organisation (requester). The task carried out by the provider on behalf of the requester is referred to as an out-sourced task. The linking of the requester and provider workflows enables the requester to initiate and to monitor and control the out-sourced task while it is being

carried out, in accordance with the contract established previously. Figure 9 shows a virtual organization where some tasks are outsourced.

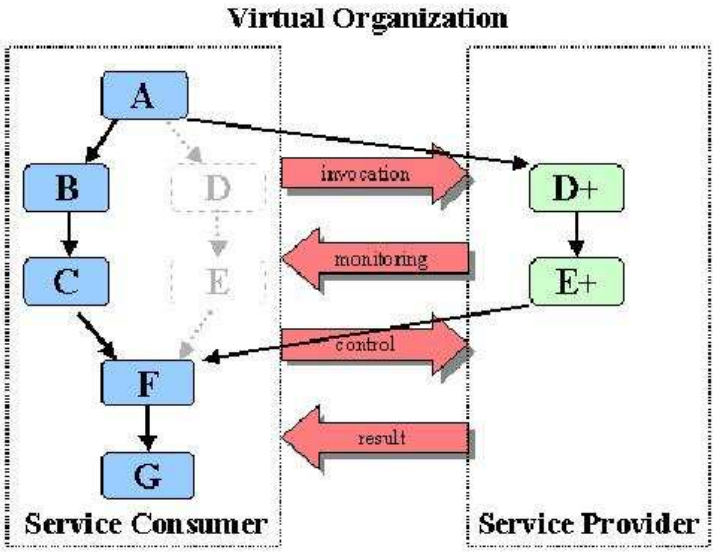


Figure 9. Example of virtual organization

A typical life-cycle of service co-operation, based on linking the workflows of the two organisations, is outlined below:

- Describing and match-making the service: Providers describe their service offerings, which are matched against the requirements specified by the requester, to reach an agreed business relationship.
- Establishing the link: The two organisations agree and sign a contract, which forms the basis for the creation of a new link between two workflows on their respective systems. Depending on the contents of the contract, the link will enable the requester to outsource a task to the provider and thereafter allows it to manage, monitor, verify and audit the task.
- Using the link: Tasks are out-sourced from the requester to the provider using the linked WfMSs of the two organisations, in accordance with the contract. If permitted by the contract, the requester may inquire about the state of an out-sourced task, its progress, and modify or abort it. Additionally, the provider may notify the requester of any problems. The results of tasks are ultimately passed back to the requester.
- Terminating the link: Upon completion of the contract, the link is removed.

The objective of the project was to design, develop and employ a contract framework, infrastructure, support services and tools to support the co-operation life-cycle between the two organisations, enabling the linking of their workflows. The nature of such co-operation between organisations cannot be understood or dealt with properly by concentrating on any single aspect of it. Instead, what is needed is an end-to-end view encompassing the full range of problems that lie on the path between the workflows in WfMSs and the organisations they serve. These problems stem from the different WfMSs and distributed platforms employed, the differences in task models, the complexity of contracts, the crossing of different technical and administrative domain boundaries, and other key issues such as monitoring, quality of service (QoS), transaction management, security and remuneration.

These aspects were addressed by the project, except security and remuneration that were outside the scope of the objectives. In addition to the project objectives stated above, the following requirements were addressed:

- The cost of buying and integrating the proposed infrastructure, support services and tools should be acceptable to small and medium-sized organisations, as well as to large ones.
- It should be possible to generalise the results of this project and apply them to co-operation among more than two organisations.
- The project results should be extendible to work with a variety of WfMSs as well as certain applications, which provide a restricted form of workflow.
- Conclusions drawn from the project should be fed to WfMS vendors and standardisation bodies dealing with this topic.

2.4.3 Crossflow technical approach

CrossFlow aims at providing high-level support for workflows in dynamically formed virtual organizations. High-level support is obtained by abstracting services and offering advanced cooperation support. Virtual organizations are dynamically formed by contract-based matchmaking between service providers and consumers. CrossFlow developments are driven by requirements from real-world scenarios, like the following ones.

- Service abstraction: In virtual organizations, a partner does not require full operational details of other partners. Rather, a well-defined abstraction of their operation should be used to obtain an effective view on both data and processes. As partners in a virtual organization often have different IT platforms, a heterogeneous environment exists. This heterogeneity should be addressed by abstraction of technical details of partners. For both reasons, CrossFlow defines the interaction between organizations not in terms of their workflow systems, but on an abstraction level above these systems.
- Cooperation support: CrossFlow addresses three areas of advanced co-operation support functionality to complement basic workflow interoperability. Quality of Service monitoring allows tracking the progress of outsourced services, both online during service execution and offline to provide aggregate information. Level of Control enactment provides means for high-level cross-organizational transaction management and consumer-controlled process control over outsourced services. Flexible Change Control allows for dynamic changes to execution paths of services during their execution.
- Contracts: The basis for cooperation in virtual organizations is the contract, in which the encapsulated service and cooperation support services can be completely specified. Partially defined contracts are used by service providers to advertise their services and by service consumers to search for services. As such, the contract is the basis for dynamic partnerships.

2.4.4 Crossflow architecture

The CrossFlow architecture supports both contract making and contract (service) enactment. The architecture is based on commercial workflow management system technology, shielded from the CrossFlow technology by an interface layer. In the project, IBM’s MQSeries Workflow (formerly FlowMark) workflow product is used. Figure 10 shows graphically the CrossFlow architecture. Contract making and contract enactment are explained in more detail below.

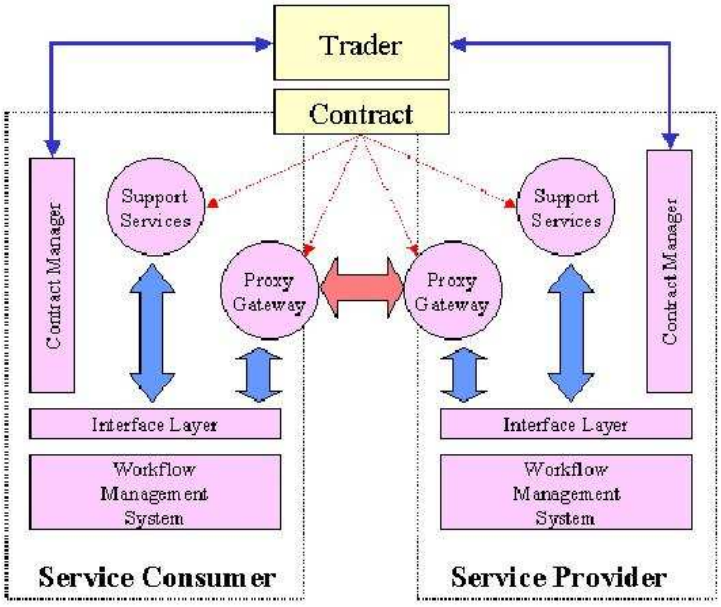


Figure 10. CrossFlow architecture

Contract making: When a service provider wants to advertise a service it can perform on another organization's behalf, it uses its contract manager to send a contract template to a trader. When a service consumer wants to outsource the enactment of a service, it uses a contract template to search for service providers via a trader. When a match between consumer's requirements and provider's offer is found, an electronic contract can be made by filling in the template.

Contract enactment: Based on specifications in the contract, a dynamic contract and service enactment architecture is set up. The symmetrical architecture contains proxy gateways that control all communication and support services for advanced cooperation functionality. After contract completion, the dynamically created modules can be disposed of.

2.5 Electronic Business using eXtensible Markup Language (ebXML)

EbXML (Electronic Business using eXtensible Markup Language) was started in 1999 as an initiative of OASIS [EBXML04a] and the United Nations/ECE agency CEFACT (Centre for Trade Facilitation and Electronic Business) [UNCE04a]. It is a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet. 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.

2.5.1 ebXML Business Process Specification Schema (BPSS)

The ebXML Business Process Specification Schema [EBXML01b] provides a standard framework by which business systems may be configured to support execution of business collaborations consisting of business transactions. It is based upon prior UN/CEFACT work, specifically the metamodel behind the UN/CEFACT Modeling Methodology (UMM) defined in the N090R9.1 specification [EBXML03a].

The specification schema supports the specification of business transactions and the choreography of business transactions into business collaborations. Each business transaction can be implemented using one of many available standard patterns. These patterns determine the actual exchange of business documents and business signals between the partners to achieve the required electronic commerce transaction.

The current version of the specification schema addresses collaborations between two parties (binary collaborations). It is anticipated that a subsequent version will address additional features such as the semantics of economic exchanges and contracts, more complex multi-party choreography and context based content. Figure 11 shows an example of use of ebXML.

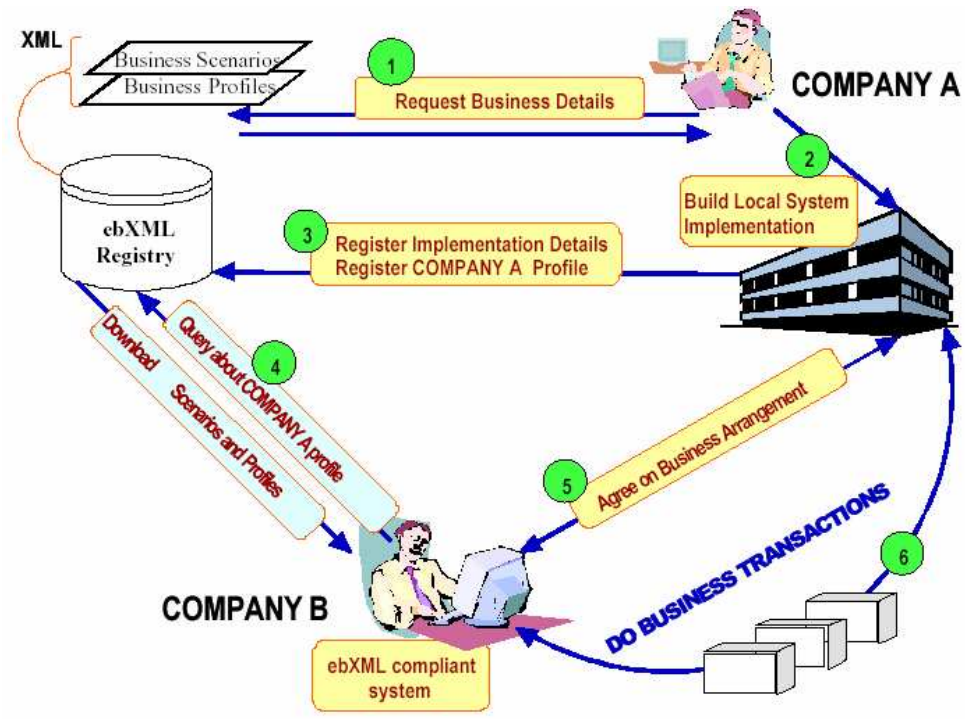


Figure 11. ebXML system overview

In Figure 11, Company A has become aware of an ebXML registry that is accessible on the Internet (step 1). Company A, after reviewing the contents of the ebXML registry, decides to build and deploy its own ebXML compliant application (step 2). Custom software development is not a necessary prerequisite for ebXML participation. ebXML compliant applications and components may also be commercially available.

Company A then submits its own business profile information (including implementation details and reference links) to the ebXML registry (step 3). The business profile submitted to the ebXML registry describes the company's ebXML capabilities and constraints, as well as its supported business scenarios. These business scenarios are XML versions of the business processes and associated information bundles (e.g. a sales tax calculation) in which the company is able to engage. After receiving verification that the format and usage of a business scenario is correct, an acknowledgment is sent to Company A (step 3).

Company B discovers the business scenarios supported by Company A in the ebXML registry (step 4). Company B sends a request to Company A stating that they would like to engage in a business scenario using ebXML (step 5). Company B acquires an ebXML compliant application.

Before engaging in the scenario Company B submits a proposed business arrangement directly to Company A's ebXML compliant software interface. The proposed business arrangement outlines the mutually agreed upon business scenarios and specific agreements. The business arrangement also contains information pertaining to the messaging requirements for transactions to take place, contingency plans, and security-related requirements (step 5). Company A then accepts the business agreement.

Company A and B are now ready to engage in eBusiness using ebXML (step 6).

2.5.2 Business Process Specification Schema (BPSS) related documents

As mentioned above, other documents provide detailed definitions of some of the components of the ebXML Specification Schema and their inter-relationship. They include ebXML specifications on the following topics:

- ebXML Technical Architecture Specification, version 1.04 [EBXML01a]
- ebXML Core Components Dictionary, version 1.04 [EBXML01c]
- ebXML Naming Convention for Core Components, version 1.04 [EBXML01d]
- ebXML Collaboration-Protocol Profile and Agreement Specification, version 1.0 [EBXML01e]
- ebXML Business Process and Business Information Analysis Overview, version 0.7 [EBXML01f]
- ebXML Business Process Analysis Worksheets & Guidelines, version 0.10 [EBXML01g]
- ebXML E-Commerce Patterns, version 0.99 [EBXML01h]
- ebXML Catalog of Common Business Processes, version 0.99 [EBXML01i]
- ebXML Message Service Specification, version 0.99 [EBXML02a]
- UN/CEFACT Modeling Methodology (UMM) [EBXML03a]

We briefly describe them in the rest of the section.

2.5.2.1 ebXML Technical Architecture Specification

This specification is an overview of the ebXML specifications and their inter-relationship. It describes the technical architecture of an ebXML compliant system. On it, ebXML components and functionality are shown. It is also described how an ebXML system must be modelled, including which functional phases involved in the creation and use of an ebXML application. The functional phases of an ebXML system are summarised next.

Implementation phase: Creation of a new ebXML application infrastructure. Figure 12 shows graphically the ebXML implementation phase.

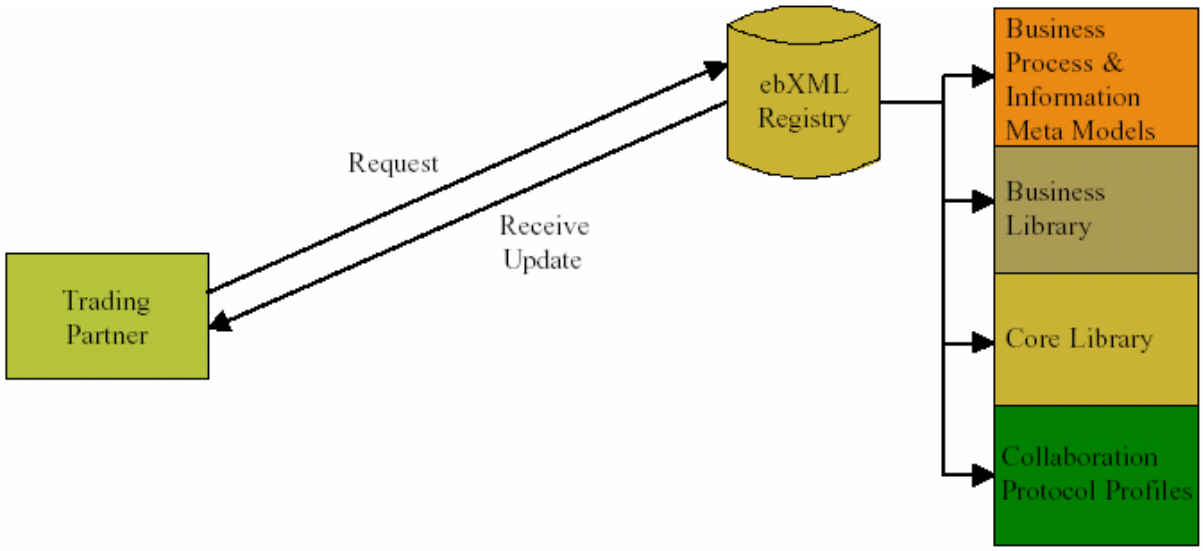


Figure 12. ebXML implementation phase

Discovery and retrieval phase: Involves the discovery and retrieval of an existing ebXML application offered by a trading partner. Figure 13 shows the ebXML discovery and retrieval phase.

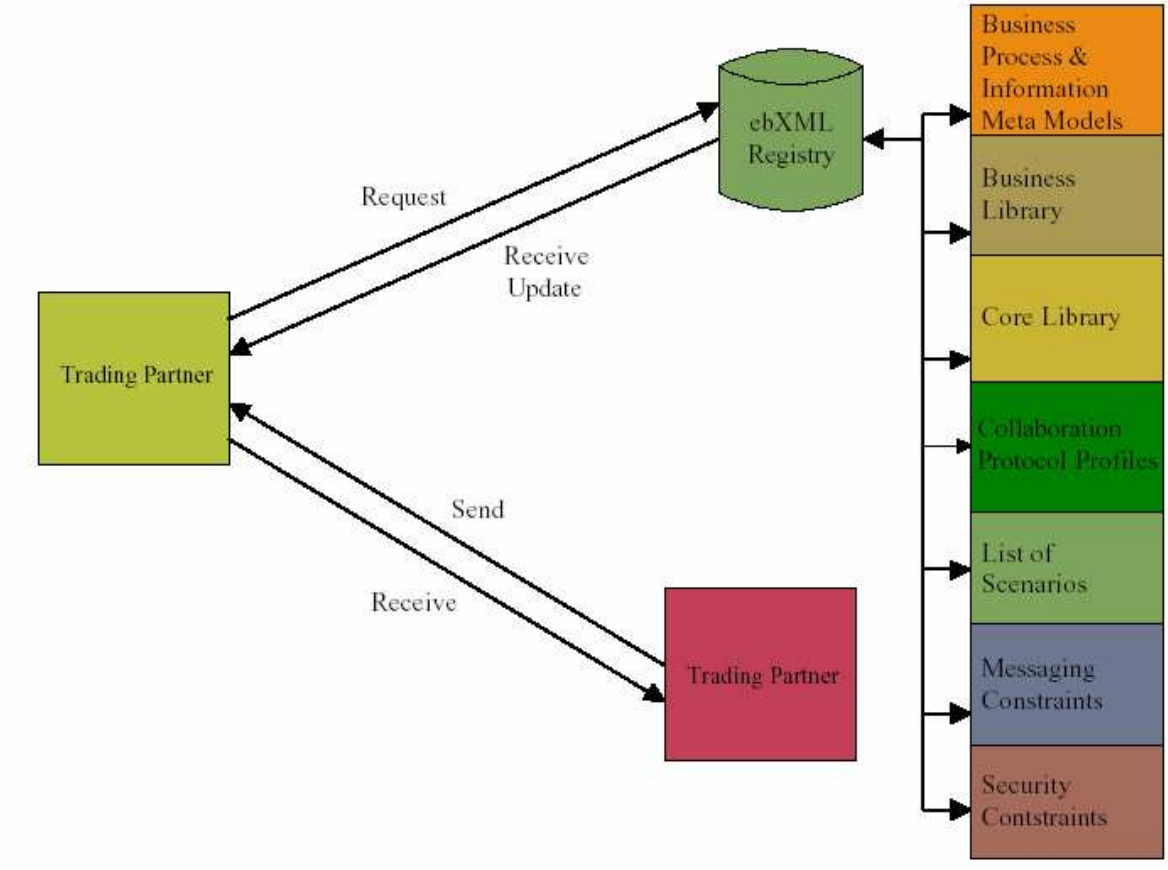


Figure 13. ebXML discovery and retrieval phase

Run time phase: Involves the discovery and retrieval of an existing ebXML application offered by a trading partner. Figure 14 shows the ebXML run time phase.

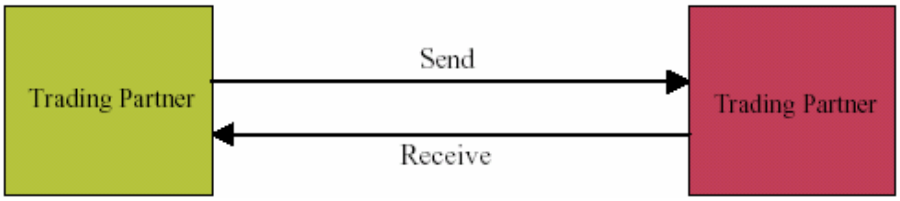


Figure 14. ebXML run time phase

The document also defines the ebXML infrastructure in terms of trading partners information, business process and information modeling, the functionality of core components and library, the functionality of the registry and the message service as well as conformance and security issues.

2.5.2.2 ebXML Core Components Dictionary

The core components dictionary defines a standard set of core aggregate information entities derived from analysis of components submitted by domain discovery groups. Its objective is to define a process, by which information components can be discovered, catalogued and analysed to identify which components are core components. The creation of such a catalogue will enable

interoperability across industries that use electronic commerce. Figure 15 shows a conceptual picture of core components, from business document to final core components.

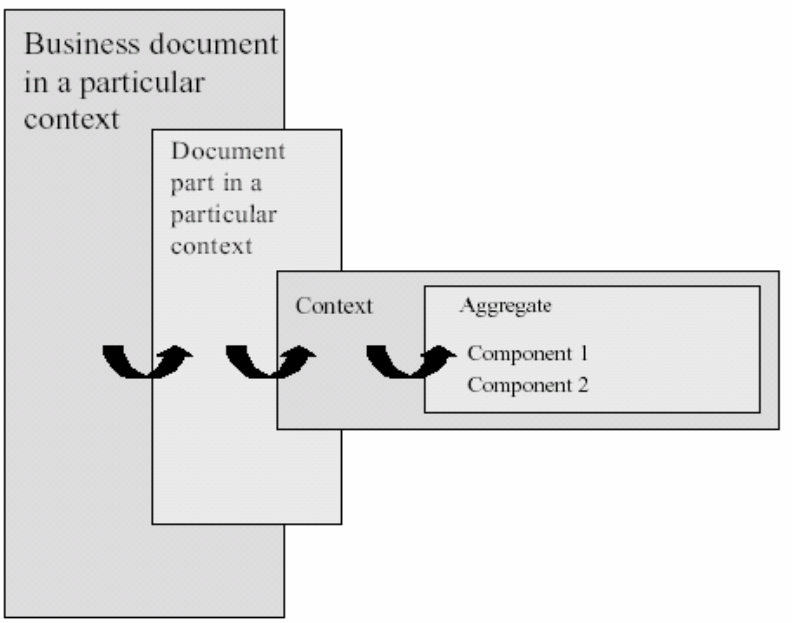


Figure 15. Conceptual picture of core components

A component is a *building block* that contains pieces of business information, which go together because they are about a single concept. An example would be bank account identification, which consists of account number and account name.

Core components are components, which appear in many different circumstances of business information and in many different areas of business. A core component is a common or *general* building block that basically can be used across several business sectors. It is therefore context free.

Re-use is the term given to the use of common core components when they are used for a specific business purpose. The purpose is defined by the combination of contexts in which that business purpose exists. Each context specific re-use of a common component is catalogued under a new business information name *that uses core component X*.

A domain component is specific to an individual industry area and is only used within that domain. It may be re-used by another domain if it is found to be appropriate and adequate for their use, and it then becomes a core or common component.

Components can be built together into aggregates. As described above for components, aggregated components can be common components. These are generic and can be used across several business sectors. They can be re-used for a specific business purpose, defined by a combination of contexts. Each context specific re-use of a common aggregate component is catalogued under a new business informant name *that uses core component X*.

Aggregates and components can be gathered into *document parts*. These are useful assemblies which can individually satisfy a business process's requirement for information, or which may be *sewn together* in a structured way to achieve the same. For example, the structured combination may be to satisfy a business process's need for information presented in a particular way for efficiency of processing.

An individual document part and the *sewn together* parts, come at increasingly domain-specific and context-specific levels. They form documents or partial documents that satisfy a business process or a part of a business process.

Figure 16 illustrates how core components can be built into business documents by explicitly linking components with the ebXML business process worksheets, and the underlying modelling approach.

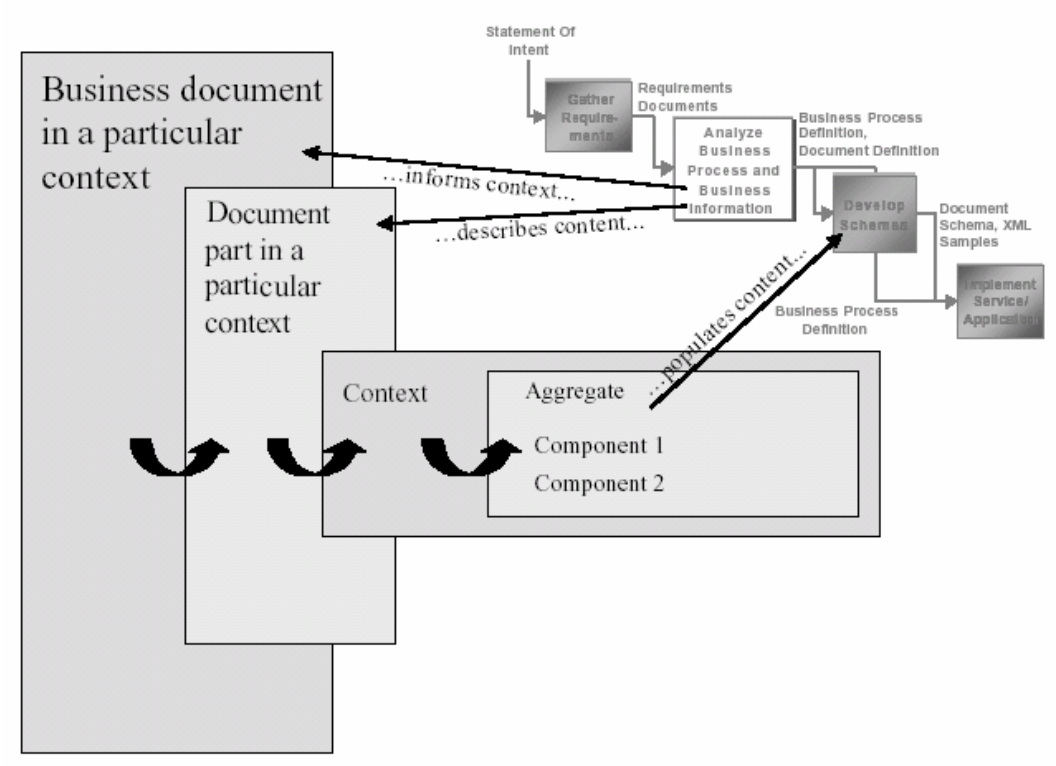


Figure 16. Core components as business documents

2.5.2.3 ebXML Naming Convention for Core Components

This document specifies the rules for naming ebXML core components and business processes.

2.5.2.4 ebXML Collaboration-Protocol Profile and Agreement Specification

The objective of this specification is to ensure interoperability between two parties even though they may procure application software and run-time support software from different vendors.

As defined in the ebXML Business Process Specification Schema [ebXML01b], a business partner is an entity that engages in business transactions with another business partner(s). Each partner's capabilities (both commercial/business and technical) to engage in electronic message exchanges with other partners may be described by a document called a trading-partner profile (TPP). The agreed interactions between two partners may be documented in a document called a trading-partner agreement (TPA). A TPA may be created by computing the intersection of the two Partners' TPPs.

The message-exchange capabilities of a party may be described by a collaboration-protocol profile (CPP) within the TPP. The message-exchange agreement between two parties may be described by a collaboration-protocol agreement (CPA) within the TPA. Included in the CPP and CPA are

details of transport, messaging, security constraints and bindings to a business-process-specification (or, for short, process-specification) document that contains the definition of the interactions between the two parties while engaging in a specified electronic business collaboration.

The CPP defines a party's message-exchange capabilities and the business collaborations that it supports. The CPA defines the way two parties will interact in performing the chosen business collaboration. Both parties shall use identical copies of the CPA to configure their run-time systems. This assures that they are compatibly configured to exchange messages whether or not they have obtained their run-time systems from the same vendor. The configuration process may be automated by means of a suitable tool that reads the CPA and performs the configuration process.

In addition to supporting direct interaction between two parties, this specification may also be used to support interaction between two parties through an intermediary such as a portal or broker. In this initial version of this specification, this may be accomplished by creating a CPA between each party and the intermediary in addition to the CPA between the two parties. The functionality needed for the interaction between a party and the intermediary is described in the CPA between the party and the intermediary. The functionality needed for the interaction between the two parties is described in the CPA between the two parties.

It is an objective of this specification that a CPA shall be capable of being composed by intersecting the respective CPPs of the parties involved. The resulting CPA shall contain only those elements that are in common, or compatible, between the two parties. Variable quantities, such as number of retries of errors, are then negotiated between the two parties. The design of the CPP and CPA schemata facilitates this composition/negotiation process. However, the composition and negotiation processes themselves are outside the scope of this specification.

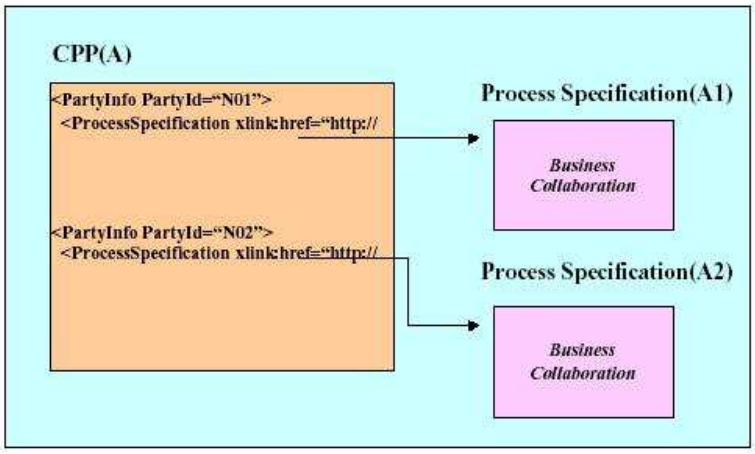


Figure 17. Structure of Collaboration-Protocol Profile & Business Process Specification in an ebXML Registry

It is a further objective of this specification to facilitate migration of both traditional EDI-based applications and other legacy applications to platforms based on the ebXML specifications. In particular, the CPP and CPA are components of the migration of applications based on the X12 838 Trading-Partner Profile to more automated means of setting up business relationships and doing business under them.

This specification defines the markup language vocabulary for creating electronic CPPs and CPAs. CPPs and CPAs are XML documents.

The CPP describes the capabilities of an individual Party. A CPA describes the capabilites that two Parties have agreed to use to perform a particular Business Collaboration. These CPAs define the *information technology terms and conditions* that enable business documents to be electronically interchanged between parties. The information content of a CPA is similar to the information-technology specifications sometimes included in Electronic Data Interchange (EDI) Trading Partner Agreements (TPAs). However, these CPAs are not paper documents. Rather, they are electronic documents that can be processed by computers at the parties' sites in order to set up and then execute the desired business information exchanges. The *legal* terms and conditions of a business agreement are outside the scope of this specification and therefore are not included in the CPP and CPA.

2.5.2.5 ebXML Business Process and Business Information Analysis Overview

The goal of this document is describe the analysis process to give a general understanding of how to conduct business process and documentation definition and identification, within the ebXML framework, and how that relates to the overall development of electronic business relationships with other enterprises.

Its objectives are the provision of an overview of electronic business collaboration, discuss the role and use of business process modelling, describe the analysis process, discuss economic elements and establish the relationship of core components to business processes.

The strength of the ebXML technical architecture is that it provides a framework for electronic business collaboration. The architecture enables businesses to work together to specify business process, discover each other, negotiate collaboration agreements and execute business processes. The significant activities implementing and executing this ebXML electronic business collaboration are shown in Figure 18.

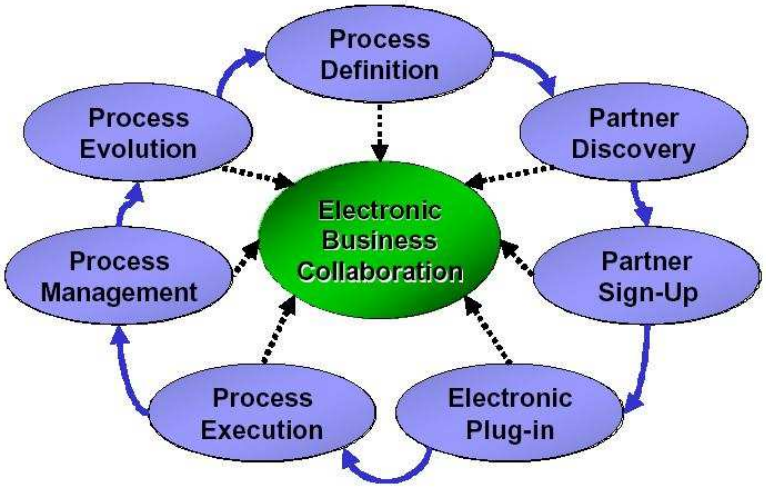


Figure 18. ebXML activities for electronic business collaboration

The overall process starts with process definition, using business process and business document analysis and logically progresses to partner discovery, partner sign-up, electronic plug-in, process execution, process management and process evolution.

Process Definition: Using business process and business document analysis, an enterprise determines and defines which processes will be necessary for electronic commerce. In some cases, a community of trading partners may define the business processes to be used in the community.

These business processes are defined according to a well-known model and described in agreed upon formats.

Partner Discovery: Enterprises identify potential electronic trading partners through a search of company profiles held in ebXML compliant registries.

Partner Sign-up: Trading partners then negotiate agreements that will serve as the terms and conditions of their collaboration.

Electronic Plug-in: The trading partners then configure their electronic interfaces and business services according to their agreements.

Process Execution: Businesses exchange documents and complete commercial transactions in accordance with their agreements and carry out the agreed upon business processes.

Process Management: The business processes defined in the process definition phase and agreed to in the partner sign-up phase are monitored for compliance with trading partner agreements and successful execution.

Process Evolution: Participants in the electronic marketplace will evaluate their existing processes, improve them through process re-engineering and create new processes to meet the needs of the market.

2.5.2.6 ebXML Business Process Analysis Worksheets & Guidelines

ebXML business processes are defined by the information specified in the UMM e-Business Process Metamodel (hereafter referred to as the *Metamodel*). The Metamodel specifies all the information that needs to be captured during the analysis of an electronic commerce based business process within the ebXML framework. ebXML recommends the use of the UN/CEFACT Modeling Methodology (UMM) in conjunction with the Metamodel. The UMM provides the prescriptive process (methodology) to use when analysing and defining a business process.

The ebXML business process worksheets are a set of business process design aids, to be used with the UMM as a reference. It is intended that the worksheets be extensible to meet specific business needs. An ebXML business process, that is defined based on the UMM Metamodel, will sufficiently reflect all the necessary components of a business process and enable its registration and implementation as part of the ebXML compliant electronic trading relationship. The worksheet based approach that provides an easier way of applying the UMM and the UMM Metamodel.

The intent of the worksheets (or a business process editor) is to capture all the bits of information that are required to completely describe a business process so that it can be registered, classified, discovered, reused and completely drive the software.

To develop company business processes for an ebXML compliant electronic trading relationship, use the UMM as a reference guideline plus the ebXML business process worksheet to create the necessary business process models. These are the recommended steps for using the ebXML business process worksheets:

1. A business need or opportunity is identified and defined before using these procedures.

2. A focus project team, usually representing a multifunctional set of experts from IT, business process ownership and business process experts needed to work out the business process using the ebXML business process worksheet.

3. Using the ebXML business process worksheets, the focus project team will be able to develop an ebXML business process specification that can be reviewed and verified by the business. In

addition, all necessary information to populate the ebXML Metamodel will be made available to enable an ebXML trading relationship.

Figure 19 represent the worksheets architectural context, with the use of browsers, worksheets and public and private registries.

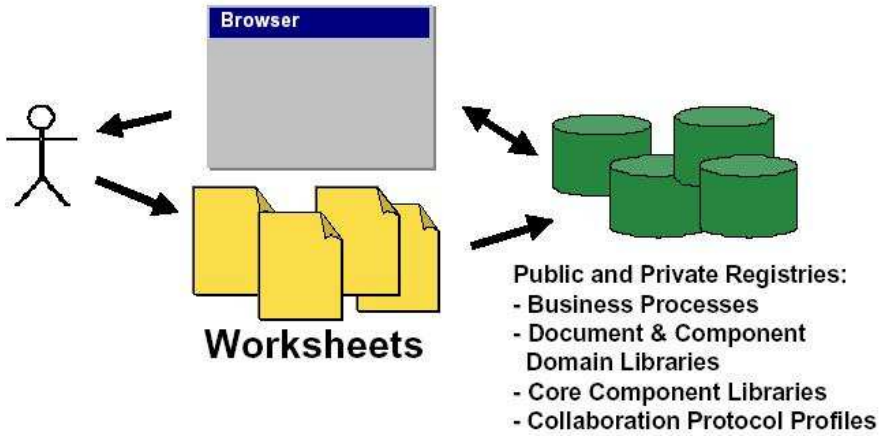


Figure 19. Worksheets architectural context

As stated above, the purpose of this document is to provide worksheets that guide the user through the construction of a UMM compliant specification of their business processes. Figure 20 shows mapping from the worksheets to the high level components of the UMM. Note, the document definition worksheet is currently not included in the set of worksheets.

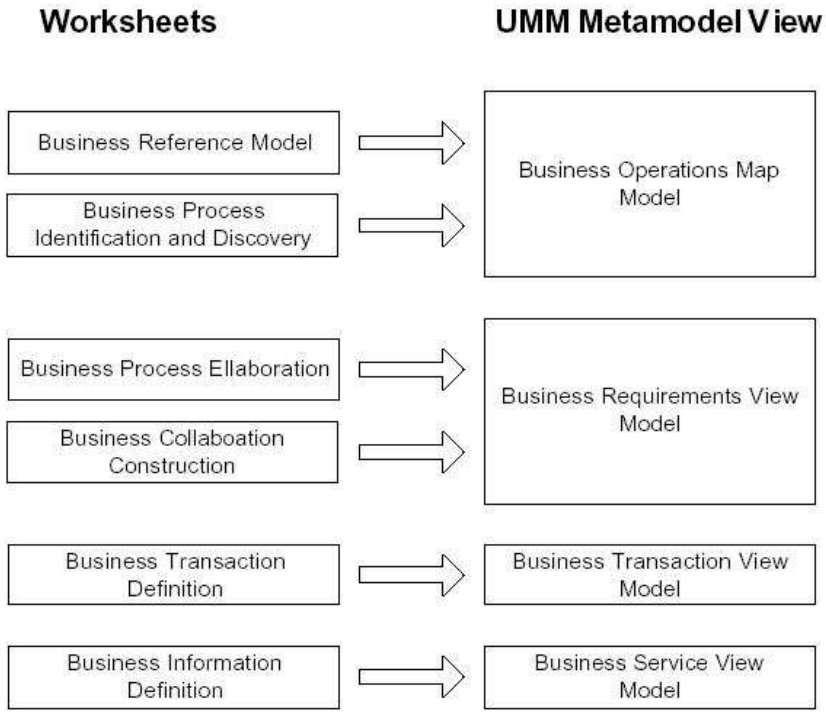


Figure 20. Mapping between worksheets and UMM Metamodel

The expectation is that after the worksheets have been completed, there will be sufficient information to mechanically produce a Metamodel based specification of the modeled business process(es). The worksheets given above are:

Business Reference Model: Use this to define the *frame of reference* of the rest of the worksheets. This provides definitions of terms and, perhaps, canonical business processes.

Business Process Identification and Discovery: Use this to do an inventory of the business processes. This is really just a set of high-level use cases merely to identify the existence of processes and the stakeholders without going into detail.

Business Process Elaboration: These worksheets are used to flesh out the business processes. This identifies the actual actors as well as pre and post conditions for the business process.

Business Collaboration Definition: In these worksheets we define the economic events that take place to fulfill the business process. This is where one defines the system boundaries and the protocols that govern the flow of information.

Business Transaction Definition: These worksheets are more technically oriented than the others. At this stage one defines the actual activities and authorized parties within the organization that initiate these transactions.

Business Information Definition: In these worksheets one defines the contents of the information field widths, data types, descriptions, requirement traceability and, perhaps, the additional context necessary to construct the document from the Core Components subsystem.

2.5.2.7 ebXML E-Commerce Patterns

This document is a supporting document to the ebXML Business Process Specification Schema (BPSS) [ebXML01b] to address common pattern implementation issues and provide examples. The *Simple Contract Formation Pattern* defined here demonstrates a non-normative rule-defined subset of BPSS use for practical contracting purposes. The *Simple Negotiation Pattern* defined demonstrates a non-normative rule-defined subset of BPSS use to allow simple exchanges of *dry run* transactions and collaborations that may result in a collective decision by trading patterns to use them on an enforceable basis. It also may be suitable to automate the negotiation of ebXML CPA terms from CPPs.

BPSS contemplates exchanges of business documents composed into atomic business transactions each between two parties. In order to achieve the desired legal and economic effects of these exchanges, the structure of the business transactions must generate a computable success or failure state for each transaction derived from the application of the BPSS standard, permit the parties to exchange legally binding statements and terms, permit the parties to exchange non-binding statements and terms and permit a logical composition of those exchanges into collaboration patterns that allow agreements about sequences of transactions to be formed.

2.5.2.8 ebXML Catalog of Common Business Processes

This document puts together an initial list of common business process names, generic in nature that can be used across various industries. This includes business processes with cross references across common industry standards. This document also illustrates how to catalog business processes.

A business process consists of a set of business collaborations, which is itself composed of one or more business transactions as defined by the UN/CEFACT Modeling Methodology (UMM) business transaction view (BTV). The behavioral aspects of a business process are defined via the UMM Metamodel.

The primary objective of this catalog is to provide the e-business community with a list of business process names and related information that are independent of any industry specifics. The generic nature of these business processes enables one to reuse them with specific context and business rules within different vertical industries. Common business processes have been grouped under various classifications. Another objective of this catalog is to provide the corresponding references to business documents and business processes defined across various industry standards.

Having a list of common business processes drives the creation of templates for each of these business processes that can be reused across industries. These processes are going to be the basis for discovery and definition of collaboration patterns.

2.5.2.9 ebXML Message Service Specification

This specification is one of a series of specifications realising the vision of creating a single global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML based messages. The set of specifications enable a modular, yet complete electronic business framework.

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 can leverage the advantages of the ebXML infrastructure along with users of emerging technologies.

This specification defines the ebXML Message Service Protocol enabling the secure and reliable exchange of messages between two parties. It includes descriptions of the ebXML message structure used to package payload data for transport between parties and the behavior of the message service handler sending and receiving those messages over a data communications protocol. This specification is independent of both the payload and the communications protocol used.

It is divided into several topics, separated into core funcionality and additional features.

The core functionality comprises the packaging specification, where it is described how to package an ebXML message and its associated parts into a form that can be sent using a communications protocol such as HTTP or SMTP. It also contains the ebXML SOAP envelope extensions, error handling, where the report of errors is described, security issues and SyncReply, to indicate if the replies are synchronous or not.

The additional features comprise the realible messaging function, the message status service, the message order and multi-hop, to describe how messages can be sent through intermediary message service handler nodes.

In its appendices, the XML Schema for the ebXML SOAP Header and Body Extensions, the communications protocol envelope mappings, which describe how to transport ebXML message service compliant messages over HTTP and SMTP and security profiles, can be found.

2.5.2.10 UN/CEFACT Modelling Methodology

United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) has developed the UMM which:

- Has a comprehensive business process and business information metamodel as well as a comprehensive process analysis methodology.
- Provides a methodology and supporting components to capture business process knowledge, independent of the underlying implemented technology
- Helps discover and define a set of reusable process and information descriptions. Patterns help enforce consistent, reproducible results from the UMM-MM across business domains and their business domain experts and analysts
- Implements processes that help insure predictable results from a software project
    - Facilitates the specification of reusable/reproducible process models, in objects and interface-specific object behavior descriptions that are technology and protocol insensitive.
    - Focuses on technology and protocol independent steps of a software engineering process.
- Is a UML profile used to describe the UMM components to specify the business domain specific stereotyping that supports a complete business process and information definition to describe and analyze individual business processes.
- Structures the Business Operational View (BOV) of the Open-edi Reference Model into layers of *views*.

Figure 21 shows the relationship between UMM and other modelling techniques and technologies.

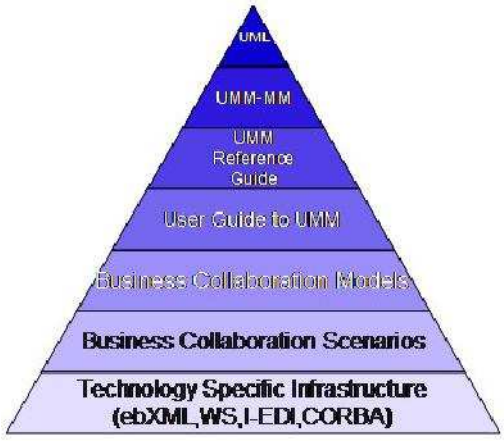


Figure 21. Modelling technologies related to UMM

The UMM can be employed by business analysts to define external and internal business collaboration frameworks. The UMM can be used to define the business collaboration framework implemented between two or more parties. The UMM can be employed from the top-down or bottom-up or using both approaches simultaneously. The end result of an integrated use of the UMM would be a defined Business Collaboration Framework.

2.6 DARPA Agent Mark-up Language – Services (DAML-S) and OWL Web Ontology Language

The DARPA Agent Markup Language (DAML) [DAML04a] program is developing an ontology of services, called OWL-S [DAML03b, OWLW04a, WEBO04a] (formerly DAML-S [DAML04b]), for enabling the access not only to content but also to services on the web. It will allow users and software agents to be able to discover, invoke, compose and monitor web resources offering particular services and having particular properties.

In [DAML03b] we can find the overall structure of the ontology and its three main parts: the service profile for advertising and discovering services; the process model, which gives a detailed description of a service's operation; and the grounding, which provides details on how to interoperate with a service, via messages. We describe OWL-S and their parts in some detail in the following sections.

2.6.1 OWL-S objectives

Services can be simple, or *primitive* in the sense that they invoke only a single web-accessible computer program, sensor, or device that does not rely upon another web service, and there is no ongoing interaction between the user and the service, beyond a simple response. Alternately, services can be complex, composed of multiple primitive services, often requiring an interaction or conversation between the user and the services, so that the user can make choices and provide information conditionally. OWL-S is meant to support both categories of services, but complex services have provided the primary motivations for the features of the language. The following four tasks give an idea of the kinds of tasks OWL-S enables:

- **Automatic web service discovery**: Automatic Web service discovery involves the automatic location of web services that provide a particular service and that adhere to requested constraints. With OWL-S markup of services, the information necessary for web service discovery could be specified as computer-interpretable semantic markup at the service web sites, and a service registry or ontology-enhanced search engine could be used to locate the services automatically. Alternatively, a server could proactively advertise itself in OWL-S with a service registry, also called middle agent, so that requesters can find it when they query the registry.
- **Automatic web service invocation:** Automatic web service invocation involves the automatic execution of an identified web service by a computer program or agent. OWL-S markup of Web services provides a declarative, computer-interpretable API for executing these function calls. A software agent should be able to interpret the markup to understand what input is necessary to the service call, what information will be returned, and how to execute the service automatically.
- **Automatic web service composition and interoperation**: This task involves the automatic selection, composition, and interoperation of web services to perform some task, given a high-level description of an objective. With OWL-S markup of web services, the information necessary to select and compose services will be encoded at the service web sites. Software can be written to manipulate these representations, together with a specification of the objectives of the task, to achieve the task automatically.

- **Automatic web service execution monitoring**: Individual services and, even more, compositions of services will often require some time to execute completely. A user may want to know during this period what the status of his or her request is, or plans may have changed, thus requiring alterations in the actions the software agent takes. For these purposes, it would be good to have the ability to find out where in the process the request is and whether any unanticipated glitches have appeared.

The primary motivation in defining OWL-S has been to support more complex tasks than the ones described above.

2.6.2 An ontology for services

The structuring of the ontology of services is motivated by the need to provide three essential types of knowledge about a service (shown in Figure 22), each characterised by the question it answers:

- What does the service require of the user(s), or other agents, and provide for them? The answer to this question is given in the *profile*. Thus, the class Service *presents* a ServiceProfile.
- How does it work? The answer to this question is given in the *model*. Thus, the class Service is *describedBy* a ServiceModel.
- How is it used? The answer to this question is given in the *grounding*. Thus, the class Service *supports* a ServiceGrounding.

The class Service provides an organizational point of reference for declaring Web services; one instance of Service will exist for each distinct published service. The properties presents, describedBy and supports are properties of Service. The classes ServiceProfile, ServiceModel and ServiceGrounding are the respective ranges of those properties. Each instance of Service will present a descendant class of ServiceProfile, be describedBy a descendant class of ServiceModel and support a descendant class of ServiceGrounding. The details of profiles, models, and groundings may vary widely from one type of service to another, that is, from one descendant class of Service to another. But each of these three classes provides an essential type of information about the service.

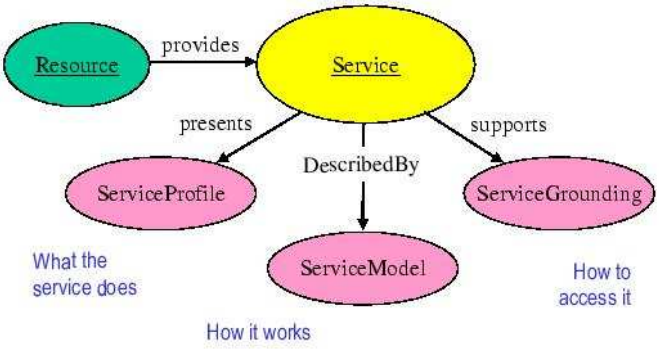


Figure 22. Top level of the service ontology

The service profile tells *what the service does*; that is, it gives the types of information needed by a service-seeking agent to determine whether the service meets its needs. In addition to representing the capabilities of a service, the profile can be used to express the needs of the service-seeking

agent, so that a matchmaker has a convenient dual-purpose representation upon which to base its operations.

The service model tells *how the service works*; that is, it describes what happens when the service is carried out. For nontrivial services (those composed of several steps over time), this description may be used by a service-seeking agent in at least four different ways:

1. To perform a more in-depth analysis of whether the service meets its needs.
2. To compose service descriptions from multiple services to perform a specific task.
3. To coordinate the activities of the different participants during the course of the service enactment.
4. To monitor the execution of the service.

A service grounding (*grounding* for short) specifies the details of how an agent can access a service. Typically a grounding will specify a communication protocol, message formats, and other service-specific details such as port numbers used in contacting the service. In addition, the grounding must specify, for each abstract type specified in the ServiceModel, an unambiguous way of exchanging data elements of that type with the service (that is, the serialization techniques employed).

Generally speaking, the ServiceProfile provides the information needed for an agent to discover a service. Taken together, the ServiceModel and ServiceGrounding objects associated with a service provide enough information for an agent to make use of a service.

The upper ontology for services specifies only two cardinality constraints: a service can be described by at most one service model and a grounding must be associated with exactly one service. The upper ontology deliberately does not specify any minimum cardinality for the properties presents or describedBy. Nor does the upper ontology specify any maximum cardinality for presents or supports.

Finally, it must be noted that while they define one particular upper ontology for profiles, one for service models, and one for groundings, nevertheless OWL-S allows for the construction of alternative approaches in each case. The main aim is not to prescribe a single approach in each of the three areas, but rather to provide default approaches that will be useful for the majority of cases. In the following three sections we discuss the resulting service profile, service model and service grounding in greater detail.

2.6.3 OWL-S Service profile

The Service Profile does not mandate any representation of services; rather, using the OWL subclassing it is possible to create specialised representations of services that can be used as service profiles. OWL-S provides one possible representation through the class Profile. An OWL-S profile describes a service as a function of three basic types of information: what organization provides the service, what function the service computes and a host of features that specify characteristics of the service. The three pieces of information are reviewed in order below.

The provider information consists of contact information that refers to the entity that provides the service.

The functional description of the service is expressed in terms of the transformation produced by the service. Specifically, it specifices the inputs required by the service and the outputs generated; furthermore, since a service may require external conditions to be satisfied and it has the effect of

changing such conditions, the profile describes the preconditions required by the service and the expected effects that result from the execution of the service.

Finally, the profile allows the description of a host of properties that are used to describe features of the service. The first type of information specifies the category of a given service. The second type of information is quality rating of the service: some services may be very good, reliable, and quick to respond; others may be unreliable, sluggish or even malevolent. The last type of information is an unbounded list of service parameters that can contain any type of information. The OWL-S Profile provides a mechanism for representing such parameters; which might include parameters that provide an estimate of the response time, to the geographic availability of a service.

2.6.4 OWL-S Process model

The primary kind of entity in the Process Ontology is, unsurprisingly, a process. OWL-S 1.0 adopts two views of processes. First, a process produces a data transformation from a set of inputs to a set of outputs. Second, a process produces a transition in the world from one state to another. This transition is described by the preconditions and effects of the process.

A process can have any number of inputs, representing the information that is, under some conditions, required for the execution of the process. It can have any number of outputs, the information that the process provides, conditionally, after its execution. Besides inputs and outputs, another important type of parameter specifies the participants in a process. There can be any number of preconditions, which must all hold in order for the process to be invoked. Finally, the process can have any number of effects. Outputs and effects can have conditions associated with them.

As shown in Figure 23, the process model identifies three types of processes: atomic, simple and composite. Each of these is described next.

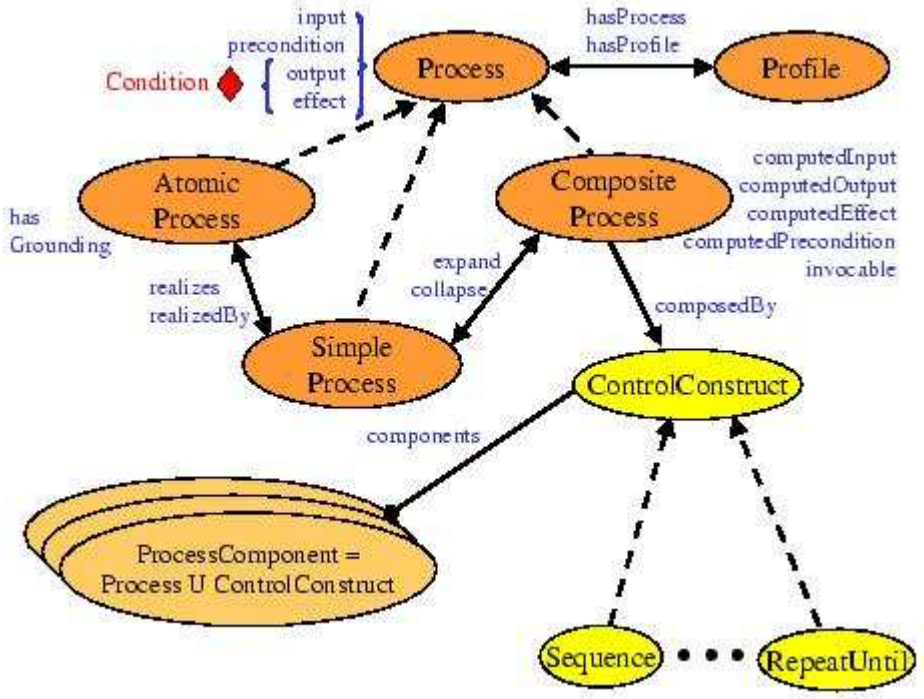


Figure 23. Process structure

Class Process collects the three types of processes: Atomic, Composite and Simple. It has related properties hasParameter, hasInput, hasOutput, hasPrecondition, and hasEffect, which range over classes Parameter, Input, ConditionalOutput, Precondition and ConditionalEffect, respectively. These properties and classes are discussed next.

2.6.4.1 AtomicProcess

The atomic processes are directly invocable (by passing them the appropriate messages). Atomic processes have no subprocesses, and execute in a single step, from the perspective of the service requester. That is, they take an input message, execute, and then return their output message.

2.6.4.2 SimpleProcess

Simple processes are not invocable and are not associated with a grounding, but, like atomic processes, they are conceived of as having single-step executions. Simple processes are used as elements of abstraction; a simple process may be used either to provide a view of some atomic process, or a simplified representation of some composite process. In the former case, the simple process is realizedBy the atomic process; in the latter case, the simple process expandsTo the composite process.

2.6.4.3 CompositeProcess

Composite processes are decomposable into other processes; their decomposition can be specified by using control constructs. Such a decomposition normally shows, among other things, how the various inputs of the process are accepted by particular subprocesses, and how its various outputs are returned by particular subprocesses.

2.6.4.4 Inputs and Outputs

Inputs and outputs specify the data transformation produced by the process. Inputs specify the information that the process requires for its execution. They are similar, in most respects, to arguments of functions in programming languages. The result of the execution of the process is the generation of a set of outputs. Depending on the specification of the Grounding for the process and on the data-flow of the process model, the inputs are either provided by other processes in the process model or by web service clients through message passing. Equivalently, the outputs are either sent to other processes through the data-flow constructs or to other web services.

2.6.4.5 Preconditions and Effects

The execution of a process may also result in changes of the state of the world. The canonical example is the process that charges a credit card. As a result of the execution of the process, a credit card is charged and the money in the account reduced. Note that there is a fundamental difference between effects and outputs. Effects describe conditions in the world, while outputs describe information. In the context of the example, the service may send a notification, or an invoice, that it charged the credit card account. This is just a piece of information that some event happened. The output describes the actual event: that the amount of money in the credit card account has been reduced.

Preconditions specify conditions that should be satisfied for a process to execute correctly. Examples of preconditions are that a credit card should be valid, or that it should not be overdrawn, and so forth.

2.6.4.6 Conditioning Outputs and Effects

OWL-S does not assume that outputs and effects are the same for every execution of the process. Rather, it allows the specification of the set of conditions under which the outputs or the effects may result.

Because of the need for conditions, OWL-S 1.0 defines the classes of ConditionalOuput and ConditionalEffect. Both classes allow a number of conditions to be associated with the outputs and the effects respectively.

2.6.5 OWL-S Service grounding

The grounding of a service specifies the details of how to access the service - details having mainly to do with protocol and message formats, serialization, transport and addressing. A grounding can be thought of as a mapping from an abstract to a concrete specification of those service description elements that are required for interacting with the service - in particular, for our purposes, the inputs and outputs of atomic processes. Note that in OWL-S, both the ServiceProfile and the ServiceModel are thought of as abstract representations; only the ServiceGrounding deals with the concrete level of specification.

OWL-S does not include an abstract construct for explicitly describing messages. Rather, the abstract content of a message is specified, implicitly, by the input or output properties of some atomic process. Thus, atomic processes, in addition to specifying the basic actions from which larger processes are composed, can also be thought of as the communication primitives of a process specification.

Concrete messages, however, are specified explicitly in a grounding. The central function of an OWL-S grounding is to show how the inputs and outputs of an atomic process are to be realized concretely as messages, which carry those inputs and outputs in some specific transmittable format. For defining a concrete message specification they have used Web Services Description Language (WSDL) [WSDL01a], a particular specification language proposal with strong industry backing. The complete description can be found in [DAML03a].

3 Metadata schemas initiatives

Different initiatives in the field of metadata schemas arose through the past years. Their scope is not the same, as each underlying organisation or group of organisations define the best-suited for their purposes. We have selected some of these initiatives because of their relationship with the XML language. They are briefly described in the next sections.

3.1 IEEE Standard for Learning Object Metadata (LOM)

IEEE Standard for Learning Object Metadata is a multi-part standard that specifies learning object metadata.

It consists on four parts:

- 1484.12.1: IEEE Standard for Learning Object Metadata [LOM02a], [LOM04a]
- 1484.12.2: Standard for ISO/IEC 11404 binding for Learning Object Metadata data model [LOM], [LOM04a]
- 1484.12.3: Standard for XML binding for Learning Object Metadata data model [LOM03a], [LOM04a]

The first part specifies a conceptual data schema that defines the structure of a metadata instance for a learning object. For this standard, a learning object is defined as any entity — digital or non-digital —that may be used for learning, education or training.

The second part specifies a binding from ISO/IEC 11404:1996 (Language Independent Datatypes) [ISO96a] to the IEEE 1484.12.1 data model. The purpose of this standard is to provide precise data model semantics, as permitted by the 11404 notation. The 11404 notation may be useful for bindings to programming languages and other systems.

The third part specifies an eXtensible Markup Language (XML) binding of the learning object metadata (LOM) data model defined in IEEE 1484.12.1–2002 Standard for Learning Object Metadata.

In each case, an implementation conforming the second or third part of the standard must conform to the first one, 1484.12.1.

3.1.1 IEEE Standard for Learning Object Metadata (LOM)

This schema is part of the Learning Object Metadata Standard and defines a conceptual data schema that describes the structure of a metadata instance for a learning object.

A learning object is defined as any entity -digital or non-digital- that may be used for learning, education or training. A metadata instance for a learning object describes relevant characteristics of the learning object to which it applies. Such characteristics may be grouped in General, Life Cycle, Meta-metadata, Educational, Technical, Rights, Relation, Annotation and Classification categories.

The conceptual data schema specified permits linguistic diversity of both learning objects and the metadata instances that describe them. It also specifies the data elements composing a metadata instance for a learning object.

The purpose of the LOM schema is to facilitate search, evaluation, acquisition and use of learning objects, for instance by learners or instructors or automated software processes. It also facilitates the sharing and exchange of learning objects among parties.

LOM precisely defines the metadata needed to describe a learning object in order to make easier the search, evaluation, acquisition and use of learning objects.

The data elements that describe a learning object are grouped into categories. Each one of these categories describes a different aspect of the learning object. The categories defined in LOM are the following:

- General: It groups the general information that describes the learning object as a whole.
- Lifecycle: It groups the features related to the history and current state of this learning object and those who have affected this learning object during its evolution.
- Meta-metadata: It groups information about the metadata instance itself (rather than the learning object that the metadata instance describes).
- Technical: It groups the technical requirements and technical characteristics of the learning object.
- Educational: It groups the educational and pedagogic characteristics of the learning object.
- Rights: It groups the intellectual property rights and conditions of use for the learning object.
- Relation: It groups features that define the relationship between the learning object and other related learning objects.
- Annotation: It provides comments on the educational use of the learning object and provides information on when and by whom the comments were created.
- Classification: It describes this learning object in relation to a particular classification system.

These categories group data elements that form a hierarchy, including aggregate data elements and simple data elements. From the two, aggregate and simple data elements, only the simple ones have an individual value. For this reason, aggregate elements do not have a value space or a datatype defined. For both data elements, aggregate and simple, the LOM v1.0 Base Schema defines its name, explanation, size, order and an illustrative example, for better understanding the purpose of the element. For more information on the components of the LOM v1.0 Base Schema, please refer to [LOM02a].

3.1.2 Standard for ISO/IEC 11404 binding for Learning Object Metadata data model

The project scope of this standard is to specify an ISO/IEC 11404:1996 (Language Independent Datatypes) [ISO96a] binding of the IEEE 1484.12.1 [LOM02a] data model. An implementation that conforms to 1484.12.2 must conform to 1484.12.1

The purpose of this standard is to provide precise data model semantics, as permitted by the 11404 notation. The 11404 notation may be useful for bindings to programming languages and other systems.

3.1.3 Standard for extensible markup language binding for Learning Object Metadata data model

This Standard provides an eXtensible Markup Language (XML) binding of the learning object metadata (LOM) data model defined in 1484.12.1–2002 Standard for Learning Object Meta-data. The purpose of this Standard is to allow the creation of LOM XML binding instances in a standard way. This allows for interoperability of LOM XML binding instances between various systems.

It differentiates between two types of XML bindings instances conformance, strictly conforming bindings and conforming bindings.

The strictly conforming binding instances have to accomplish the following conditions:

- shall conform to the constraints expressed in the strict schema profile defined in [LOM03b]
- the root element of the instance shall be lom
- the root element of the instance shall contain an xmlns declaration for the LOM namespace. The LOM namespace is defined to be http://ltsc.ieee.org/xsd/LOMv1p0 [LOM04b]
- shall conform to the LOM data model requirements of IEEE 1484.12.1–2002
- shall conform to the requirements of XML binding definition
- shall not contain any extensions to the LOM data model defined in IEEE 1484.12.1–2002 [LOM02a]

The conforming binding instances have to accomplish the following conditions:

- shall conform to the constraints expressed in the custom schema profile or the loose schema profile [LOM03a]
- the root element of the instance shall be lom
- the root element of the instance shall contain an xmlns declaration for the LOM namespace. The LOM namespace is defined to be http://ltsc.ieee.org/xsd/LOMv1p0 [LOM04b]
- shall conform to the LOM data model requirements of IEEE 1484.12.1–2002
- shall conform to the requirements of XML binding definition

This standard defines three different XML schema bindings, top-level general, top-level custom, top-level loose and top-level strict. Moreover, several supporting schemas for datatypes and elements are also defined.

The top-level general schema describes a schema for validating XML binding instances using a common set of validation assumptions.

The top-level custom schema represents a top-level schema for validating XML binding instances using custom validation for vocabulary values.

The top-level loose schema represents a top-level schema for validating XML binding instances using loose validation for vocabulary values.

The top-level strict schema represents a top-level schema for validating XML binding instances using strict validation for vocabulary values.

The supporting schemas include, data types, element names, element types, vocabulary types, vocabulary values, unique loose, unique strict, extend custom, extend strict, vocabulary custom, vocabulary loose and vocabulary strict. They provide definitions of the schemas for the different metadata elements provided in LOM. The complete description can be found in [LOM03a].

3.2 Dublin Core Metadata Initiative

The Dublin Core Metadata Initiative (DCMI) [DCMI04a] is an organization dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing resources that enable more intelligent information discovery systems.

The mission of DCMI is to make it easier to find resources using the Internet through the following activities:

- Developing metadata standards for discovery across domains
- Defining frameworks for the interoperation of metadata sets
- Facilitating the development of community- or disciplinary-specific metadata sets that are consistent with the previous items in the list

The range of activities of DCMI includes standard development and maintenance by means of international workshops and working group meetings. It also comprises tools, services and infrastructure for supporting the management and maintenance of DCMI metadata in multiple languages. Moreover, they develop and distribute educational and training resources for educational outreach and community liaison, coordinating activities within and between metadata communities.

3.2.1 Dublin Core Metadata element set

The Dublin Core metadata element set is a standard for cross-domain information resource description. An information resource is defined to be anything that has identity. There are no fundamental restrictions to the types of resources to which Dublin Core metadata can be assigned. It is described in the following standards:

- ISO Standard 15836-2003 [ISO03a]
- NISO Standard Z39.85-2001 [NISO01a]
- CEN Workshop Agreement CWA 13874, currently not supported because of the publication of the ISO Standard

In the element descriptions, each element has a descriptive label intended to convey a common semantic understanding of the element, as well as a unique, machine-understandable, single-word name intended to make the syntactic specification of elements simpler for encoding schemes.

Each element is optional and repeatable. Metadata elements may appear in any order. The ordering of multiple occurrences of the same element may have a significance intended by the provider, but ordering is not guaranteed to be preserved in every system.

For each metadata element, some information is given. In particular, for each of the 15 elements conforming Dublin Core it is defined its name, a label, a definition and a comment. The metadata elements are described from Table 3 to Table 17:

Table 3. Dublin Core Title Element

Element Name	Title
Label	Title
Definition	A name given to the resource
Comment	Typically, Title will be a name by which the resource is formally known

Table 4. Dublin Core Creator Element

Element Name	Creator
Label	Creator
Definition	An entity primarily responsible for making the content of the resource
Comment	Examples of Creator include a person, an organization or a service. Typically, the name of a Creator should be used to indicate the entity

Table 5. Dublin Core Subject Element

Element Name	Subject
Label	Subject and Keywords
Definition	A topic of the content of the resource
Comment	Typically, Subject will be expressed as keywords, key phrases or classification codes that describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme

Table 6. Dublin Core Description Element

Element Name	Description
Label	Description
Definition	An account of the content of the resource
Comment	Examples of Description include, but are not limited to, an abstract, table of contents, reference to a graphical representation of content or free-text account of the content

Table 7. Dublin Core Publisher Element

Element Name	Publisher
Label	Publisher
Definition	An entity responsible for making the resource available
Comment	Examples of Publisher include a person, an organisation or a service. Typically, the name of a Publisher should be used to indicate the entity

Table 8. Dublin Core Contributor Element

Element Name	Contributor
Label	Contributor

Definition	An entity responsible for making contributions to the content of the resource
Comment	Examples of Contributor include a person, an organisation or a service. Typically, the name of a Contributor should be used to indicate the entity

Table 9. Dublin Core Date Element

Element Name	Date
Label	Date
Definition	A date of an event in the lifecycle of the resource
Comment	Typically, Date will be associated with the creation or availability of the resource

Table 10. Dublin Core Type Element

Element Name	Type
Label	Resource Type
Definition	The nature or genre of the content of the resource
Comment	Type includes terms describing general categories, functions, genres or aggregation levels for content. To describe the physical or digital manifestation of the resource, use the Format element

Table 11. Dublin Core Format Element

Element Name	Format
Label	Format
Definition	The physical or digital manifestation of the resource
Comment	Typically, Format will include the media-type or dimensions of the resource. Format may be used to identify the software, hardware or other equipment needed to display or operate the resource

Table 12. Dublin Core Identifier Element

Element Name	Identifier
Label	Resource Identifier
Definition	An unambiguous reference to the resource within a given context
Comment	Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system

Table 13. Dublin Core Source Element

Element Name	Source
Label	Source
Definition	A reference to a resource from which the present resource is derived
Comment	The present resource may be derived from the Source resource in whole or in part. Recommended best practice is to identify the referenced resource by means of a string or number conforming to a formal identification system

Table 14. Dublin Core Language Element

Element Name	Language
Label	Language
Definition	A language of the intellectual content of the resource
Comment	Recommended best practice is to use RFC 3066 [RFC01a], which, in conjunction with ISO 639 [ISO02a], defines two-and three-letter primary language tags with optional subtags

Table 15. Dublin Core Relation Element

Element Name	Relation
Label	Relation
Definition	A reference to a related resource
Comment	Recommended best practice is to identify the referenced resource by means of a string or number conforming to a formal identification system

Table 16. Dublin Core Coverage Element

Element Name	Coverage
Label	Coverage
Definition	The extent or scope of the content of the resource
Comment	Typically, Coverage will include spatial location (a place name or geographic coordinates), temporal period (a period label, date or date range) or jurisdiction (such as a named administrative entity)

Table 17. Dublin Core Rights Element

Element Name	Rights
Label	Rights Management
Definition	Information about rights held in and over the resource
Comment	Typically, Rights will contain a rights management statement for the resource or reference a service providing such information. Rights information often encompasses Intellectual Property Rights (IPR), Copyright and various Property Rights. If the Rights element is absent, no assumptions may be made about any rights held in or over the resource.

3.2.2 Dublin Core Qualifiers

Apart from Dublin Core Elements, also Dublin Core Qualifiers have been defined. They are a refinement of the 15 elements, whose need was identified by many organisations wanting to implement the Dublin Core. The DCMI also recognised this need, and mechanisms and procedures have been defined to further qualify the 15 elements.

A first set of qualifiers was published in July 2000, after an agreement on the basic principles for the mechanism of qualification was reached. This first set of qualifiers was not intended to satisfy all applications. In determining the makeup of these qualifiers, preference was given to vocabularies, notations, and terms already maintained by established agencies. A registration process will be used to identify relevant vocabularies, encoding and schemes.

Inevitably, there will be situations where an agent or client will encounter Dublin Core Metadata Element Set (DMES) descriptions that use unfamiliar qualifiers developed by implementors for specialized local or domain-specific needs. The useful interpretation of such a DCMES description will depend on the ability of an application to ignore the unknown qualifiers and fall back on the broader meaning of the element in its unqualified form. The guiding principle for the qualification of Dublin Core elements, colloquially known as the *Dumb-Down Principle*, is that a client should be able to ignore any qualifier and use the information as if it were unqualified. While this may result in some loss of specificity, the remaining element value (without the qualifier) should continue to be generally correct and useful for discovery.

Currently, the DCMI recognizes two broad classes of qualifiers:

- **Element Refinement**: These qualifiers make the meaning of an element narrower or more specific. A refined element shares the meaning of the unqualified element, but with a more restricted scope. A client that does not understand a specific element refinement term should be able to ignore the qualifier and treat the metadata value as if it were an unqualified (broader) element. The definitions of element refinement terms for qualifiers must be publicly available.
- **Encoding Scheme**: These qualifiers identify schemes that aid in the interpretation of an element value. These schemes include controlled vocabularies and formal notations or parsing rules. A value expressed using an encoding scheme will thus be a token selected from a controlled vocabulary or a string formatted in accordance with a formal notation. If a client or agent does not understand an encoding scheme, the value may still be useful to a

human reader. The definitive description of an encoding scheme for qualifiers must be clearly identified and available for public use.

All of the qualifiers listed in Table 18 fall into one of these two categories. There is one more element, Audience, which is at the element level but not one of the original 15 elements, and it should only be used with qualified Dublin Core.

Additional qualifier categories may evolve over time and with implementation experience. The qualifiers listed here do not constitute a closed set, designed to meet all of the descriptive needs of implementors. Rather, they form the foundation for a larger body of qualifiers that will evolve as various communities develop additional qualifiers. The complete description of element refinements can be found in [DCMI03c].

Table 18. Dublin Core Element Refinements and Element Encoding Schemes

Element	Element Refinement(s)	Element Encoding Scheme(s)
Title	Alternative	-
Creator	-	-
Subject	-	LCSH MeSH DDC LCC UDC
Description	Table Of Contents Abstract	-
Publisher	-	-
Contributor	-	-
Date	Created Valid Available Issued Modified Date Copyrighted Date Submitted	DCMI Period W3C-DTF
Type	-	DCMI Type Vocabulary
Format	-	IMT
	Extent	-
	Medium	-
Identifier	-	URI
	Bibliographic Citation	-
Source	-	URI
Language	-	ISO 639-2 RFC 3066

Relation	Is Version Of Has Version Is Replaced By Replaces Is Required By Requires Is Part Of Has Part Is Referenced By References Is Format Of Has Format Conforms To	URI
Coverage	Spatial	DCMI Point ISO 3166 DCMI Box TGN
	Temporal	DCMI Period W3C-DTF
Rights	Access Rights	-
Audience	Mediator Education Level	-

3.2.3 Implementing Dublin Core in XML

To sum up with the explanation about the Dublin Core Metadata Initiative, we describe in this section the way of implementing Dublin Core in XML. A series of guidelines have been defined in order to facilitate this task [DCMI03a]. Both qualified and non-qualified Dublin Core is considered.

Guidelines can be grouped into three groups: General implementation guidelines, Simple Dublin Core implementation guidelines and Qualified Dublin Core implementation guidelines.

The general implementation guidelines mostly refer to the proper use of XML Schemas and namespaces in order to describe Dublin Core information.

The Simple Dublin Core implementation guidelines define an abstract model and several recommendations. The abstract model can be summarised as follows:

- A simple Dublin Core (DC) record is made up of one or more properties and their associated values.
- Each property is an attribute of the resource being described.
- Each property must be one of the 15 Dublin Core Metadata Element Set [DCMI03b] elements.
- Properties may be repeated.

- Each value is a literal string.
- Each literal string value may have an associated language.

The recommendations refer to the proper representation in XML language of the DC elements and their contents. In particular, it is said that property names for the 15 DC should be lower-case or that properties should be encoded as XML elements and the values of the properties should be the content of the element. Someone following these recommendations must also follow general ones.

The Qualified Dublin Core implementation guidelines define an abstract model and several recommendations. The abstract model can be summarised as follows:

- A qualified DC record is made up of one or more properties and their associated values.
- Each property is an attribute of the resource being described.
- Each property must be either:
    - one of the 15 DC elements,
    - one of the other elements recommended by the DCMI (e.g. audience) [DCMI03d],
    - one of the element refinements listed in the DCMI Metadata Terms recommendation [DCMI03d].
- Properties may be repeated.
- Each value is a literal string.
- Each value may have an associated encoding scheme.
- Each encoding scheme has a name.
- Each literal string value may have an associated language.

The recommendations refer to the description of element refinements using XML schemas. In particular, it described that an element refinement should be treated in the same way as other properties. It is also described how names for element refinements and encoding schemes have to be specified. Someone following these recommendations must also follow general ones and Simple Dublin Core ones.

Finally, it is described in [DCMI03a] how DC metadata can be mixed with other metadata schemas. Some examples are given that relate DC metadata with other metadata schemas. The objective of mixing different kinds of metadata schemas is to take profit from the properties already defined in each schema for creating richer metadata applications.

3.3 CEN/ISSS Workshop on Dublin Core Metadata

3.3.1 European Committee for Standardization (CEN)

CEN [CEN04a], the European Committee for Standardization, was founded in 1961 by the national standards bodies in the European Economic Community and European Free Trade Association countries (Iceland, Liechtenstein, Norway and Switzerland).

Now CEN is contributing to the objectives of the European Union and European Economic Area with voluntary technical standards which promote free trade, the safety of workers and consumers, interoperability of networks, environmental protection, exploitation of research and development programmes, and public procurement.

CEN is a system of formal processes to produce standards, shared principally between:

- 28 National Members and the representative expertise they assemble from each country. These members vote for and implement European Standards;
- 8 Associate Members and two Counsellors;
- The CEN Management Centre, Brussels.

It works closely with other relevant standardization committees and organizations.

A CEN Workshop is an open process that aims to bridge the gap between industrial consortia that produce de facto standards with limited participation of interested parties and the formal European standardization process, which produces standards through consensus under the authority of the CEN member bodies.

A CEN Workshop offers a new mechanism and approach to standardization because it gives the opportunity to clients to bring their standardization and specification requirements, providing a solution according to their real needs.

The set up and operation of a Workshop is very simple, giving the opportunity to the members of the Workshop to make all relevant decisions over it. These include all market players (industry, service providers, administrations, users and consumers) and can come from any part of the globe. They are responsible for the funding and direction of the Workshop and for the approval of the deliverables.

The main activity of a CEN Workshop is the development and publication of the CEN Workshop Agreement. CEN Workshop Agreements (CWAs) are consensus-based specifications, drawn up in an open Workshop environment.

Workshops can be introduced anywhere in the CEN environment, but the pressing needs of the Information Society meant that they were introduced first in the information and communication technology (ICT) area under the direction of the CEN Information Society Standardization System (CEN/ISSS).

3.3.2 CEN/ISSS Workshop on Dublin Core Metadata

The MMI-DC Workshop provides an open forum at the European level in which Dublin Core Metadata standards related issues are addressed. At an international level, work resides within the Dublin Core Metadata Initiative [DCMI04a]. The name MMI-DC comes from the fact that this Workshop was the successor of the MMI Workshop (Metadata for Multimedia Information) which was active in the period 1998-1999 and which dealt with metadata more generally.

The Workshop has recently finished a number of CEN Workshop Agreements, which can be downloaded from CEN Web site.

- CWA 14860 - Dublin Core eGovernment Application Profiles [CWA03a], which presents a proposed metadata application profile based on Dublin Core for eGovernment in Europe.
- CWA 14859 - Guidance on the use of metadata in eGovernment [CWA03b], which gives guidance on the application of metadata for describing resources in the domain of eGovernment.
- CWA 14855 - Dublin Core Application Profile guidelines [CWA03c], specifies Dublin Core Application Profile. It is a declaration specifying which metadata terms an organization, information provider, or user community uses in its metadata.
- CWA 14857 - Mapping between Dublin Core and IS 19115, Geographic Information – Metadata [CWA03d], whose main objective is to define a mapping between the Dublin core metadata specifications and DIS 19115 to improve the discovery of geographical information in cross-domain searches.
- CWA 14856 - Guidance material for mapping between Dublin Core and ISO in the Geograpic Information domain [CWA03e], whose main objective is to define a mapping between the Dublin core metadata specifications and DIS 19115 to improve the discovery of geographical information in cross-domain searches.
- CWA 14858- Dublin Core Spatial Application Profile [CWA03f], gives an example of how Dublin Core metadata can be used inside the geographic information context, based on application profiles defined in [CWA03c].

Some more CWAs are going to be done during 2004 inside the Workshop:

- CWA1 - General guidance for the deployment of Dublin Core metadata (to supersede CWA 13988: 2003) [CWA04a]
- CWA2 - EU eGovernment Metadata Framework (see also CWA14859 and 14860) [CWA04a]
- CWA3 - Guidance for the deployment of the EU eGovernment Metadata Framework (see also CWA14859 and 14860) [CWA04a]
- CWA4 - Guidance for the deployment of Dublin Core metadata in Corporate environments [CWA04a]
- CWA5 - Specification of machine-readable representation of Application Profiles [CWA04a]
- CWA6 - Guidance for naming, versioning, evolution and maintenance of element declarations and Application Profiles [CWA04a]

Several activities where abandoned inside this workshop:

- CWA 13988: 2003 containing "Guidance on use of Dublin Core in Europe"
- The metadata observatory that was developed by WS/MMI-DC during 1999-2001 is no longer maintained. Its contents will be integrated into the MetaGuide at the State and University Library Göttingen, Germany.
- CWA 13874: endorsing Dublin Core Metadata Element Set Version 1.1 is now withdrawn now that it has been published as ISO 15836.

Contribution

4 Outline of the contribution

The research done in this thesis work has analysed several aspects of electronic commerce of services and other related concepts whose need has appeared during the development of the research. As a result, we have obtained a series of contributions, further developed in next sections. These contributions can be summarised in the following list:

- Definition of the electronic commerce of services, e-services
    - Description of e-service components
    - Use of workflow for the definition of the structure of e-services
- Identification of the characteristics of e-services
    - Degree of dynamism
    - Degree of knowledge
    - Generic vs. specific e-services
- Description of e-services based on their characteristics
- Classifications of e-services
- Relationship among different e-services classifications
- Description of self-learning mechanism for dynamic e-services
- Methodology for the definition of e-services
    - Description of the different levels of metadata present in a system offering e-services
    - Definition of e-services workflow
        - Use of XML language
        - Use of DAML-S
    - Definition of the control and information flow inside an e-service
    - Functional model
        - Entities
        - Operations
- Validation of the methodology
    - Real system implementation: Legal and administrative services
    - Collaborative editing service
        - Metadata inside documents generated by means of collaborative editing

Each item in the list is described in more detail in the different sections of this contribution part, as explained in next section.

4.1 General overview of the contribution

This section describes the evolution of the research, since its beginning, relating each research field with the appropriate contribution section involved.

After an initial version of a system for the provision of services by electronic means was implemented in the European project TRADE, presented in section 9.1.6.1, we started the study of the general features of electronic commerce of services. This research work was also based on previous work done in our research team [GALLE01a], concerning the categorisation of electronic commerce and the functional model behind it. This study led us to the definition of the electronic commerce of services, in short, e-services, as explained throughout section 5, Electronic commerce and e-services.

The next step taken was the definition of the different components required for services offered by electronic means. As a result of this definition work, the following components, which are described in section 5.2.1, were found:

- Service structure, based on the steps to be followed for its development and the dependencies and relationship among these steps
- Participants of the service, defining their roles
- Information needed for the service, either information interchanged by the participants as information needed for the development of the service

Other aspects of the definition of services were evaluated at that point, especially how a service structure could be created or refined after its definition. As a result of this evaluation, we were able to define different categories for services. These categories are completely described in section 6, Classification of e-services. In section 6 it is also described a self-learning mechanism for facilitating the creation and refinement of a special category of e-services, the dynamic ones.

Then, we started with the definition of each service component, considering what was needed to specify it.

For the service structure component, we adopted a workflow approach, which is put into context in section 7, Workflow in electronic commerce, including process descriptions, as we could relate steps, participants and information. An important decision taken during the definition of this component was the use of XML language to describe the structure of the service. The main reasons for making this decision were, among others, the wide support made by vendors and researchers over XML language, its ease of use and adoption in networked architectures, especially in electronic commerce ones, and the possibility of implementation of other services associated to XML documents, such as advanced searches over document content or creation of new services.

Once the structure was defined, the rest of components had to be described. As steps, participants and information could be seen as data about data, that is, data about the service, but not the service itself, we treated these components as metadata associated to the service. Inside these metadata, we established different levels, based on the description we made of a system for providing electronic commerce of services.

The description of the components of services led us to the description of an initial version of our methodology, explained in detail in section 8, Methodology, where the following parts were defined:

- Metadata and information associated to e-services
- Workflow definition using XML
- Control and information flow
- Functional model

In order to apply the concepts described in the methodology, a new version of the system initially implemented in the European project TRADE was built under the auspices of the Área 2000 project; it was the newTRADE sub-project, presented in section 9.1.6.3.

Our additional requirement of using XML language for the description of the service structure was also followed by several international initiatives like the ones presented in the state of the art. In this way, we selected OWL-S, formerly known as DAML-S, as the one best suited to perform the workflow definition using an XML-derived language. This decision had some effect over our methodology, as the workflow definition and functional model components had to be further refined, as described in section 8.3.2, Workflow definition using DAML-S.

This methodology refinement implied a new validation that was done by defining legal and administrative services and collaborative editing services with the help of it, as explained in section 9, Validation of the model: e-Services implementation, Appendix A, Legal e-service states, and Appendix B, Collaborative editing e-services states.

Finally, the description of collaborative editing service, explained in detail throughout section 9.2 and Appendix B, opened a new research field in the area of metadata description. Although a solution is not provided in this research work, the metadata initiatives described in the state of the art have been studied in order to evaluate their applicability to the documents created using the collaborative editing service, and, by extension, to the definition of services.

5 Electronic Commerce and e-services

Electronic commerce is a research area that is expanding continuously. Nevertheless, we are able to clearly distinguish among several kinds of electronic commerce, each of them with a very different degree of expansion: Electronic commerce of products (that may be electronic or not), electronic commerce of customised products and electronic commerce of services (***e-services***, in short) [LLOR01a].

We have made this distinction based on what is sold: Products (including customised ones) and services. Considering these three classes of electronic commerce, the one that has reached the highest degree of relevance or expansion until the moment is electronic commerce of products. For this reason, we have decided to concentrate this research work especially on one of the other two classes of electronic commerce: electronic commerce of services, e-services.

For the sake of brevity, we will only talk about electronic commerce of services or electronic commerce of products throughout this work. Electronic commerce of customised products is something in-between products and services. It can share properties with both, being different from both at the same time. If there is any characteristic that especially applies to customised products it will be explained separately on the corresponding section.

5.1 e-Services vs. electronic commerce of products

e-Services (electronic commerce of services), like electronic commerce of products, pass through different phases or steps to successfully reach their objective: The sale of a product offered by a seller or the provision of a service offered by service provider. Depending on the e-service offered the service provider could be a professional, a company or a government institution.

Nevertheless, to accomplish their goals, electronic commerce of products and e-services may use different approaches, those completely depending on the specific characteristics of each one.

So, whilst electronic commerce of products basically consists on selecting items we want to buy from a catalogue and pay for them (willing to receive them safe and sound either in an electronic or in a physical way), electronic commerce of services consists on searching for a professional or company that offers a service and, after coming to an agreement, contracting the selected professional or company.

Afterwards, during the course of the e-service, there can be document and information requests, payments at different moments of the development of the e-service or information interchange with another users. When we refer here to another users, we mean users different from the customer contracting the service and the supplier offering it (either the professional or the company).

Figure 24 illustrates electronic commerce of products together with the actors involved in it. If we pay attention on the buyer side, we see that the initiative of the actions comes from this side: the user, acting as a buyer, wants to purchase a product. For this reason, she makes a search of the products to purchase. The seller side must send a result for this search. If the buyer decides that the products found during the search are the ones she wanted, then, she requests them to the seller and pays for them. If everything is all right, the products will be delivered to the buyer. We are not distinguishing here if the delivery is electronic or physical.

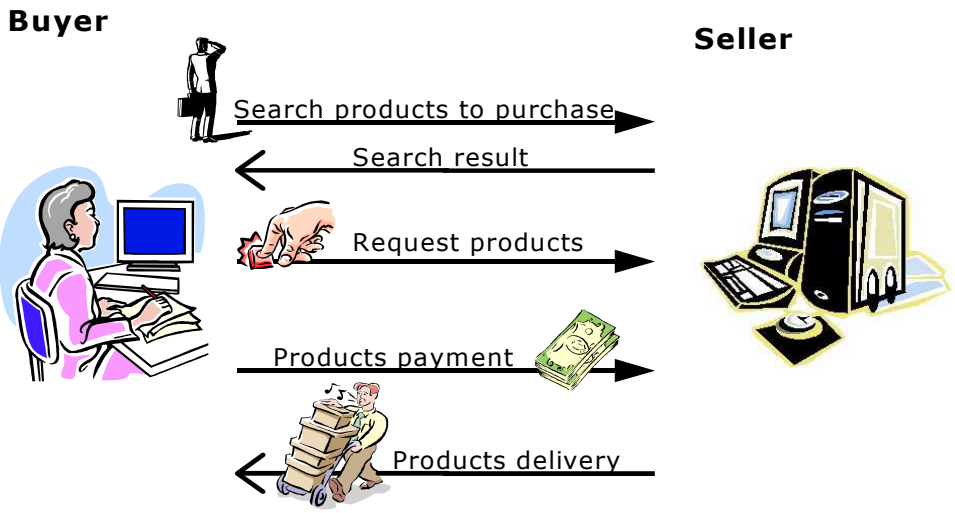


Figure 24. A typical electronic commerce of products scenario

Figure 25 illustrates e-services together with the main actors involved in them. The service provider can be either a professional or a company. Again, the customer side has the initiative of the actions to be taken to contract the e-service. First, she searches for a professional or company offering an e-service with some specific features. As for electronic commerce of products, the customer side receives a result of her search, containing information about who offers the e-service. After that, customer has to agree the contracting conditions of the e-service (for example, how much it will cost or how much time it will last). If customer and e-service provider come to an agreement, then the e-service starts, involving payments, document and information interchanges and also the introduction of new actors during the e-service development.

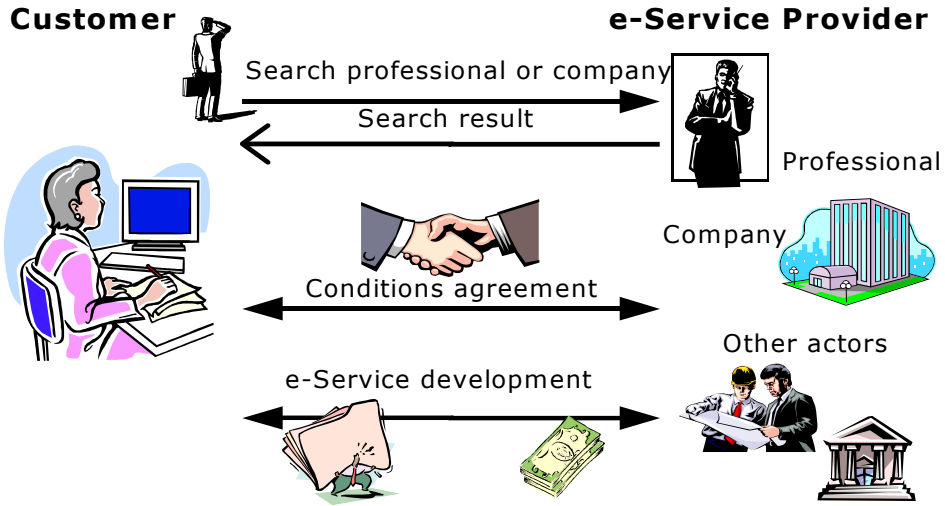


Figure 25. Representation of an e-service scenario

In the previous figures, we have implicitly shown that electronic commerce (e-commerce, in short) of products and e-services are composed by several phases. Our research group has done much work in this area [GALL01], which leaded to the identification of the phases present in electronic commerce of products, customised products and services. For each phase, we have defined its

functionality and, when needed, the sub-phases in which it decomposes. Some phases are common or have a similar functionality in e-commerce of products and e-services, but others have much more functionality and importance in e-services than in e-commerce of products.

Table 19 lists the phases identified together with the operation of each one. These phases are: identification, request, agreement (which contains two sub-phases, payment and delivery) and post-agreement. In Table 19 it is also shown the comparison between the functionality of these phases for the case of e-commerce of products and e-services, highlighting the differences found for each phase.

Table 19. e-Commerce of products and e-services phases' comparison

Phase		e-Commerce of products	e-Services	Differences
Identification		Product identification	Search for the professional or company offering the e-service	This phase has basically the same operation for e-commerce of products and e-services, the identification of the item (product or service) that will be purchased. The way of performing the identification is what can be different
Request		Product request	Ask for service conditions to the professional or company found in the identification phase	As for the identification phase, the operation is very similar for e-commerce of products and e-services, we ask for the item we want to purchase (does not matter if it is a service or a product)
Agreement		Buy the product	Contract the service	Customer and supplier get to an agreement in this phase. In the case of e-service, it does not start in this phase but in the post-agreement one
Sub phases	***Payment***	Pay for the product purchased	Make a prepayment for starting the service (this is optional)	In the case of e-commerce of products, the product will be delivered in this phase, so the payment should be completely done here. On the other hand, for the e-service case, only a prepayment is done. Later, more payments may be needed depending on the e-service.
	Delivery	Receive the product purchased	The e-service cannot be delivered at this moment. See post-agreement	An e-commerce of products transaction finishes here
Post-agreement		Customer support	Service development	In this phase, the main differences appear. e-Commerce of products will only get to this phase if there is a problem in the product purchased while an e-service really ***starts*** at this point.

Figure 26 is an example to illustrate how e-commerce of products goes through the phases listed in Table 19. In Figure 26 it is shown how a CD or DVD can be purchased and the activities that take place on each of the phases defined in Table 19 for the case of e-commerce of products.

First, inside the ***identification phase***, the buyer browses the on-line information of the CD or DVD she wants to purchase. In response to this request, the seller sends information about the CD's or DVD's the buyer has asked for.

In case there is any CD or DVD that she wants to purchase, the buyer inserts it into some kind of shopping cart. These actions may happen several times until the buyer decides that she has identified all the items she wanted to purchase.

Next, we have the ***request phase***, where the buyer indicates that he wants to purchase the items inside the shopping cart to the seller. If everything is correct with the items requested by the buyer, the order is processed into the next phase, ***agreement***. The payment and the delivery of the products purchased are also done in this phase.

Finally, we have ***post-agreement phase***, where the buyer can solve any problem related with the payment or the delivery of the products purchased in the ***agreement phase***.

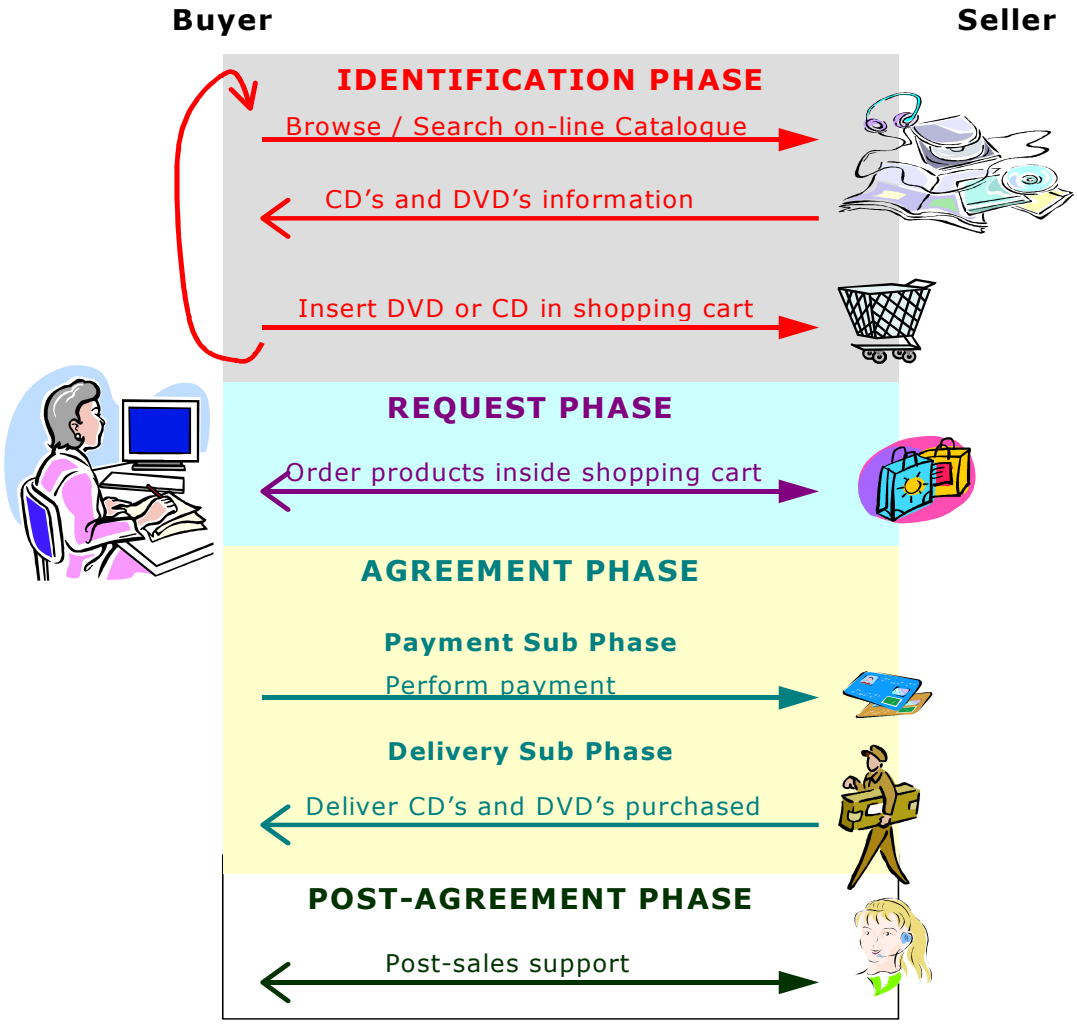


Figure 26. Buying a CD or DVD

Figure 27 shows how an e-service goes through the same phases listed in Table 19 (identification, request, agreement and post-agreement). This will help us in illustrating how these phases can be applied for contracting an e-service offered by an e-service provider.

First of all, the customer wants to hire an e-service. To do so, in the ***identification phase***, she looks for professionals or companies that offer this e-service. The result of this search should be information about the e-service providers, from which the customer has to select one.

Then, the customer asks for e-service conditions to the e-service provider selected in the ***request phase***. Afterwards, the e-service provider will respond with the contracting conditions, or, maybe, he can send a negative response because he is not the best suited to carry on with the e-service requested by customer. In this case, the customer will go back to the ***identification phase***, where she can look for another e-service provider.

For the hiring of the e-service, customer and the e-service provider have to agree in the e-service contracting conditions. At this point, the e-service is contracted being now in the ***agreement phase***. In this phase, the e-service is not delivered as for the case of electronic commerce of products. However, depending on the nature of e-service, it may be needed that the customer performs a pre-payment to the e-service provider in order to start it.

Next, in the ***post-agreement phase***, the e-service will be developed (that is, delivered), performing document interchanges, involving several actors (administrations, other professionals, third parties) and also making some more payments to the e-service provider or to the other actors added during the ***post-agreement phase***.

It is worth noting that post-agreement phase is the more relevant for e-services and it is the one we will focus on this thesis work.

In Figure 24, Figure 25, Figure 26 and Figure 27 we can see that the phases described in Table 19 follow a consecutive order, involving different actions inside each phase. In the same way, e-services post-agreement phase usually involves several dependent actions for accomplishing the final goal of the service, its development.

The transitions among these phases have to be controlled in some way. We propose the definition of a workflow for performing this control.

By using a workflow, we will be able to control the actors involved in each phase, the information and documents they must interchange and the order that phases and information interchanges must follow. Throughout section 7, Workflow inside electronic commerce, we will show how workflow is present in electronic commerce.

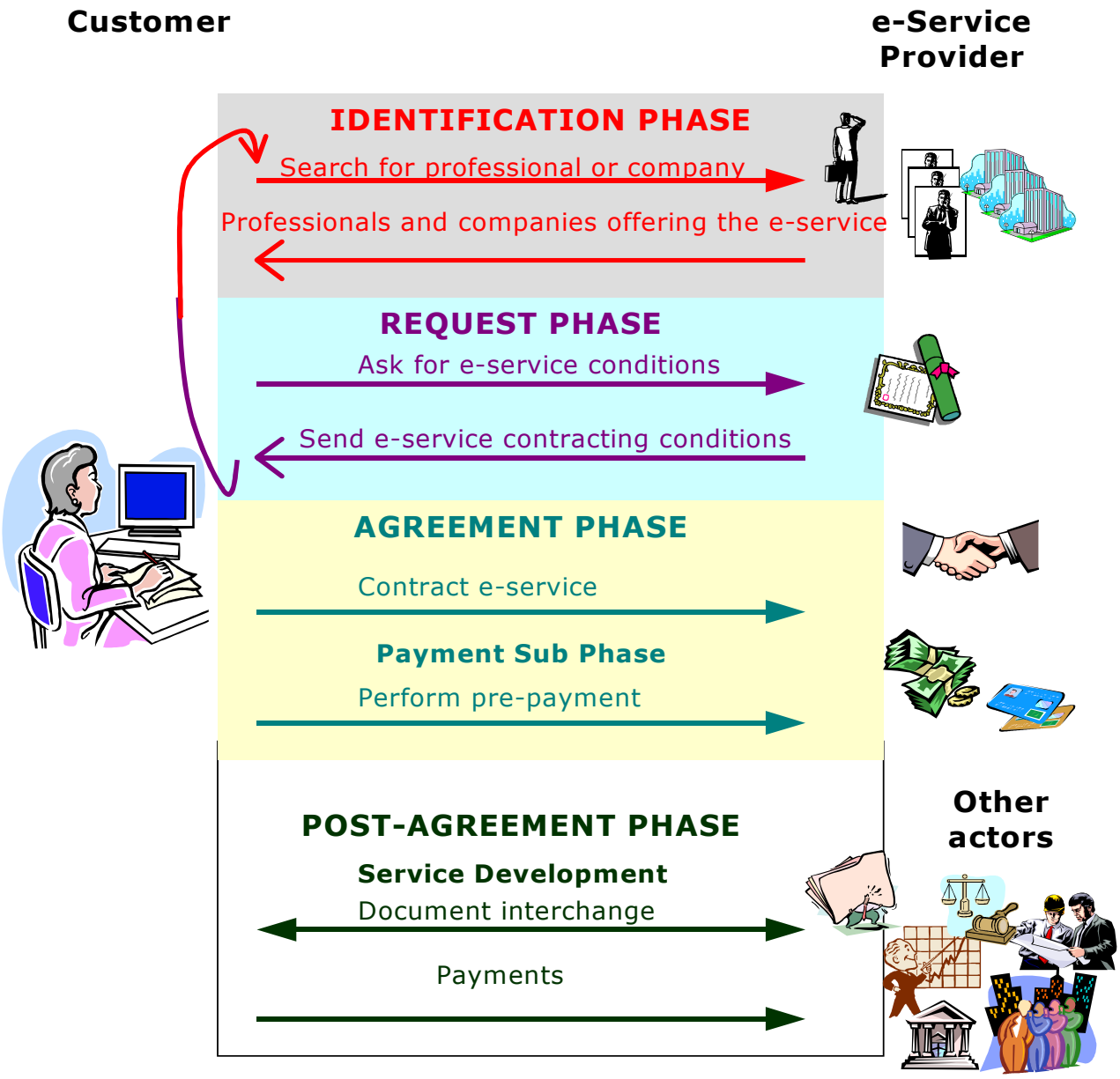


Figure 27. Contracting an e-service

5.2 Definition of e-service

Until now, we have mentioned many times the expression e-service but what do we exactly mean by e-service? Our definition for e-service is: A real-world service carried out by electronic means. With the term real-world we mean traditional.

For instance, a real-world service could be a credit application where a client asks for a specific money amount to a financial institution. The financial institution may determine if the client can afford the credit payment assisted by a computer system. This system, with the aid of a set of financial rules, could decide in most cases if the money can be given to the client or not. However, there are some other cases where the participation of a person is needed for the authorisation of the

credit, because special circumstances arise (special kind of client, more documents needed than the usual ones, etc.), that cannot be evaluated by using automatic mechanisms.

Although it is a very simple example of what an e-service could be, it illustrates the most important components of e-services, which are:

- Participants: Many people may be involved in the development of an e-service. These people can play different roles inside it.
- Steps: Several steps are needed to achieve the final goals of the e-service. These steps have usually some kind of dependency or relationship among them.
- Information interchanges: Much information contained in both documents and forms has to be interchanged among the participants, possibly several times. In some cases, these documents or forms have to be signed by one or more participants of the e-service.

As the expression "e-service" is frequently used nowadays when we refer to any kind of service that is offered in a networked way, we considered important to define exactly what we mean by using this expression in the context of electronic commerce of services. The e-services we mention throughout this research work have the characteristics outlined in the list above: Many users, many steps, much information to interchange into these steps by the users and complex relationship among steps.

The presence of many possibly interconnected steps involving many information interchanges among several users adds a new characteristic not mentioned until now: Some kind of control mechanism is needed for the development of the e-service.

5.2.1 e-Services' components

During the development of this work [LLOR01a, LLOR02a], we have identified e-services' components. They are listed on Table 20, together with their description.

Component	Description
Step	This is the basic component of an e-service. Each of them defines a stage an e-service may have to pass through for its final development.
Steps dependencies	They define the relationship among the steps (which step or steps go before another, if they have a compulsory or optional relationship, etc).
Participant roles	They describe the different user roles that can be involved inside an e-service. A participant role may appear in one or many steps of the e-service.
Information	It defines the different kinds of data that has to be interchanged among the participants of the e-service. This information can be required at different steps inside an e-service.

Table 20. e-Services' components and their description

To completely describe an e-service, the components listed on Table 20 have to be identified. Nevertheless, on next sections we show in more detail how this identification has to be done, as

some components of an e-service may be unknown until the precise moment of the e-service execution. This will lead to different categorisation for e-services, depending on the description of components, among other features. These categories are described in detail in section 6, Classification of e-services.

5.3 e-Services vs. Web Services

Web services have already been described in section 2.2, but we would like to outline some differences among e-Services and Web services in this section.

As we have seen throughout section 5.1, an e-service pass through several phases until it is completely developed. A web service could be one of these phases, or even more. Nevertheless, the e-service must coordinate the web service(s) in order to control the complete e-service phases correct execution. We are not tied at all to web services inside e-services, but we could use them when needed.

6 Classification of e-services

e-Services can be classified from several viewpoints, depending on the different e-service features evaluated [LLOR01a, LLOR02a, LLOR03a]. We have evaluated the following two features for performing a classification of e-services:

- Knowledge
- Degree of dynamism

There is also another classification, but it is not based in an e-service feature. It divides services into generic and specific ones. This distinction is explained in more detail throughout the next sections.

6.1 e-Service classifications

For each classification, we describe the corresponding value of the feature evaluated, the e-service category associated to this value and a brief description of the category. They are developed in the following sections.

6.1.1 Classification depending on the knowledge

The knowledge feature refers to how much we know about an e-service. This understanding about an e-service can be achieved by studying it before it is offered electronically or after it has been developed. Depending on this knowledge, we can classify e-services into the categories shown in Table 21.

Table 21. e-Services classification depending on the knowledge

Knowledge about e-service	e-Service category	Description
Yes	Predefined e-service	e-Services components are described before starting it
No	Non-predefined e-service	The components of an e-service are not described before its execution

In later sections, these categories are explained in more detail. It is also described how we can move from non-predefined category to the predefined one.

6.1.2 Classification depending on the dynamism

The degree of dynamism feature allows us to define another classification for e-services, listed in Table 22. This classification refers to the definition of the e-service, indicating if it can change or not during its execution.

This classification is not excluding with the previous one, since we can find an e-service that can be predefined and dynamic at the same time.

Table 22. e-Services classification depending on their dynamism

Degree of dynamism	e-Service category	Description
High	Dynamic e-service	The structure or components of the e-service change during its execution
Low	Static e-service	The structure of the e-service is maintained during its execution

The differences among static and dynamic e-services and even a mechanism for changing from one category to the other are explained in sections 6.5 to 6.7.

6.1.3 Generic and specific e-services

The last classification, as mentioned in the heading of the section, is not based specifically in a feature. Establishing an analogy with the world of object orientation (O.O.), a generic e-service should be seen as a class while a specific e-service should be seen as an instance of this class, that is, an object. The main difference in this analogy lies on the fact that a specific e-service not always is an instance of a generic e-service, while an object is always an instance of a class. If the later happens, a specific e-service would also be a non-predefined e-service.

It is rather a distinction among e-services, shown in Table 23.

Table 23. e-Services classification depending on their application

e-Service category	Description
Generic e-service	A generic e-service represents a class of services, expressing the common characteristics of all or them.
Specific e-service	A specific e-service represents the instances of running e-services (contracted by someone and being developed). A specific e-service is usually created from a generic e-service description, but this is not compulsory.

In the next sections, we are going to describe in more detail each e-service category previously defined. Nevertheless, an e-service belonging to a category can jump into other category under special circumstances.

6.2 Predefined e-services

An e-service belongs to the predefined category when its structure is defined before it is contracted and started. This represents a big effort, as we have to contemplate every possibility inside the execution of an e-service, before having developed any.

When the e-service is executed, the predefined structure has to be adapted to the specific characteristics of the e-service, that is, the real users participating in the e-service are defined, not the user roles. Moreover, deadlines for information interchange or step execution are defined if

needed. This adaptation could also involve the definition of new steps being part of the e-service, the removal of steps, the addition or deletion of participants or the description of new information interchanges among existing or new participants.

6.3 Non-predefined e-services

Non-predefined e-services are those whose structure is not defined before the e-service is contracted and started. That means that it has to be defined as the user or users in charge of the e-service are developing it.

This definition may consist on the creation of the steps of the e-service, together with the relationships among them and their characteristics, for example, the name or the deadline of the step.

Apart from steps, also participants and the role of these participants have to be defined and the information interchanges among the participants in the different steps of the e-service.

6.4 Issues in the construction of predefined e-services

We have explained the difference among predefined and non-predefined e-services. We have also mentioned the possibility of converting non-predefined e-services into predefined ones. The other way around is not possible, as we cannot convert a predefined e-service into a non-predefined one, as it will be a new completely kind of e-service.

The way of doing this conversion could start from the analysis of similar non-predefined e-services, that is, referring to the same real world service, and extract the common points among them.

The analysis of the non-predefined e-services should be centred in the following aspects:

- Steps composing the e-service and the relationship among them. It is very important to extract the structure of the steps of each different e-service independently of the name of the steps, as the users that have created them could have used different names referring to the same step. Later, a user in charge of creating the new predefined e-service could name the steps accordingly.
- User roles participating on each step of the e-service. This point is not so critical, as the user roles inside any e-service could be predefined by the system offering them. Moreover, it is worth noting, that each user participating inside a non-predefined e-service must belong to a user role. In the worst case, also user roles must be added to the system offering the e-service.
- Information interchanges among user roles inside each step. Again, the information interchanges could be predefined in the system offering e-services. These interchanges of information could be as much documents as forms. Again, if the information interchange did not exist in the system, it should be added.

Having these aspects in mind, the complete structure of the e-service could be extracted from one or more non-predefined e-services. The more e-services to analyse, the more accurate the description of the predefined e-service will be. Nevertheless, we should not forget that this new predefined e-service would have to be adapted to specific e-service characteristics for its later execution.

6.5 Dynamic e-service

Dynamic e-services are those whose structure may vary. This variation may be caused by several reasons:

- The e-service is dynamic by nature, that is, the structure cannot be defined completely, independently of how many times this e-service has been executed or contracted.
- The e-service is dynamic because it is the first time it is being done and we have not been able to define its structure yet. This does not mean that this service is inherently dynamic but circumstantially dynamic, as we could later define its structure.

The mechanism for converting a dynamic e-service into a static one is rather similar to the way of converting a non-predefined e-service into a predefined one.

6.6 Static e-service

We call static to the e-services that do not change form execution to execution. Their structure, together with their participants, information and user roles are known before the e-service is executed, that is, it is predefined and it is not modified during executions. The adaptations permitted are those that assign users to user roles inside specific e-services.

6.7 Dynamic vs. static e-service

An initially dynamic e-service can be converted into a static one if, for example, the participants of a e-service define its structure as it is being developed and, after the e-service is finished, this structure is extracted to conform a new generic e-service to be used by next specific e-services based on it.

On the other hand, an e-service that was thought to be static becomes dynamic because some part of its structure has to be modified for the correct development of one specific e-service based on it. Later, this dynamic e-service can become static again, if the modifications added can be applied to the generic e-service structure, evolving the definition of the e-services as they are executed and refined by the participants.

6.8 Generic e-services

This category directly derives from the fact that we can have a general description of an e-service. This description or definition is what we use for permitting the hiring of the e-services corresponding to the next category, the specific or contracted e-services.

Making an analogy with object orientation concepts, generic e-services could be seen as classes while specific e-services could be seen as objects. For this reason, specific e-services could also be called e-services instances.

6.9 Specific e-services

It has been already mentioned that specific e-services correspond to the e-services being executed by their participants. They describe the execution of one specific instance of a service, determining the current state of the e-service in terms of steps being executed, users participating in the e-service or information interchanges.

6.10 Combination of classifications

To summarise this section, we are going to describe the relationship present between the two categories related to the knowledge we have about an e-service and its degree of dynamism. In Table 24 we show which of these subcategories can be related, as not all combinations are possible. These combinations apply to generic e-services. In the case of specific e-services derived from generic e-services, the first ones initially belong to the same category as the second ones.

	Static	**Dynamic**
Predefined	Yes	Yes
Non-predefined	Not possible	Yes

Table 24. e-Services classification

The following list shows the combinations permitted, together with a brief explanation about each one:

- Predefined static e-service: This combination corresponds to e-services whose structure is defined before it can be started. This structure does not change during e-service execution.
- Predefined dynamic e-service: This combination corresponds to e-services that, although its structure is defined before starting them, we still may have changes on them.
- Non-predefined dynamic e-service: This combination corresponds to e-services whose structure is unknown before they are contracted. For this reason, it structure is defined dynamically, during e-service execution.

6.11 e-Services self-learning

We have seen throughout the previous sections that the categories for classifying e-services are not fixed. In particular, we have tried to explain that the best option is to have predefined static e-services. However, this is not always possible. For this reason, we propose a self-learning mechanism for converting any e-service into as far as possible predefined static e-service.

In the next section, this mechanism is described in detail.

6.11.1 Dynamic e-services self-learning

When the structure of a generic e-service, either dynamic or static converted into dynamic [LLOR03a], is modified during the execution of a specific e-service, we could use these modifications to improve the generic e-service itself. To do so, some sort of self-learning mechanism has to be used.

The operation of this self-learning mechanism must be as follows: First, it should extract the new information from the specific instance of the service by comparing its structure with the generic one, highlighting the differences. Then, a user in charge of the administration of the system offering the service could decide if the newly found information has to be added or not to the generic structure of the service to be used by future instances.

The self-learning mechanism may be refined as more instances of services are developed, to make it as automatic as possible.

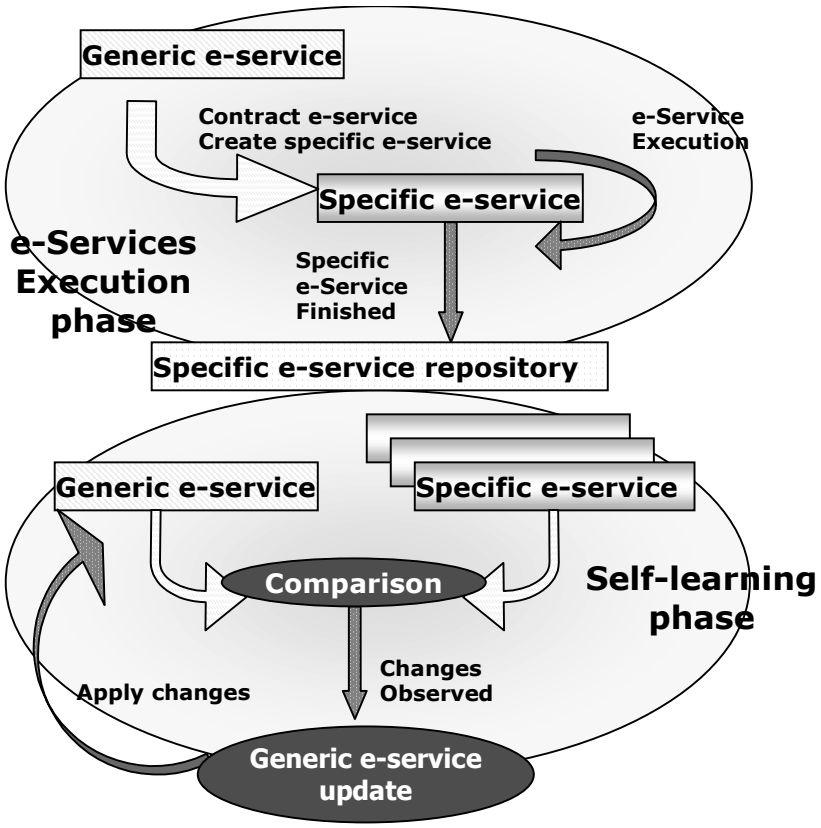


Figure 28. Self-learning mechanism for existing e-services

Figure 28 shows how the self-learning mechanism works. First part of the process corresponds to the e-service development. In this phase, the generic e-service is used by a specific e-service contracted by a user. Then, the e-service is developed and, when the specific e-service has finished, it is stored in the e-service repository. Later on, we can enter on the self-learning phase. In this phase, the specific e-service or e-services are compared to the generic e-service. If there are differences among them, they will be highlighted. An administrative user must decide if they are inserted into the generic e-service.

We could also create new generic e-services by using this mechanism. In this case, the way of working is slightly different. When we create a specific e-service from scratch, the steps conforming it, the users participating in each step or the interchanges of information occurred among these participants are defined as the e-service is developed. Then, when the e-service is finished, with the help of our self-learning mechanism, the generic structure (steps, user roles [not users], information interchanges) is extracted. Again, a user in charge of the administration of the system decides which part of the extracted structure must be in the generic e-service and, finally, the new generic e-service structure is created and it could later used by new specific e-service instances.

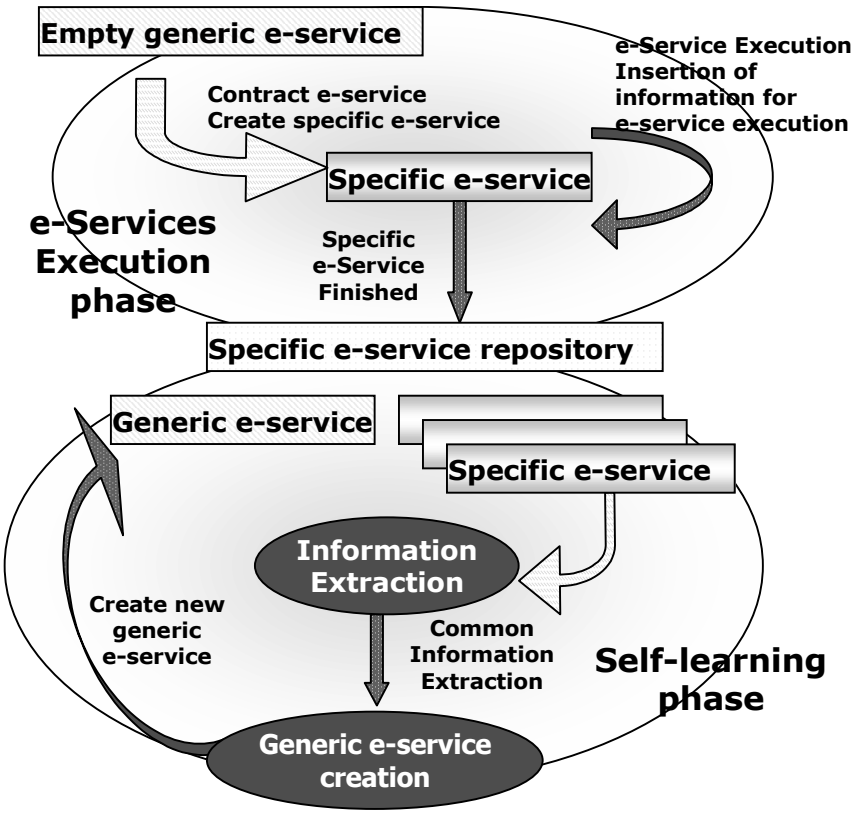


Figure 29. Self-learning mechanism for newly created services

Figure 29 shows the self-learning mechanism for the service instances created from scratch, without having a service template. The service instance is defined at the same time it is being developed. Then, when the service has finished, the common information from the service is extracted (user roles, steps composing the service, relationship among the steps, etc.). Again, an administrative user can decide which information must form part of the service template for this service, and the new service template is created.

7 Workflow inside electronic commerce

7.1 What is workflow?

According to Workflow Management Coalition (*WfMC*) [WFMC04a], workflow can be defined as:

The automation of a business process, in whole or part, during which documents, information or asks are passed from one participant to another for action, according to a set of procedural rules.

In the next section we explain the relationship among this definition of workflow and electronic commerce.

7.2 Use of workflow in electronic commerce

An essential concept in the provision of services and to a lesser degree in the selling of products through electronic commerce is workflow. Although it is not usually considered in this context, workflow is nearly always present in electronic commerce, as much in e-commerce of products as in e-services.

Returning to the definition of workflow, electronic commerce can be generically described as a process where some documents, information or asks are passed from one participant to another for action, according to a set of procedural rules. The final result of this process is typically the purchase of a product or products. Nevertheless, to get to this final result, some series of steps have to be accomplished and they define the flow or work (workflow) in e-commerce.

It is worth noting that we do not restrict here the process as being a business process. If it accomplishes some conditions, anything can be considered as a process that can be controlled with the help of a workflow. We summarise the conditions for a process to be able to be controlled with the help of a workflow as follows:

- The process can be described as a series of steps. We do not limit the relationship among steps. The usual relationship between steps is that one can start when the previous finishes, that is, they are consecutive, but, in some situations, we can find steps that occur in parallel. Moreover, we do not impose any restriction in the initial number of steps, either, as new steps could appear during e-services execution.
- Several user roles participate inside the e-service, interchanging information among them. There are no restrictions on information interchanges, except that they can occur only among user roles present inside a step.

By means of workflow, we could control the general phases of e-commerce described in section 5, Electronic Commerce and e-services, identification, request, agreement and post-agreement. A first approach in the definition of this workflow could be the decomposition of these phases into the corresponding sub-phases. Then, the description of the interchanges of information between users involved in these phases is needed. For the case of e-commerce of products this should not be a difficult task, as not many users are usually involved, but it is not the same for e-services, as we explain in next section.

7.3 Workflow inside e-services

Inside e-services, we still have the same phases as in e-commerce, but there is a difference in the number of users involved in the e-service and also in the phase where the e-service is developed [LLOR02a].

In Electronic commerce and e-services section, section 5, we explained that e-commerce of products usually finishes in the agreement phase, where payment and delivery of product or products purchased is done. We have also explained that for the case of e-services, the *delivery* of the e-service is not done in this phase, but in the post-agreement one. The first three phases, identification, request and agreement, have their own workflow definition for e-services, as they are usually independent of the kind of service being contracted. However, in the post-agreement phase is where workflow plays the most important role, as it can help us in the definition and control of the e-services. The main task for workflow in e-services post-agreement phase is to lead the development of the e-service to its final delivery.

8 Methodology for the definition of e-services

8.1 Motivation

The description of this methodology was motivated by the fact that we made an implementation of a distributed application for providing legal and administrative services in an electronic way. We found that some of the concepts appearing during the implementation of this application could be applicable to any kind of service accomplishing some conditions or characteristics. In particular, if we want to describe a service with this methodology, it should have some of the following characteristics: Several participants, much information associated to the service, many information interchanges among the different participants, several steps in which the service can be decomposed, etc.

For this reason, we identified the general concepts present in electronic commerce of services and started detailing each of them. Afterwards, for a specific kind of services (legal, edition, etc), the concepts have to be further refined.

In the following sections, we describe in detail how to specify an e-service. First of all, we show the metadata needed by a system, which wants to implement e-services. Then, we explain how XML is used for the definition of e-services workflow and the data needed by this workflow. Afterwards, several options in data storage and control organization are presented. Finally, the entities present in e-services, together with some of their operations are shown.

8.2 Metadata and information associated to e-services

Inside a system offering e-services, we need to know several metadata in order to correctly describe and finally implement this system [LLOR01a]. A possibility for doing this is to determine the metadata at the different levels of such a system. We have identified three different levels for grouping metadata inside a system offering e-services: one level referring to the complete system, one level referring to generic e-services and, yet one more level, referring to specific e-services. Each level contains different metadata attaining only to this level, but in some cases, with a direct relationship with the rest of levels.

Some of the metadata we can find on each of these three levels is the following:

- Metadata referring to the complete system: e-Service contractual conditions, users offering the e-service, user contact information, and e-service guidelines.
- Metadata that define the skeleton of a generic e-service: Users involved (roles), associated documentation, e-service steps to accomplish with the objectives.
- Metadata for a specific e-service: Users appearing in the e-service, status of the associated documentation, steps taken in the e-service and its state (in progress, finished, etc).

Apart from metadata, there could be documents associated to the e-services, either in their active phase (documents needed for the development of the e-service that depend on its nature) or in their initial phase (for example a contractual document agreed between parties).

8.2.1 Metadata associated to e-service workflow

The basic information associated to an e-service workflow is:

- Data about e-service: Name, identifier, start date, end date, status.

- Steps that compose the e-service: Name, identifier, start date, end date, reference to the previous step, reference to the next step and status of the step.
- Users involved in the e-service: When they appear, their task, role played inside e-service.
- Documents and information that appears in the e-service: Name, identifier, step which it belongs to, the operations to be applied by the corresponding users (send, request, sign, review, cipher, etc.), status, deadline of the operations to be applied over documentation, date of application of associated operation.

8.2.2 Summary of metadata associated to e-services

Table 25 shows the metadata that can be found inside e-services.

Table 25. Metadata inside e-services

Metadata type	Description	Example
Complete system	Defines the general features needed inside a system offering e-services.	Contractual conditions, e-service guidelines.
Generic e-service	Defines information inside a generic e-service.	User roles, documentation, steps.
Specific e-service	Defines information inside a specific e-service.	Users participating, status of documentation, status of steps.
e-Service workflow	Defines information needed for the workflow of the e-service.	Name of associated e-service, steps of the e-service, user roles inside e-service.

8.3 Definition of e-service workflow

In this section it is described in detail how the definition of an e-service workflow can be done, as we can use different mechanisms to do this. Two of them are described, one using XML [XML04a] and other using DAML-S [DAML04a, DAML03b].

8.3.1 Workflow definition using XML

For the formalisation of workflow definition, we use the XML language [LLOR01a]. We need several files to define completely the system:

- DTD file for the validation of the workflow files.
- Empty e-service XML file for the creation of new e-services from scratch. It will only contain the definition of the skeleton of an e-service with empty values.
- XML files for the definition of the workflow of generic e-services. Each of them will contain the skeleton of a generic e-service, filled in with the information needed for its development.
- Specific e-service XML files.

In Figure 30 it is graphically shown the relationship among generic and specific e-services. Generic e-services are templates or basis for specific e-services. Moreover, here also appears the idea of empty e-service, being also used as a basis for either generic or specific e-services.

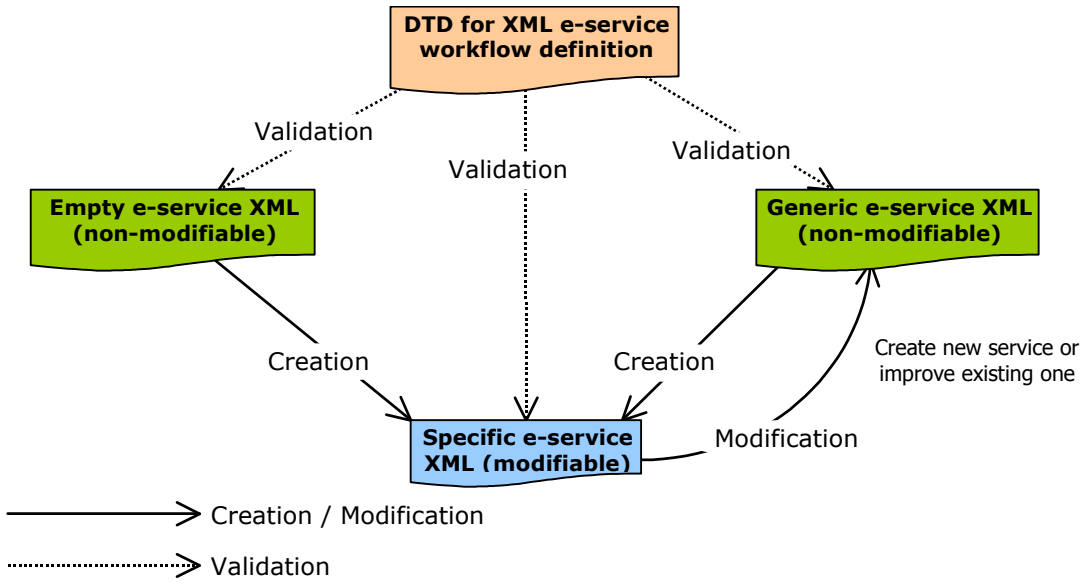


Figure 30. Workflow definition using XML

The workflow definition DTD file checks if the XML files of our system are valid and well formed. Well-formed XML documents only require that all elements in the document be syntactically correct. Valid XML documents, apart from syntactical correctness, must also include a DTD (or a reference to an existing DTD). All elements in the XML document should be defined in the DTD.

When a new instance of an e-service starts, the working e-service XML files are created from empty e-service files or specific e-service files, depending on if the e-service previously existed or not. These files contain the e-service's workflow structure.

When the development of the e-service finishes, the resulting XML file will allow us to create a new specific e-service XML file (which could be used for new working e-services of this type) or modify an existing one with the changes introduced in the working e-service XML file.

In Figure 31 we can find an example of description of an e-service structure using this approach.

```
<service id="Divorce">
	<action>
		<id>Document Acquisition</id>
		<state>started</state>
		<initial/>
	</action>
	<action>
		<id>Serve Proceedings</id>
		<state>started</state>
	</action>
	<action>
		<id>Wait for Respondent Summons</id>
		<state>started</state>
	</action>
	<action>
		<id>Wait for Answer</id>
		<state>started</state>
	</action>
	<action>
		<id>Public Prosecutor Hearing</id>
	</action>
	<action>
		<id>Discovery and Examination of Evidence</id>
	</action>
	<action>
		<id>Trial</id>
	</action>
	<action>
		<id>Judgment</id>
	</action>
	<action>
		<id>Enforcement of Judgment</id>
		<final/>
	</action>
	<action>
		<id>Appeal</id>
		<final/>
	</action>
	<dependency>
		<ini_action>Document Acquisition</ini_action>
		<next_actions>
			<id>Serve Proceedings</id>
		</next_actions>
	</dependency>
	<dependency>
		<ini_action>Serve Proceedings</ini_action>
		<next_actions>
			<id>Wait for Respondent Summons</id>
		</next_actions>
	</dependency>
	<dependency>
		<ini_action>Wait for Respondent Summons</ini_action>
		<next_actions>
			<id>Wait for Answer</id>
		</next_actions>
	</dependency>
	<dependency>
		<ini_action>Wait for Answer</ini_action>
		<next_actions>
			<id>Public Prosecutor Hearing</id>
			<id>Discovery and Examination of Evidence</id>
		</next_actions>
	</dependency>
	<dependency>
		<ini_action>Public Prosecutor Hearing</ini_action>
```

```
            <next_actions>
                  <id>Discovery and Examination of Evidence</id>
            </next_actions>
      </dependency>
      <dependency>
            <ini_action>Discovery and Examination of Evidence</ini_action>
            <next_actions>
                  <id>Trial</id>
            </next_actions>
      </dependency>
      <dependency>
            <ini_action>Trial</ini_action>
            <next_actions>
                  <id>Judgment</id>
            </next_actions>
      </dependency>
      <dependency>
            <ini_action>Judgment</ini_action>
            <next_actions>
                  <id>Enforcement of Judgment</id>
                  <id>Appeal</id>
            </next_actions>
      </dependency>
</service>
```

Figure 31. Example of workflow defined using XML

8.3.2 Workflow definition using DAML-S

8.3.2.1 DAML and OWL

DAML (DARPA Agent Mark-up Language) [DAML04a] is an initiative for the creation of ontologies for any domain, which permits the description of sophisticated class definitions. It extends RDF (Resource Description Framework), a resource description language, which allows the description of basic structures such as classes and properties. DAML-S (S stands for services) [DAML03b] is an ontology created for process description on top of DAML+OIL. [DAML01a]

The Web Ontology Language OWL [OWL04a] is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is developed as a vocabulary extension of RDF (the Resource Description Framework) [RDF04a] and is derived from the DAML+OIL Web Ontology Language. It is planned to base next versions of DAML-S on OWL.

8.3.2.2 Workflow definition

The decision of formalising workflow definition using XML language, as explained in previous sections, was made because it wide support and use by major software vendors and companies. Nevertheless, after making our own workflow definition using XML, we studied other initiatives in workflow and process definition based on XML. The most interesting for our purposes was DAML-S.

In DAML-S, services are modelled as processes. That is, several processes that we must identify and define compose each e-service. For each of these processes, we have to define its inputs, outputs, preconditions and effects (IOPE's). The following list explains each of these concepts in detail.

- Input: Represents the information that is required for the execution of the process. A process can have any number of them.
- Output: Represents the information that the process may provide after its execution. A process can have any number of them. They can have associated conditions.
- Precondition: Represents the conditions that have to be accomplished for the process to be executed. A process can have any number of them.
- Effect: Represents the effects of the process execution. They can have associated conditions.

The e-services conformed by processes can be serialised as XML documents, using DAML-S serialisation. In particular, these descriptions can describe e-service families as well as specific instances of an e-service.

Figure 32 shows a sample process named *Document Interchange*, which is part of an e-service. This process represents an interchange of information between two users. This interchange implies some effect; for instance, the document is accepted and the e-service can continue, moving to next step, or the document is not accepted, and the *Document Interchange* process has to be repeated.

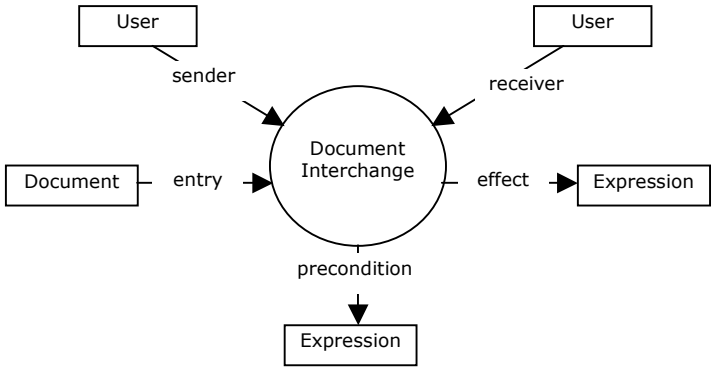


Figure 32. Process description using DAML-S

The process described in Figure 32 can be serialised as an XML file using the DAML-S syntax. This serialisation is shown in Figure 33.

```
<daml:Class rdf:ID="DocumentInterchange">
      <rdfs:subClassOf rdf:resource="&process;#SimpleProcess"/>
      <rdfs:subClassOf>
            <daml:Restriction daml:minCardinality="1">
                  <daml:onProperty rdf:resource="#entry"/>
            </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <daml:Restriction daml:minCardinality="1">
                  <daml:onProperty rdf:resource="#sender"/>
            </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <daml:Restriction daml:minCardinality="1">
                  <daml:onProperty rdf:resource="#receiver"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<rdf:Property rdf:ID="entry">
      <rdfs:subPropertyOf rdf:resource="&process;#input"/>
      <rdfs:domain rdf:resource="#DocumentInterchange"/>
      <rdfs:range rdf:resource="#Document"/>
</rdf:Property>

<rdf:Property rdf:ID="receiver">
      <rdfs:subPropertyOf rdf:resource="&process;#participant"/>
      <rdfs:domain rdf:resource="#DocumentInterchange"/>
      <rdfs:range rdf:resource="#User"/>
</rdf:Property>

<rdf:Property rdf:ID="sender">
      <rdfs:subPropertyOf rdf:resource="&process;#participant"/>
      <rdfs:domain rdf:resource="#DocumentInterchange"/>
      <rdfs:range rdf:resource="#User"/>
</rdf:Property>

<rdf:Property rdf:ID="precondition">
      <rdfs:subPropertyOf rdf:resource="&process;#precondition"/>
      <rdfs:domain rdf:resource="#DocumentInterchange"/>
      <rdfs:range rdf:resource="#Expression"/>
</rdf:Property>

<rdf:Property rdf:ID="effect">
      <rdfs:subPropertyOf rdf:resource="&process;#effect"/>
```

```
      <rdfs:domain rdf:resource="#DocumentInterchange"/>
      <rdfs:range rdf:resource="#Expression"/>
</rdf:Property>
```

Figure 33. DAML-S representation of the process description

8.4 Control and information flow

The definition of control and information flow is very important when designing a system that is going to provide e-services. In order to do this definition, we can take two completely unrelated viewpoints.

The first one concerns to the storage of information in the system. In this sense, we can clearly distinguish between two kinds of information, Documentation information (DOCi) and workflow control information (WFCi) [LLOR01a, LLOR02a].

The second viewpoint refers to workflow control mechanism and shows how it is generated. The subject of this point of view completely depends on the kind of e-service. We describe in detail these two points of view throughout this section.

From the first point of view, storage of DOCi and WFCi, we have the following possibilities:

- DOCi storage
    - Centralised: The DOCi is always stored in the same location.
    - Distributed information storage: The DOCi moves from user to user.
- WFCi storage
    - Centralised: Only one user involved in the e-service has the WFCi and informs the rest that they have to perform some action.
    - Distributed control: The WFCi moves from user to user, when they have to perform some action.

Both can be combined to provide the system we need: Centralised control and DOCi storage, distributed control and DOCi storage, centralised control and distributed DOCi storage and distributed control and centralised DOCi storage.

Figure 34 shows the centralised DOCi storage, where several users access to a centralised location in order to access to documents associated to the e-service.

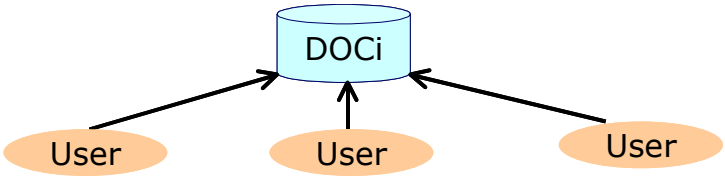


Figure 34. Centralised DOCi storage

Figure 35 shows the distributed DOCi storage, where several users access to different locations in order to access to documents associated to the e-service.

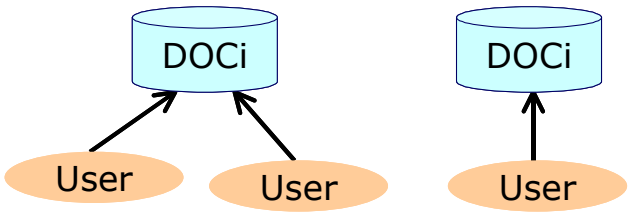


Figure 35. Distributed DOCi storage

The information of the action/s to be done is included in the WFCi.

In case of distributed control, the user who has the control calls the operation/s needed to perform the action. If the control is centralised, the user controlling the workflow advises to the corresponding user of the action that he must perform.

Figure 36 shows the centralised WFCi control. In this case, one user is in charge of controlling the e-service development, the rest of users follow his directions.

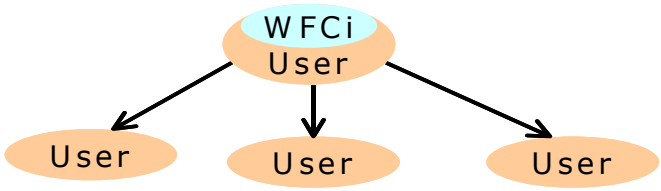


Figure 36. Centralised WFCi control

Figure 37 shows the distributed WFCi control. This control option implies that WFCi travels from user to user during the e-service development, as the user having the WFCi is who controls the development of the e-service.

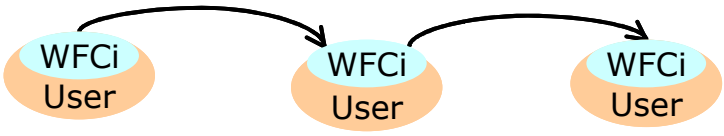


Figure 37. Distributed WFCi control

From the second point of view, the generation of the workflow control mechanism, we can have the following possibilities:

- Predefined workflow: It corresponds to the predefined static e-service presented in section 6.10. If we have a predefined workflow, then the workflow is completely defined before starting the e-service and users must follow it.
- Generated on the fly: It corresponds to the non-predefined e-service. The workflow of the e-service is unknown and participating users indicate what has to be done next for the correct development of the e-service.
- Mix of the two possibilities: It corresponds to the predefined dynamic e-service. In this case we have some knowledge about the e-service. We work with a predefined workflow that can be extended/modified to be adapted to each particular working e-service.

8.5 Functional model

The functional model describes the entities that form part of our system and the operations they provide to other entities [LLOR01a, LLOR02a]. We are going to show in more detail these entities, the operations that they offer to other entities and their relationship.

Nevertheless, during this research work we have refined the functional model. For this reason, we present a preliminary version, which describes entities and their associated operations, together with a possible architecture. We also present the refined final version of functional model.

8.5.1 Entities

We have identified five different classes of entities needed in an e-service [LLOR01a]. They are:

- User entities: The entities that directly dialogue with final users or represent final users (human or machines).
- Knowledge entities: The entities that have the intelligence or knowledge inside the distributed system.
- Notification entities: Entities that offer an interface for the sending of notification among the rest of entities (normally user entities and knowledge entities). They usually have a store-and-forward behaviour.
- Document entities: The entities that are in charge of the management of the documents needed by the different entities of the workflow system.
- Metadata entities: The entities that deal with the system metadata.

Figure 38 shows the relationship among the different entities. The dotted arrows represent the communication between peer entities, but no direct operations invocation is achieved. That is, the user entities do not communicate directly but through the knowledge entity. The solid arrows represent operation invocation and indicate which entities communicate through operations. The direction of the arrows goes from operation caller to operation provider. The results of the operations go on the opposite direction.

This is only a possible functional model. We can easily think about some different models: several notification entities per user entity, several document and metadata entities, several knowledge entities in one system, document and metadata entities accessible directly from user entities, a combination of user entity and knowledge entity in only one entity, etc. We can adapt the functional model to the needs of our system.

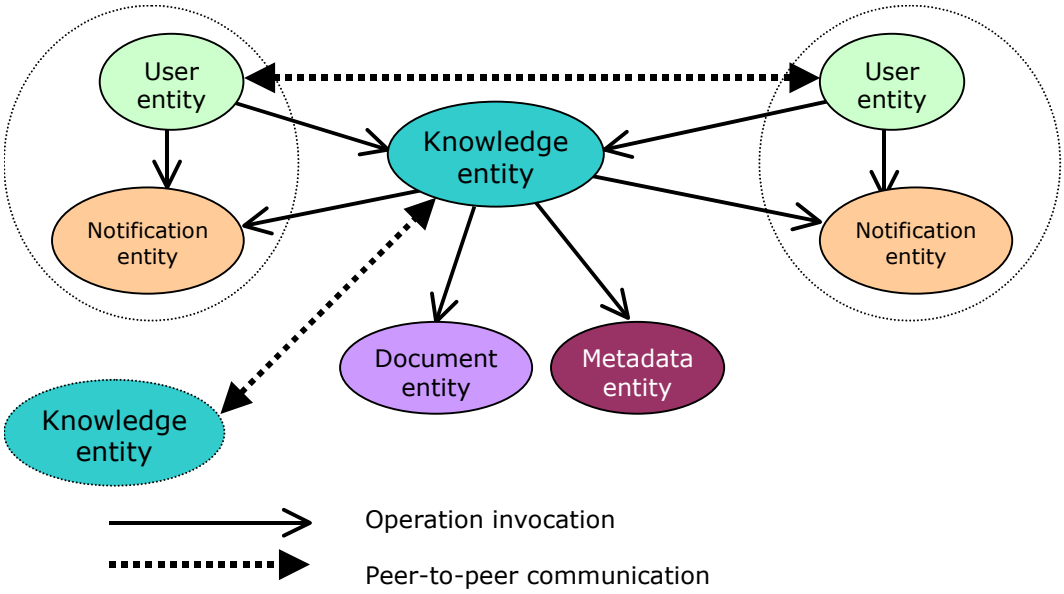


Figure 38. Functional model

The knowledge entities that appear in Figure 38 belong to different systems. It could be interesting to communicate them in order to interchange user profile information, e-service workflow control files and other information related to the system operation. For example, if we have two systems installed in collaborating companies, which have the same system installed but they do not share information, we can interchange the information of the new e-services created by one of them. This scenario can be extended at will (as many companies as you want).

8.5.2 Operations

Each kind of entity described in the functional model offers several operations that other entities can invoke [LLOR01a].

We have identified a restricted set of operations for each class of entity (those on Table 26). These operations are grouped by the different kinds of information they apply to.

As an example, we can see in Table 26 that the operations offered by the knowledge entity over documents are: Create, Read, Write, Search, List, Delete, Sign, Cipher and Review.

Table 26. Entities operations

Entity	Operation / Information	Create	Modify	Read	Write	Search	List	Delete	Sign	Cipher	Review
User	e-Service	✔	✔	✔	✔	✔	✔	✔			
	Document			✔	✔	✔	✔	✔	✔	✔	✔
Knowledge	Metadata	✔	✔	✔	✔	✔	✔	✔			
	Document	✔		✔	✔	✔	✔	✔	✔	✔	✔
Notification				✔	✔		✔	✔			
Document		✔		✔	✔	✔	✔	✔			
Metadata		✔	✔	✔	✔	✔	✔	✔			

8.5.3 System architecture

In this section, we show a possible architecture for the implementation of the functional model previously presented [LLOR01a].

The architecture organises in *layers* the entities involved and Figure 39 shows the protocols that could be used for the communication between these *layers*.

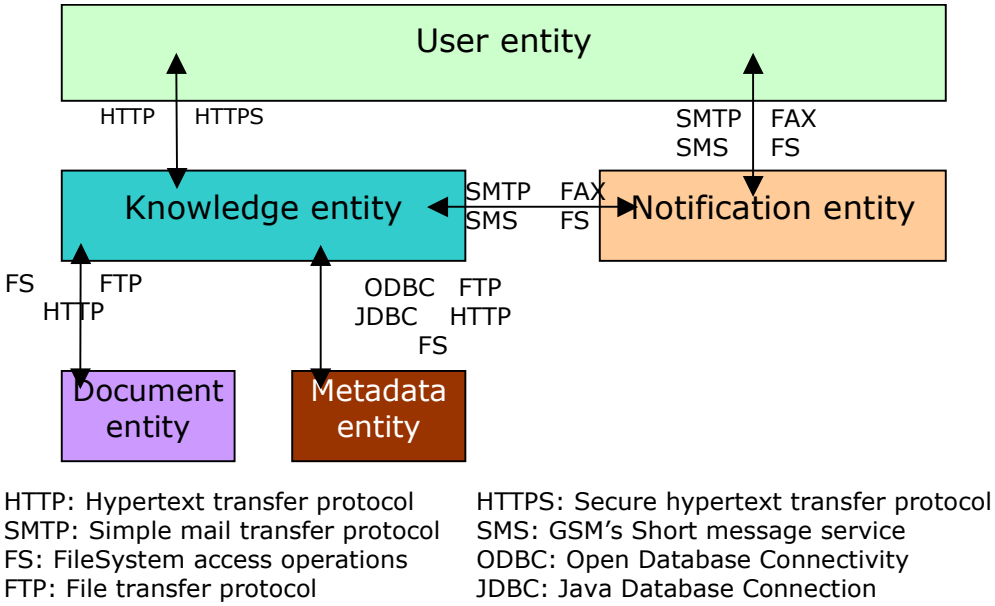


Figure 39. Possible system architecture

8.5.4 Refined entities

In the refinement of the functional model, we have also identified five different main classes of entities needed in an e-service [LLOR02a]. The difference between the refined entities and the entities described in section 8.5.1, Entities, is that we have found two different kinds of knowledge entities, server entities and knowledge entities. All entities are described in the following list:

- User entities: The entities that directly dialogue with final users or represent final users (human or machines).
- Knowledge entities: The entities that have the intelligence or knowledge inside the distributed system. They are divided into Server entities and Service entities.
    - Server entities: The entities that have the intelligence or knowledge related to the complete system.
    - Service entities: The entities that have the intelligence or knowledge related to the e-services inside the system.
- Notification entities: Entities that offer an interface for the sending of notifications among the rest of entities (normally user entities and knowledge entities). They usually have store and forward behaviour.
- Document entities: The entities that are in charge of the management of the documents needed by the different entities of the workflow system.
- Metadata entities: The entities that deal with the system metadata.

This entity classification will permit us to see each entity as a separate agent with a very specific functionality and use an agent platform to implement a system to provide e-services. This platform will offer us most of the needed functionality for monitoring and communicating the different entities collaborating in such a system.

Figure 40 shows the different kind of entities together with their relationship.

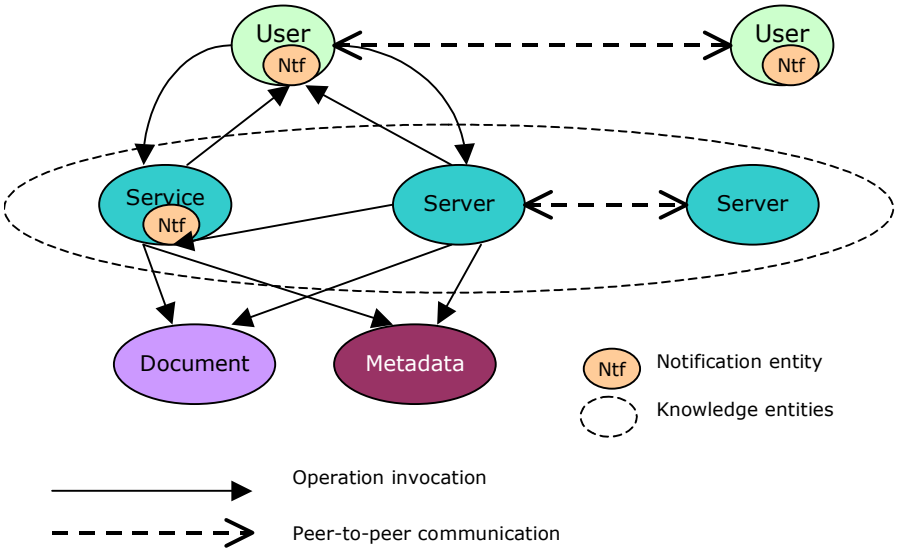


Figure 40. Entities and their relationship in the functional model

As before, this is a possible functional model. We can easily think about some different models: several notification entities per user entity, several document and metadata entities, document and

metadata entities accessible directly from user entities, etc. We can adapt the functional model to the needs of any specific system.

The server entities that appear in Figure 40 belong to different systems. It could be interesting to communicate them in order to interchange user profile information, e-service workflow control files and other information related to their operation. For example, if we have a system installed in collaborating companies or administrations, but they do not share information, we can interchange the information of the new e-services created by one of them. This scenario can be extended at will (as many companies as you want).

8.6 Summary of the methodology

Throughout section 8, we have described the different components of the methodology we propose for the definition of e-services. In this section, we want to summarise the methodology, giving a complete view of its components.

Table 27 shows this summary, giving, for each methodology component, the different types of them we have in the methodology, a brief description of the component and an example of what can be found inside each kind of component.

Table 27. Summary of methodology components

Methodology component	Type	Description	Example
Metadata and information	Complete system	Defines the general features needed inside a system offering e-services.	Contractual conditions, e-service guidelines.
	Generic e-service	Defines information inside a generic e-service.	User roles, documentation, steps.
	Specific e-service	Defines information inside a specific e-service.	Users participating, status of documentation, status of steps.
	e-Service workflow	Defines information needed for the workflow of the e-service.	Name of associated e-service, steps of the e-service, user roles inside e-service.
	Documentation	Defines the documents and information needed for the correct development of an e-service.	Contractual documents, documents interchanged during the execution of an e-service.
Workflow	XML	Defines the workflow of an e-service using XML language.	
	DAML-S / OWL	Defines the workflow of an e-service using DAML-S. The e-service is decomposed into processes, each of them described in terms of DAML-S Inputs/Outputs/Preconditions/Effects.	
Control and information flow	WFCi	Defines the workflow control information present in an e-service. This is achieved describing the way WFCi is generated.	Distributed control, centralised control.
	DOCi	Defines the documentation information present in an e-service. This is achieved describing the way DOCi is stored.	Centralised storage, distributed storage.
Functional model	Entities	Defines the different classes of entities and their relationship for describing an e-service system.	User entities, knowledge entities, document entities.
	Operations	Defines the operations provided by an entity to the rest of them.	Read, write, search, delete, sign.

9 Validation of the model: E-Services implementation

In this section we explain in detail how the concepts explained in previous sections can be applied to two different kind of e-services: Legal and administrative e-services [LLOR00a, LLOR02a] and collaborative editing e-service [LLOR03a, LLOR03b].

Furthermore, other possible areas of application of e-services are mentioned as future research lines, in section 11.

9.1 Legal and administrative e-services

This section describes the characteristics of electronic commerce of legal and administrative services. We have selected these services because they can be clearly mapped into the general methodology for the definition of e-services that we have presented on previous sections of this work.

9.1.1 Characteristics of legal and administrative services

We have studied in detail legal and administrative services to be able to describe them [LLOR00a]. We outline here some of their main features:

- There is a purchase of services. The result of these services (normally documents) is delivered after off-line development.
- Many official documents must be interchanged.
- There are business-to-consumer and business-to-business aspects.
- Payments can be requested in different moments (in advance, after service, periodically, etc) and several mechanisms can be used (credit card, cheque, bank transfer, etc).
- Confidentiality and non-repudiation is needed.
- Many actors are involved in the services. These actors have a lot of interaction among them.
- A legal service can be described as a series of steps. These steps may vary from service to service depending on different factors (more documents needed, more steps, more actors involved, etc), but an initial structure can always be defined.

9.1.2 Advantages of providing legal and administrative services through electronic commerce

In legal and administrative services, many documents are interchanged among the users involved. These documents are still often delivered by fax, courier, post, in person or even electronically, by using e-mail and ftp and/or web servers. If there is an error, the document must be resent, involving a waste of time and money (another fax, another courier). In case of an error in the document, the use of e-mail as document sending mechanism in legal and administrative services represents a clear advantage over the rest of sending mechanisms presented. The professional can resend the document from his own PC, after correcting the document, at low cost and, the most important, as many times as needed. E-mail, an electronic mechanism of document delivery, represents a first advantage we could find when providing these services in an electronic way.

Most of the steps of legal and administrative services have deadlines that lawyers or administrative consultants must control. If we have an electronic commerce application that controls service development, it could also control deadlines for each case of a lawyer or administrative consultant and notify the involved users that some actions must take effect.

Official documents normally have to be signed by several actors involved in the case. At the present moment, some laws and measures are being developed in order to accept digital signatures as the handwritten ones. In this situation, electronic documents digitally signed will have the same value as paper documents in front of administration.

As we have said, we can now send documents in an electronic way using e-mail or ftp. There are also centralized applications that control the workflow of legal and administrative services automatically. Our proposal is to define a distributed system that integrates all of these functionalities using electronic commerce, permitting that all users involved in a case had access to its information and status at any time, in a customized manner also integrating digital signature mechanisms when possible.

9.1.3 Business model of electronic commerce of legal and administrative services

The actors involved in legal and administrative services and the interactions that occur among them during the development of one of these services define the business model of these services [LLOR00a].

Figure 41 and Figure 42 show the business model of the legal and administrative scenarios, respectively. In these figures, the arrows represent the interactions among actors.

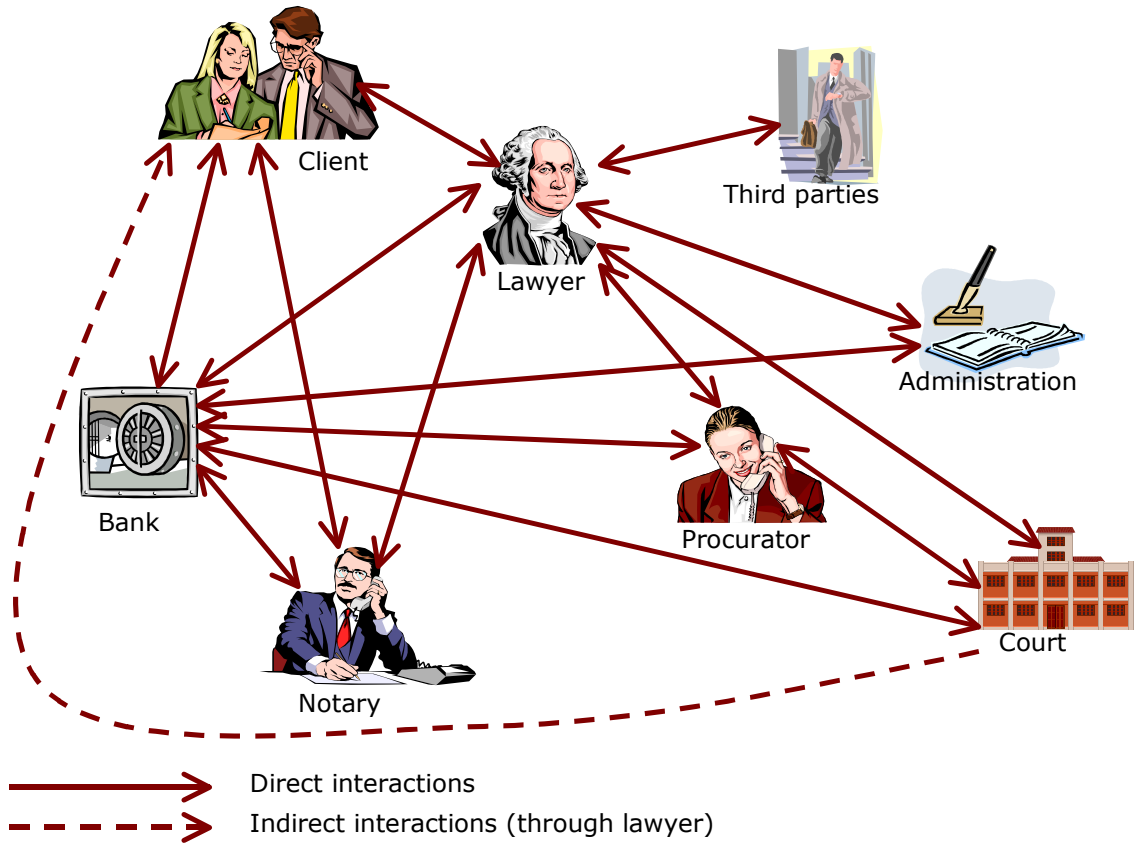


Figure 41. Actors in legal scenario

The main actor in legal services is the lawyer. All the flow of information of a legal case passes through him and has the control over the legal case.

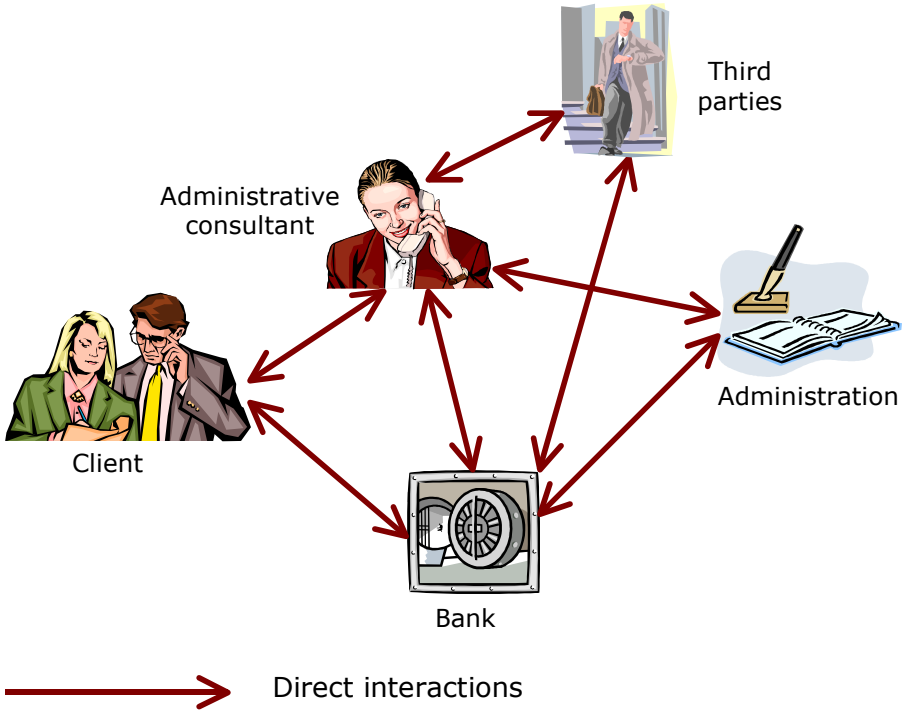


Figure 42. Actors in administrative scenario

In administrative services, the administrative consultant is the main actor. He collects the information and is in charge of sending it to the corresponding actor. The administrative services are a subset of the legal services, so, from now on, we will only explain legal services (they have much more relevance), but the features defined for them completely apply to the administrative services also.

9.1.4 Workflow in legal services

We have refined the description of the workflow of legal services during our research. This section shows the first workflow models for legal services that we defined. In later sections, where our methodology is mapped to legal services description, the selected approach for describing legal services workflow is explained in more detail.

Our first approach for defining workflow of legal services was based on the phases of a legal process. A legal process is divided into two different phases [LLOR00a]:

- *Preliminary phase*: This is a negotiation phase, where the client and the lawyer decide if a legal case is going to be developed.
- *Case development phase*: This is the main phase of a legal process. All legal actors (notary, procurator, administration, court and 3rd party) can be involved (it depends on the nature of the case which actors are really involved). This phase was initially divided into three parts: Document Acquisition, Procedures and Result.

We described the preliminary and case development phases from a business model point of view, that is, what the actors see during the legal service development.

Figure 43 shows graphically what happens on the preliminary phase from the business model point of view.

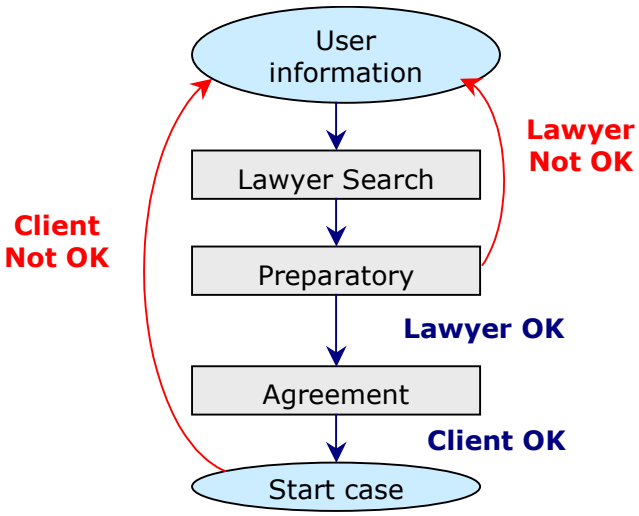


Figure 43. Preliminary phase from a business model point of view

The preliminary phase defines the negotiation for contracting a legal service that occurs between the client and the layer. The client searches a lawyer and sends him an exposition of the problem. If they come to an agreement, the lawyer starts a case to provide the service requested by the client. If they do not come to an agreement, the client can look for another lawyer going back to the preliminary phase.

Next phase in a legal process is the case development phase. The main actor appearing in this phase is the lawyer. He plays a controller role inside this phase, as he manages the legal process by acting over some or all of the following components of the case:

- State of the flow of execution: He starts, finishes or adds steps to the case.
- Actors involved: He adds new ones when necessary (these new actors can be notaries, procurators, other lawyers, other customers, courts, administrations or third parties). The sender and the receiver of the information interchanged in the case define the actors that have to be involved inside it.
- Information sending and retrieval: Almost all the information of the case must be received or sent by the lawyer (for example, the customer receives a document from the court and then he must send it to his lawyer). This information can be digital documents, forms, paper documents or anything necessary for the case.

As we have said, the case development can be divided into three consecutive sub-phases: *Document Acquisition* in which the lawyer requests the necessary documents to start the case, *Procedures* in which the lawyer sends these documents to court or administration (depending on the nature of the legal service) and finally the *Result*.

Figure 44 shows schematically what happens on the case development phase from the business model point of view. *Start Case* in the case development phase corresponds to the last stage of the preliminary phase.

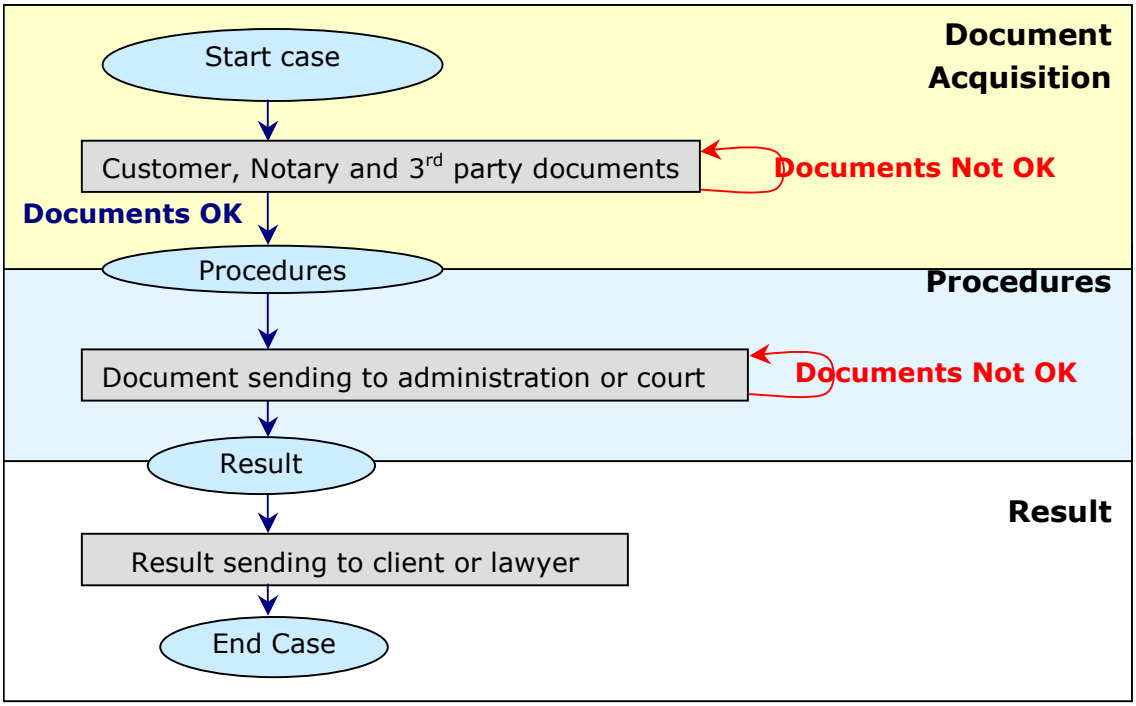


Figure 44. Case development workflow

- *Document Acquisition*: The lawyer requests the documents he needs to all the actors involved in the case (at this point the actors can be: Customer, notary and 3rd party).
- *Procedures*: The lawyer sends the documents directly or through a procurator to court or administration (it depends on the specific legal service).
  Court or administration carry on with the procedure depending on the legal service (the taken steps differ from legal service to legal service). They usually have to contact with different actors.
- *Result*: The court or administration sends the result to the lawyer (directly or through a procurator) or the customer (it depends on the specific legal service). The case is finished. After the case is finished, it can be continued if the parties do not agree with the final result, but this corresponds to a new legal case, where new steps and actors may be involved.

9.1.5 Mapping of the methodology for the definition of e-services to legal and administrative services

In this section it is described in detail how we have applied the methodology we have presented in section 8, to the specific case of legal and administrative services.

9.1.5.1 Metadata in legal and administrative services

In our methodology for describing e-services, we have identified different levels of metadata: System, generic service and specific service. Apart from them, we also have the documentation present in the services. These different levels of metadata are also present on legal or administrative services. Table 28 shows some of the metadata we can find for legal services on each of these levels together with its description.

Table 28. Metadata inside legal and administrative services

Metadata level	Metadata name	Description
System	Users	Lists of the different kinds of users of the system. For each user, many contact information can be defined, for example, their address, e-mail, etc.
	Services	Services that clients can contract, after looking for a lawyer that provides it.
Generic service	States or steps	Stages where the service can be during its development.
	User roles	User roles involved in the service.
	Documents	Documents interchanged among user roles inside the different states.
Specific service	Status of states or steps	Status (in progress, not started, skipped, etc.) of the steps that are being taken for the development of the service.
	Users	Users of the system that participate in the service. They are usually associated to the user roles defined in the generic service corresponding to this specific service.
	Status of documents	Status (pending, sent, etc.) of the documents interchanged among users of the service.
Documentation	Documents and information	The different documents and information interchanged among the users of the services. They can be digital or not.

9.1.5.2 Workflow definition using DAML-S

In order to continue with the validation of our model, we have to define which is the workflow of a legal service. Each legal service can have a different series of steps or dependencies among these steps.

As an example, Figure 45 shows the definition of Input/Output/Preconditions/Effects (IOPE's) of one of the steps, Document Acquisition, of specific legal service, a divorce. The inputs represent the information needed for executing the service, in this case, they are documents provided by the client user role. The effects of the process execution are the documents that the lawyer user role has to receive. The number of effects will depend on the number of lawyers involved in the e-service. We have two more inputs, the client user role and the lawyer user role (this one can appear from 1 to n times, as more than one lawyer could be involved inside a legal case). We also have a precondition, which affects to client user role, as he must be plaintiff of the legal service for this process to be executed.

On Figure 45 the interchange of information occurring between the client (that acts as plaintiff in the divorce) and his lawyer(s) is represented. They are the sender and the receiver of the information respectively. To distinguish between the same document but owned by a different person (in this case, lawyer or client), we have added the user role name to the name of the document.

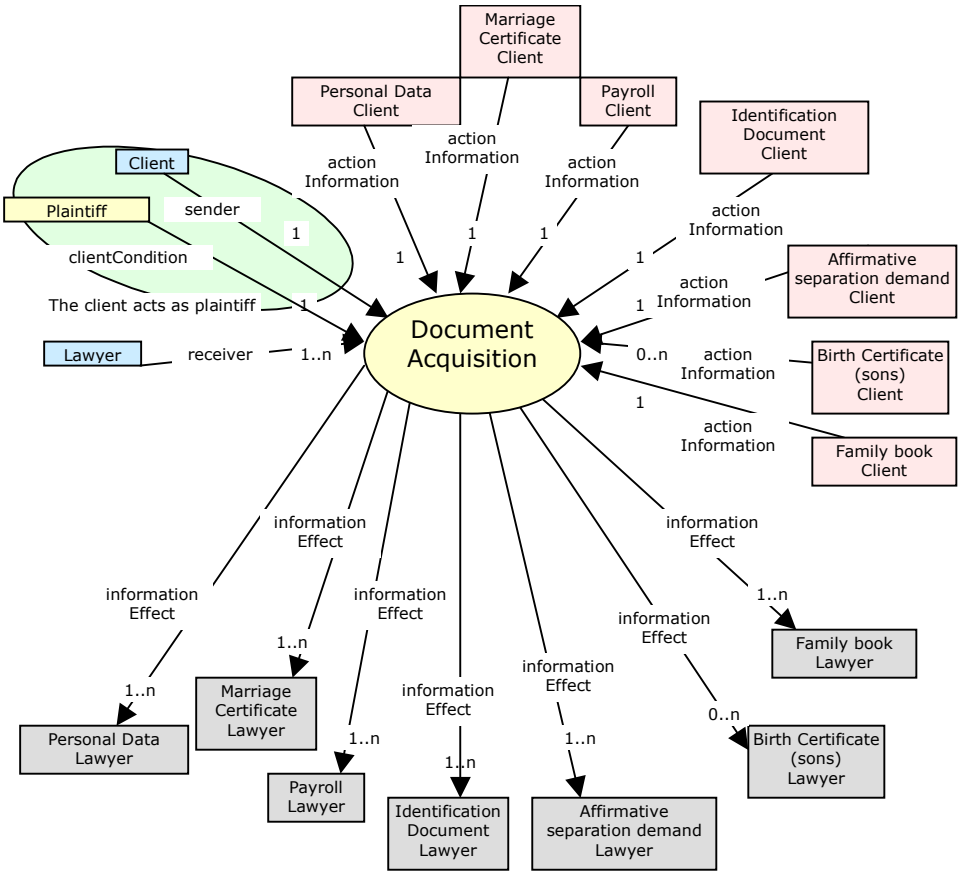


Figure 45. Document Acquisition description of DAML-S IOPE's

9.1.5.3 Functional Model

Figure 46 shows the functional model for legal services [AREA00a, LLOR02a]. We have a centralised server, which offers some part of the functionality to the user entities. This server is the server entity defined in the general functional model of the methodology. The rest of the needed functionality is offered by the service entities, which are created by the server entity when an agreement between a user entity of type client and a user entity of type lawyer is done. After this, the user entities involved in the service interact with the service entity, but is the lawyer who takes a controller role. The knowledge entities of the system (server and service instances) encapsulate the access to document and metadata entities present in general functional model.

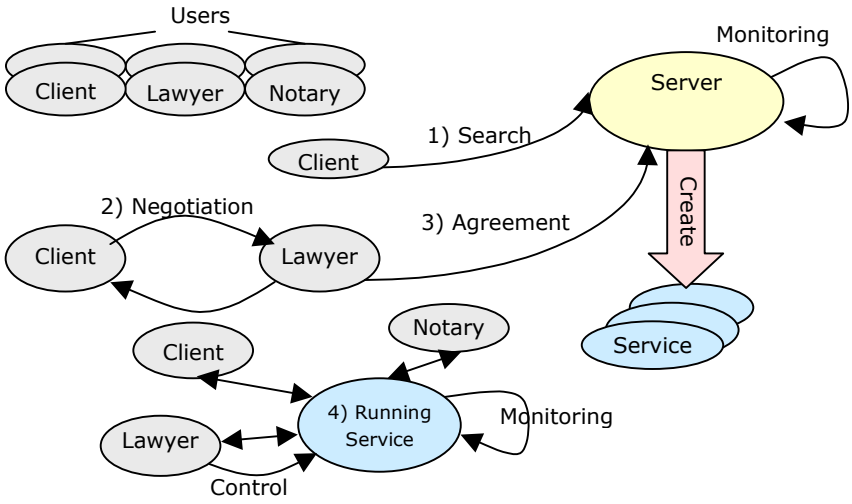


Figure 46. Legal services functional model

9.1.5.4 Control and information flow

The control and information flow inside legal services completely depends on the implementation of a system for providing them electronically. In this way, we will describe in the sections corresponding to the implementation of legal and administrative e-services, the solution we have adopted for the control and information flow.

Nevertheless, as the lawyer is the responsible of controlling the workflow of the service, and we can have several of them inside a service, the best-suited solution seems to be centralised workflow control where all lawyers can access. We will go back to this in the implementation sections.

The document information (DOCi) storage will also be explained in detail inside the implementation section, but, as we have already said, the documents and information interchanged in a legal service must go from one user to the other. In this case, the best solution could be that every user stores locally any document or information addressed to him.

9.1.6 Implementations of legal and administrative services

In this section we present two different implementations of legal and administrative services done inside the TRiAls in the Domain of Electronic commerce (TRADE) [TRAD04a, TRAD04b] project and the newTRADE project, sub-project of the Área 2000 project [AREA00a].

9.1.6.1 TRADE project

The TRADE (TRiAls in the Domain of Electronic commerce) project was a European Commission co-funded project inside the ACTS programme (ACTS 328) [ACTS04a, TRAD04b]. This project started in March 1998 and finished in March 2000. Inside this project we implemented a distributed system where lawyers and administrative consultants offered their services by electronic means. The clients of the system were able to contract these electronic services.

The development of a distributed application for offering legal and administrative services through electronic means inside TRADE project, gave us a very good starting point for our research on

e-services. The idea of defining a methodology for describing e-services (electronic commerce of services) was based on the experience we obtained in this project on the development electronic commerce applications for complex services, like legal and administrative ones.

Inside the TRADE project not only were implemented legal and administrative scenarios applications but also there were some other applications and building blocks. Those building blocks and applications are not relevant for this research work, so we are not going to talk about them.

9.1.6.2 TRADE application for legal and administrative services

Using the specifications made for legal and administrative services in TRADE project we developed a prototype application [LLOR00a, LLOR02a].

We started the prototype development in mid-98 and the first version was running at the beginning of 1999. More functionality has been added in consecutive versions.

We have validated the application with real cases. They are stored in the application database, but the lawyer can create new services as needed. The legal cases we have used for this test are:

- Divorce
- Dismissal
- Ordinary labour proceeding
- Proceeding involving claims less than a standard amount
- Undefended separation
- Contentious separation

The administrative cases used to test the application are the following:

- VAT declaration
- Personal income tax
- Vehicle number plate registration
- Sick leave certificate
- Driving license renewal

They are stored in the application database, but administrative consultant can create new ones.

The application and some demonstrations are available in [TRAD04a]. Demonstrations allow users to familiarize with the program before using it.

9.1.6.2.1 Architectural model

The architecture of the system implemented in the TRADE project was the following: We had a central server, called the TRADE server, and the different actors connected to this server when they wanted or when they were required by notifications. The TRADE server generated notifications under different situations. For instance, the user received a notification when he received a document, when he was requested to send a document to another user or when a new client wanted to contract a new service. In the last case, the provider received a notification from the TRADE server.

We divided actors into customers and suppliers, depending on if they provided or received services from other actors of the system. Lawyers were at the same time customers and suppliers, as they

played the two roles depending on which actor they were working with. Also administrative consultants played the two roles, customers and suppliers, inside administrative services.

Figure 47 shows the actors present in the business model separated into customers and suppliers, depending on their role on the services [LLOR00a].

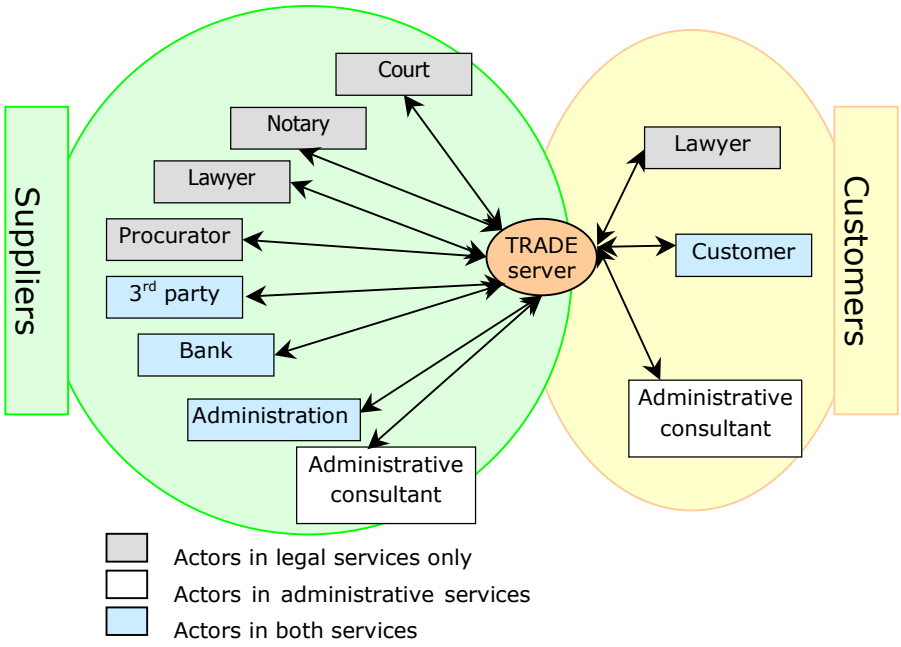


Figure 47. TRADE project architectural model

9.1.6.2.2 Legal and administrative services preliminary phase

In this project [LLOR00a], we implemented the two different phases present inside legal services, the preliminary phase and the case development phase. We have described the business model of these two phases, but we also want to describe these phases from a different point of view, the communications one, where it is shown how the actors involved in a service interchanged information among them through the TRADE server.

In Figure 48 and Figure 49, the schema of the communications that happen between the client and the lawyer through the TRADE server are shown. We have separated the schema into two figures, as many communications occur. The numbers represent the order of the actions performed.

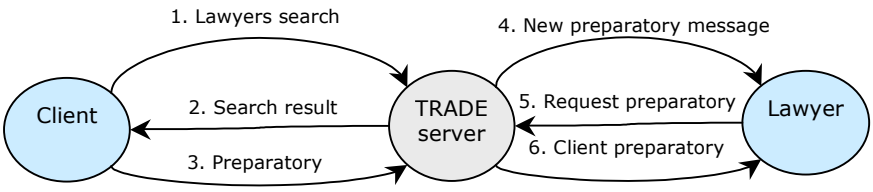
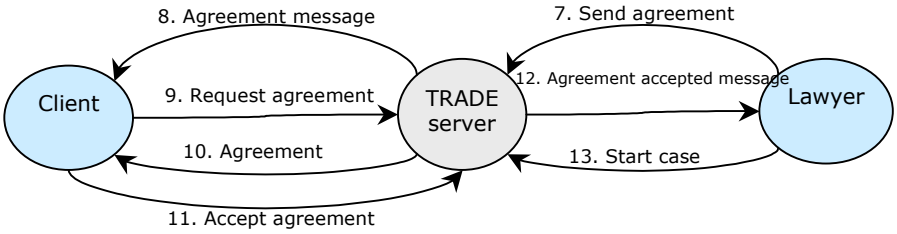


Figure 48. Lawyer search and preparatory

1. The client requests the lawyer search to the TRADE server.
2. The TRADE server returns the list of lawyers who accomplish the selected criteria.
3. The client selects one of the lawyers and sends him an explanation of his problem.
4. The lawyer receives a message from the TRADE server indicating that he has a new explanation to read.
5. The lawyer requests the preparatory to the TRADE server.



6. The lawyer receives the preparatory.

Figure 49. Agreement and start case

7. The lawyer sends the agreement to the client's exposition.
8. The TRADE server sends to the client a message to indicate that the lawyer's agreement has arrived.
9. The client requests the agreement.
10. The TRADE server sends the agreement to the client.
11. The client reads and accepts the agreement
12. The lawyer receives a message from the TRADE server indicating that the client has accepted the agreement.
13. The lawyer starts a new legal case.

Figure 50 shows graphically the interchanges of information that occur in the preliminary phase among the entities defined in the functional model of our methodology. These interchanges of information are represented by the operations invoked by each entity. These operations were defined on section 8.5.2 [LLOR01a]. The description of the steps from 1 to 7 is the following:

1) The client searches all the professionals accomplishing some criteria calling to the knowledge entity. The server entity calls the search operation from the metadata entity. The result is returned to the client.

2) The client selects one of the lawyers returned by the knowledge entity and sends him an explanation of his/her problem. This information is stored by the metadata entity and a result is returned. If the result was positive, the knowledge entity sends a notification to the lawyer's notification entity.

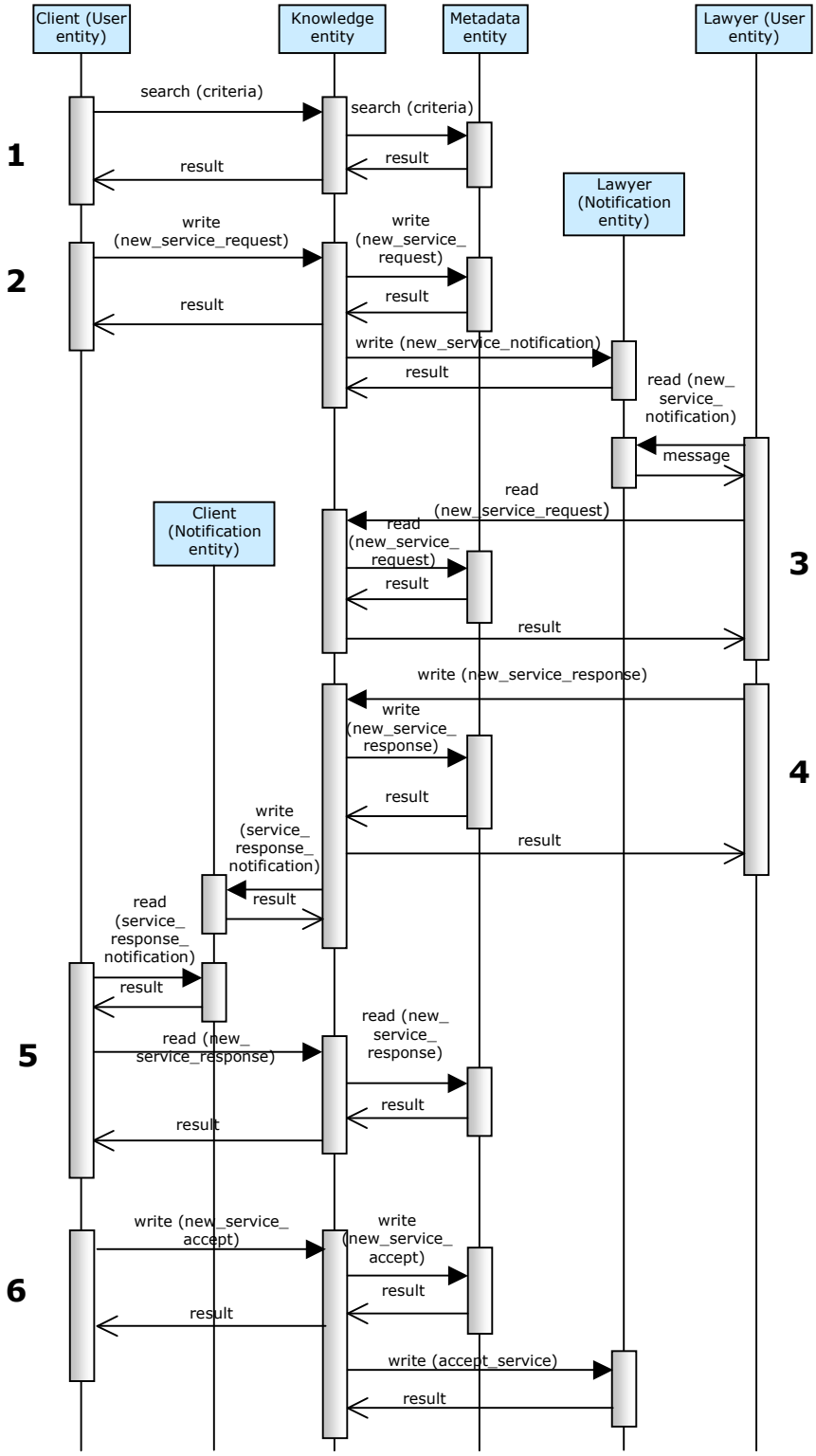
3) The lawyer reads the notification and asks the knowledge entity about the information sent by the client.

4) The lawyer writes a response for the client (in this case, it is a positive response). This response is stored in the metadata entity by the knowledge entity and a notification is sent to the client's notification entity.

5) The client reads the notification and asks the knowledge entity for the lawyer's response.

6) The client agrees with the response sent by the lawyer and he accepts the e-service. The knowledge entity stores the acceptance in the metadata entity and sends a notification to the lawyer's notification entity.

7) The lawyer reads the notification and starts the e-service. In order to start the e-service; the e-service information, the users and the documentation must be created. All this information is stored by the metadata entity.



(Continuation of the diagram)

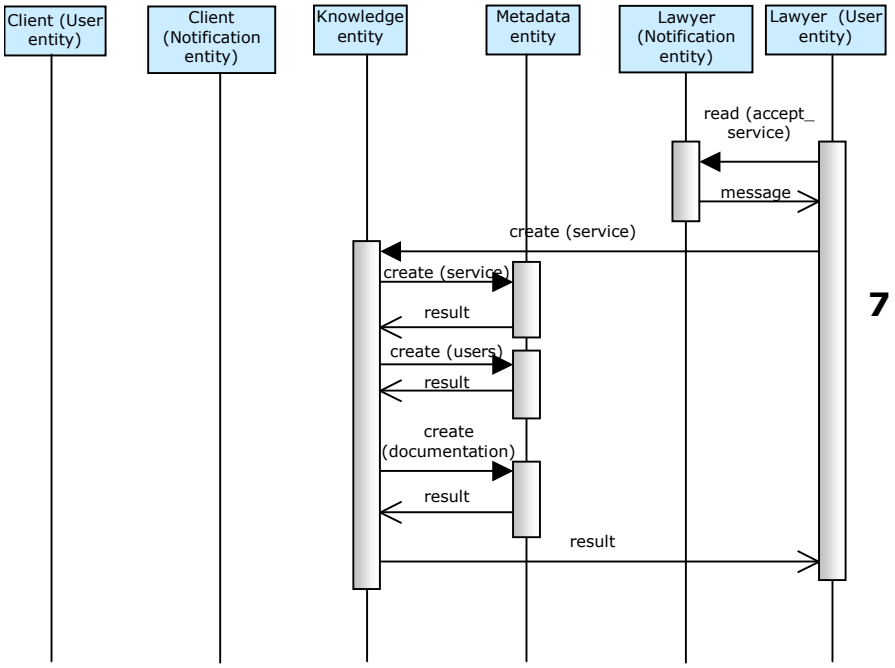


Figure 50. Interchange of information to start an e-service

9.1.6.2.3 Legal service case development phase

In Figure 51 it is shown the communications that occur in the document acquisition sub-phase of the case development phase between client and lawyer, again through the TRADE server. In the schema of the communications that happen between the client and the lawyer through the TRADE server are shown. The numbers represent the order of the actions performed.

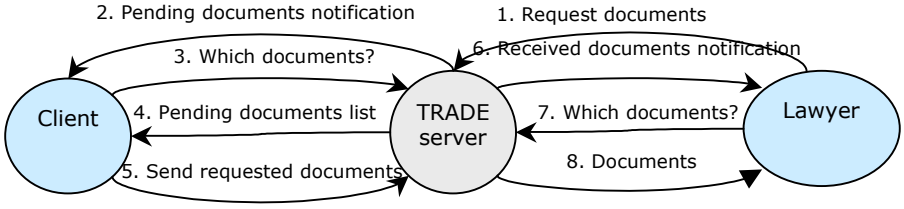


Figure 51. Document Acquisition communications

1. The lawyer requests some documents to the client in order to start the legal case.
2. The client receives a message from the service agent, indicating that he has to send some documents to the lawyer.
3. The client requests the information of the documents he has to send.
4. The service agent sends the information to the client.
5. The client sends the requested documents to the service agent.
6. The service agent sends a message to the lawyer telling him that the client has sent the requested documents.
7. The lawyer requests the documents to the service agent.
8. The service agent sends the documents to the lawyer.

The communications diagram for the Procedures and Result sub-phases are very similar to the Document Acquisition one. One actor requests documents or information necessary to continue the development of the case to another actor through the TRADE server.

9.1.6.3 Área 2000 project: newTRADE

Área 2000 is a project financed by the Spanish Ministry of Science and Technology [AREA00a, LLOR02a]. This project started on November 2001 and lasted for 2 years, until November 2003. Inside this project, several sub-projects have been developed, in completely different areas. We centre here only on the area relevant for this research work, newTRADE sub-project.

In newTRADE, we have applied the methodology described for performing a new implementation of legal and administrative services. Some of the characteristics of this project are outlined in the following sections.

9.1.6.3.1 newTRADE functional model

Figure 52 shows newTRADE functional model [LLOR02a]. It is very similar (in fact, it is the same) as the functional model defined generically for describing legal services.

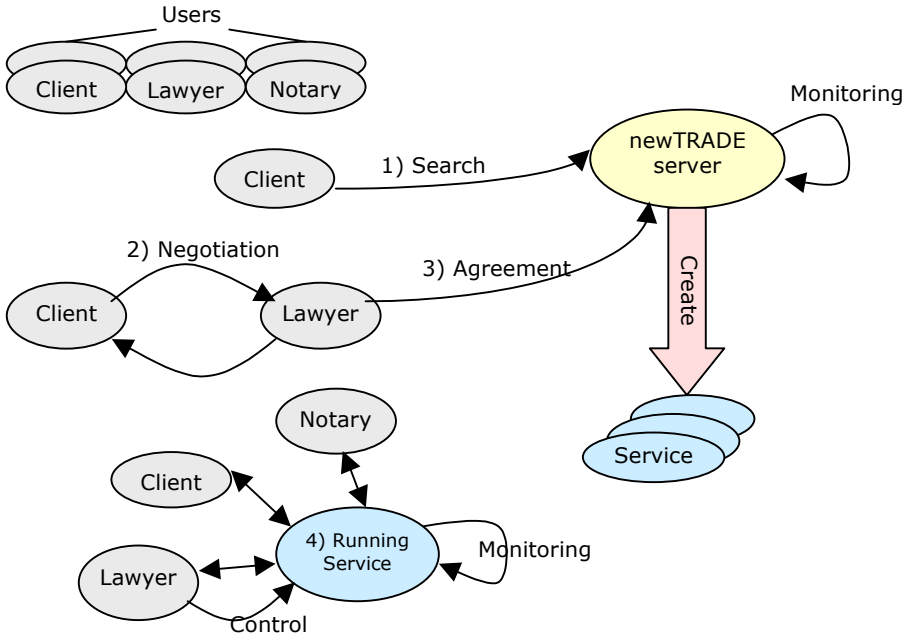


Figure 52. newTRADE functional model

9.1.6.3.2 newTRADE system architecture

The lawyer is responsible of controlling the workflow of the service. As we can have several lawyers inside a service and each of them can have the possibility of controlling the workflow, we have decided to implement in newTRADE a centralised workflow control, as the distribution of it would lead us to an undesirable level of complexity. The lawyer, playing the role of workflow controller, is responsible of deciding when a service enters into the next step, when the current step has finished, if one step of the service has to be skipped and so on. Any lawyer inside a service,

unless they agree on appointing one of them as responsible, is able to modify the state of the service.

For the DOCi storage, we have chosen a mixed solution. The documents pass through a centralised location (newTRADE server) in order to reach its final destination (the receiver user), but the receiver is who has to the responsibility of storing the documents on his local system.

Figure 53 shows the architecture for newTRADE.

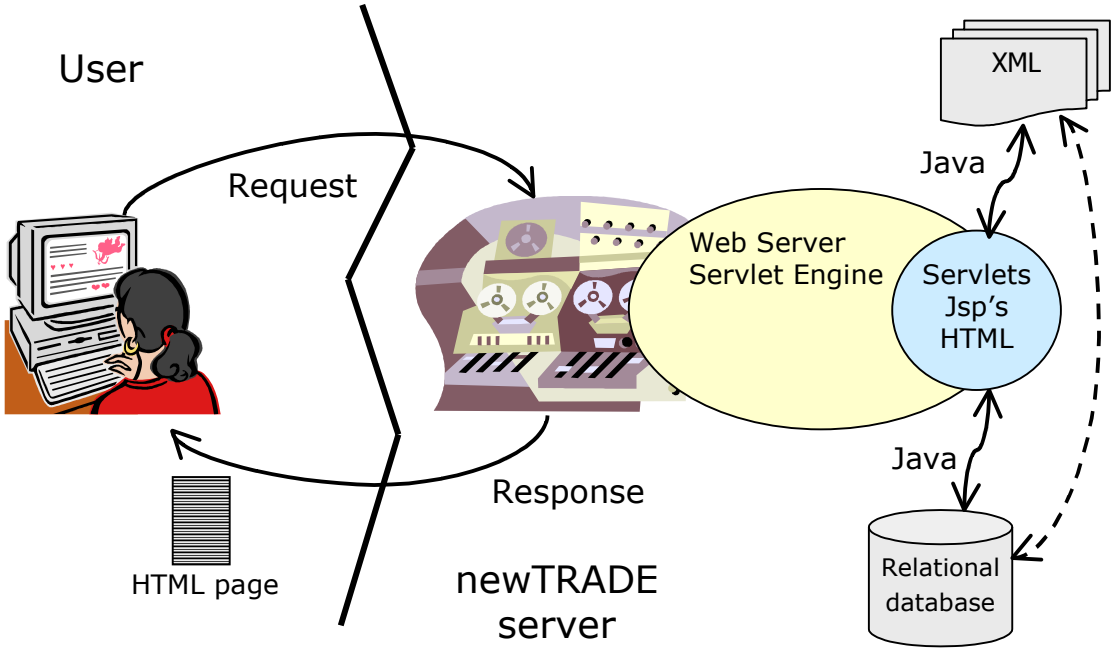


Figure 53. newTRADE architecture

9.2 Collaborative editing e-service

Once we had validated our methodology with legal and administrative services, we observed that it was needed to perform this validation with a completely different kind of service. If we were able to apply the methodology to services with very different characteristics we could better justify its relevance in the definition of electronic commerce of services. The selected service was collaborative editing. This section describes the characteristics of electronic commerce of collaborative editing services, applying the methodology to its definition. As we have already said, we selected these services because they are completely different from legal and administrative ones, in order to validate the general methodology developed for the definition of e-services.

9.2.1 Characteristics of collaborative editing

In collaborative editing [LLOR03a, LLOR03b], several users are involved in the production of a document, but at a given moment, only one user has the permission to manipulate it. At the end of the contribution, the next user can access to the document to perform more manipulations over it. The moderator determines who will be the next user in accessing the document. He makes this decision based on the turn requests performed by the participants that can make modifications over the document present in the service. Moreover, mechanisms to warn users when they have permission to manipulate the document have to be provided.

We do not impose any restriction on the type of document being edited, it can be either text or image or whatever, i.e. any kind of multimedia document. We are only concerned here in the control of the modifications performed by the different users over this document, but not in the document itself.

The degree of dynamism of the collaborative editing services is determined by the fact that so much the number of users that participate as the order in which they invoke the operations over the document are unknown. What we do know is the operations that each user role can invoke and the states in which the edition can be from different points of view (the user, the document, etc).

9.2.1.1 Sample use case: Collaborative editing of multimedia documents

The research on collaborative editing of multimedia documents can be viewed from two different points of view: One is the collaborative editing service itself, and the other one is the document being edited. We can find metadata in both of them, but we mainly focus here on the metadata related to the document.

Starting however from the first point of view, the service associated to collaborative editing [LLOR03b], we can say that we make use of a workflow to control the process. This workflow defines the order in which the operations permitted in collaborative editing can be executed and the user roles that can execute them.

In the edition process, several users may be involved. They may add new content to the document, make comments to the content of the document or only view the document and the modifications performed by other users. A special user role of the service, the moderator, with the help of the workflow, controls the order in which the operations are called, because they cannot be arbitrary.

All information related to workflow used to control the service, the users involved and the operations permitted on each moment can be collected as metadata. This metadata allows us to extract knowledge to refine the use of collaborative editing service and also in the definition of new complex services. Depending on what the metadata references, the way of extracting it may differ.

For example, we could have obtained metadata from users by registering them in the service, but we do not know the order of operations until they are performed. So, the metadata related to operations execution should be extracted after executing them.

The second point of view is the multimedia document being edited and the information associated to it. It is very important to represent the different parts in which the document is structured, together with the metadata needed to identify each of these parts and other related information (date of creation / modification, user that created this part of the document, modifications history, etc). This information will permit us to deal with version control of the multimedia document, search the content generated using the metadata associated to it or manipulate intellectual property rights of the different parts of the document.

9.2.2 Users participating in collaborative editing

The users that participate in collaborative editing can be grouped into three main roles: Editor, commentator and viewer. There is another user role in this service, the moderator, which has the control of the service (assigns turns, initiates or finishes the edition service, etc). Each role has specific features as each can perform a different set of operations over the document, as defined in [LLOR03a]. We summarise their characteristics next.

The viewer role, as its name indicates, can only view the document. This role cannot perform modifications over the document, but he can see all modifications and comments made by other users participating in the edition. Maybe other communication channel should be added for permitting this user to advise the rest about any problem with the document being edited. A user cannot change his role during the edition.

The commentator role can view the document and also make comments over its content. These comments can then be added as content of the document if a user playing the editor role accepts them. This role is a superset of the viewer role, as it has the same operations together with the functionality needed for making comments.

The editor role can view the document and make modifications and comments over the document. He can of course accept or reject comments performed by other editors or commentators. This user role has the complete functionality for performing modifications over a document. It is a superset of the commentator role and also the user role with the full edition operation.

The moderator is the user role responsible for the control of the turns and, in general, of the whole collaborative editing service instance. This user role can start and finish the edition and decides which is the next user that will be able to access to the document (either for making a comment or a modification), based on the requests performed by the edition participants corresponding to the editor and commentator user roles.

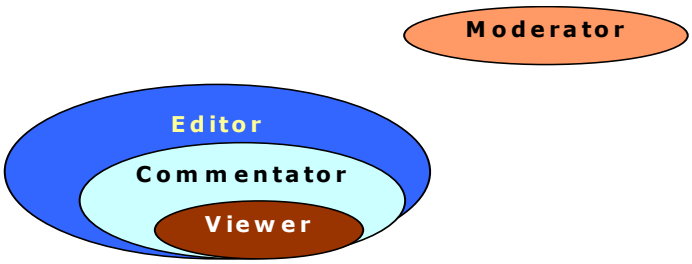


Figure 54. Graphical representation of the relationship among collaborative editing user roles

As Figure 54 shows, editor role is a superset of commentator in the same way that commentator is a superset of viewer. The moderator role has a completely different set of operations, and this is why it is completely separated from the other roles in Figure 54. For the simplicity of the model, in a specific instance of the collaborative editing service, a user cannot play at the same time an editor/commentator/viewer role and a moderator role. However, we can easily think about an implementation of this service where the user acting as moderator is at the same time viewer, editor or commentator of the service. Anyway, a user can be moderator in one instance of the edition service and a different user role on another instance of the collaborative editing service at the same time.

9.2.3 Mapping of the methodology for the definition of collaborative editing e-service

The rest of this section explains in detail how our methodology is used to define the collaborative editing e-service.

9.2.3.1 Metadata in collaborative editing e-service

Our methodology contemplates different levels of metadata needed to define an e-service. In each level, we can find information related to the different aspects of the service: The complete system, collaborative editing e-service in general (generic collaborative editing e-service), or a specific instance of the collaborative editing e-service (specific e-service). For some of these aspects, we have identified the metadata present in the collaborative editing service.

In the category of complete system, the information of which specific service instances are active has to be stored, together with the list of users that can join on each of them. The moderator of each specific instance determines this information. This list allows to a user the selection of an active specific instance from the ones he can access.

Metadata present in complete system is shown in Table 29.

Table 29. Metadata present in complete system

Metadata	Definition
Active instance	Contains the list of active collaborative editing services
Users in instance	Stores the list of users that can access to an active service together with their access rights
Users	Contains a list of all users in the system

In the category of generic services we have identified the following metadata: States in which the service can be, the user roles that can work in the service, and the access rights over the document being edited. This information is extended in Table 30, Table 31 and Table 32.

Table 30 shows the list of possible states for generic collaborative editing e-service.

Table 30. States for collaborative editing e-service

State	Definition
Idle	The edition has not been started yet
Wait Turn Request	The moderator waits for token requests performed by the editors and the commentators of the document
Document Edition	The user with the token can perform operations over the document.
View Edition	The users without token can view the operations performed over the document, but no modification is allowed

Table 31 shows the user roles present in the generic collaborative editing e-service.

Table 31. User roles inside collaborative editing

User roles	Definition
Moderator	User who has control over the token. He decides who is the next user that can perform operations over a document
Editor	User that can get turn to perform modifications over the document being edited
Commentator	User that can make comments to the document, but it is needed that an editor accepts these comments in order to add them to the document
Viewer	User that is able to view the document, but he cannot perform modifications over the document during the edition

The access rights that users can have over a document in collaborative editing are listed in Table 32.

Table 32. Access rights present in collaborative editing

Access Rights	Definition
View	Indicates that a user can view the document
Edit	Indicates that a user can perform modifications over the document
Comment	Indicates that a user can make comments on the document

In the specific e-services categories, we can also find several metadata, described in Table 33.

Table 33. Metadata found in specific e-services

Metadata	Definition
Token	Identifies the user that can make modifications/comments over the document
Turns	List of users that have requested the turn for modifying/commenting the document
Editors	List of users that can make modifications over the document
Commentator	List of users that can make comments over the document
Viewers	List of users that can only view the document
Document Modifications	The history of the modifications performed by the users over the document being edited

Finally, regarding to documents and information inside collaborative editing, the only document needed is the one being edited by the different users inside the e-service. However, in collaborative editing e-services, there is also another kind of metadata related with the document produced. In the next section we explain in more detail, the metadata we can find inside this document.

9.2.3.1.1 Metadata associated to documents created in collaborative editing e-service

In collaborative editing, several users collaborate to produce a multimedia document. Nevertheless, we can distinguish among several levels of metadata inside a multimedia document: metadata affecting the whole document, metadata applying to different fragments of the document and even metadata applying to different contents inside a fragment. These concepts were also already defined in Open Document Architecture (ODA) standard [ITUT95a] and they completely apply to multimedia documents created with the collaborative editing service (and, of course, to any other kind of structured document).

To illustrate the different levels of metadata present in different kinds of content, Table 34 and Table 35 show the metadata associated to textual and image content inserted inside a document. The metadata is separated depending on its application inside the document.

The definition and storage of the metadata associated to the documents generated with the collaborative editing service can have several applications.

One application for document metadata is to maintain the history of the document, indicating which user added/modified each part of it. This will help in the implementation of version control for multimedia documents. Another application is to maintain separate information for each fragment composing the document. For instance, if a user adds an image to the document, maybe this image has Intellectual Property Rights (IPR) associated to it. The underlying idea is to be able to determine the IPR of a generated document, based on the IPR of the fragments added to it. The idea of organising documents into fragments is not new, but it is still valid as there is no a standard way of doing it and it depends on the application creating them. One more application could be the construction of an e-service for searching content based on the document's metadata generated when using collaborative editing e-service.

Table 34. Metadata inside a text fragment of a document

	Metadata Name	Value
Document General Metadata	Document ID	XYZ
	Creator	User B
	Creation Date	2003/03/18
	Document IPR	Defined by user B
Fragment General Metadata	Fragment ID	XXXX
	Fragment IPR	Defined by user A
	Creator	User A
	Creation date	2003/04/24
	Content type	Text
Content Dependent Metadata	Creator	User A
	Creation date	2003/04/24
	Number of words	25
	Number of characters	150
	Character set	ASCII
	Etc…	

Table 35. Metadata inside an image fragment of a document

	Metadata Name	Value
Document General Metadata	Document ID	XYZ
	Creator	User B
	Creation Date	2003/03/18
	Document IPR	Defined by user B
Fragment General Metadata	Fragment ID	XXXX
	Fragment IPR	Copy
	Creator	User A
	Creation date	2003/04/24
	Content type	Image
Content Dependent Metadata	Creator	Sebastião Salgado
	Creation date	1993
	Height	11
	Width	15
	Etc…	

9.2.3.2 Workflow control and definition using DAML-S

Moderator controls the flow of information in the e-service. He decides which participant will be the next to access to the document, based on the turn requests made by the participants. The

number of participants can be unknown and also the sequence of steps that will take place in order to reach the final objective of the e-service: the edition of a document thanks to the contribution of several participants. The workflow definition of the collaborative editing process can be seen from different points of view. The first one is the user's point of view. Each user inside the collaborative editing has a personalised workflow depending on the operations that he or she performs over the document. This workflow is more or less independent from the rest of users workflow. Nevertheless, some coordination is needed because only one user is allowed to edit the document at a time. This one should be named *Collaborative editing workflow by user role*. The second one is the document's point of view. Based on the operations that users perform over the document, it can be in a different state, for example, in edition. This one could be named *Collaborative editing workflow by document*. The last point of view is the service set up point of view. In it, the moderator communicates with the collaborative editing system in order to create a new collaborative editing service. This one could be named *Collaborative editing set up workflow* [RFC02a].

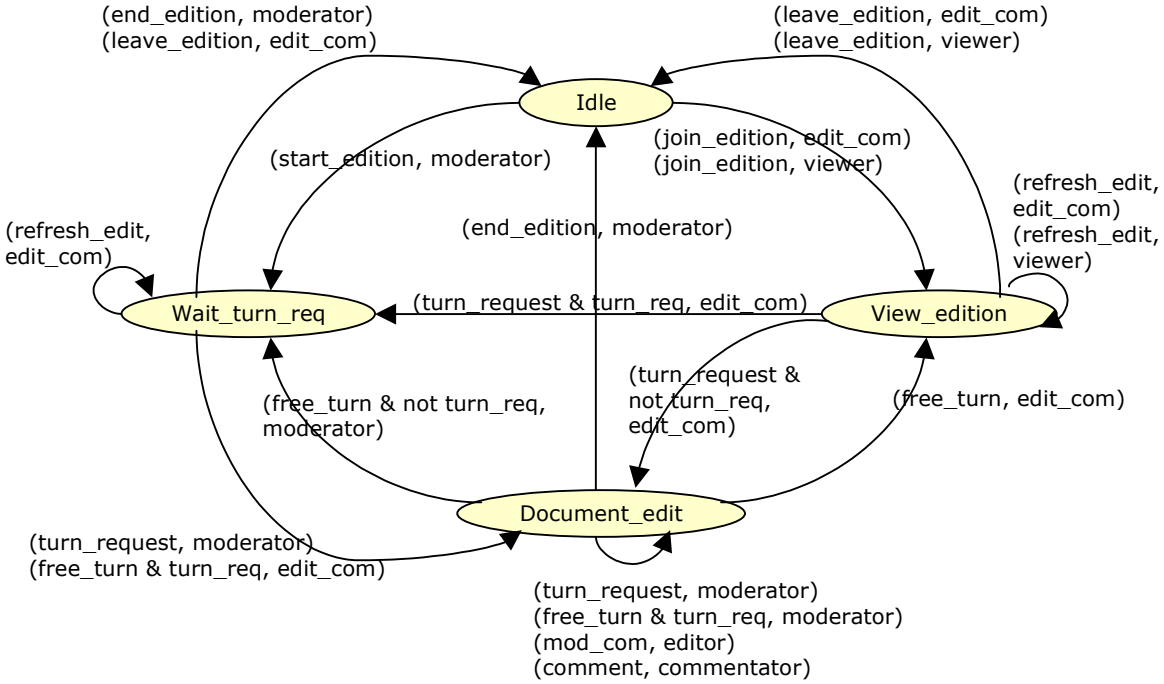


Figure 55. Collaborative editing e-service states

Nevertheless, although the workflow of the collaborative editing is unknown, we have detected the processes that can form part of it. Afterwards, we have defined them using DAML-S determining the IOPEs (Input/Output/Precondition/Effect) for each process. These processes have some kind of relationship among them (a document cannot be modified before opening it) and that is what we want to control by defining them.

For the determination of these processes, we first identified the states where the edition of a document can be depending on the user role. Then, we defined the transitions among these states depending on the operation performed over the document and the user role. Figure 55 shows the different states in which a document edition could be (Idle, View_edition, Wait_turn_request, Document_edit) and the transitions between them. The text over the arrows indicates the operation performed over the document (for example, Start_Edition) and, after the comma, the user role (for example, Moderator) that performs it.

All user roles are combined here into one state diagram, but in Appendix B, Collaborative editing e-service states, the different state diagrams for each user role can be found. Moreover, in this diagram, the operations considered are only related to the edition of the document. Other operations have to be defined for the administration of the service. For example, indicate which users can access to the document and their permissions, etc.

Table 36. Processes inside collaborative editing

Process Name	Operation meaning
Start edition	The edition of the document starts
Join edition	A commentator, viewer or editor enters in the edition service
Leave edition	A commentator, viewer or editor leaves the edition service
End edition	The edition of the document ends
Turn request	A user requests turn for modification. This causes that moderator receives a turn request and decides if the turn has to be given to the user or if he has to wait for its turn
Refresh edition	The view a user has over a document is refreshed
Free turn	The turn is freed. This operation occurs when the token is freed by the user who had it
Modification	Perform a modification over the document
Comment	Perform a comment over the document

The processes in Table 36 can be described in terms of the Input/Output/Preconditions/Effects as defined in DAML-S. Figure 56 shows the graphical representation of the Start Edition process.

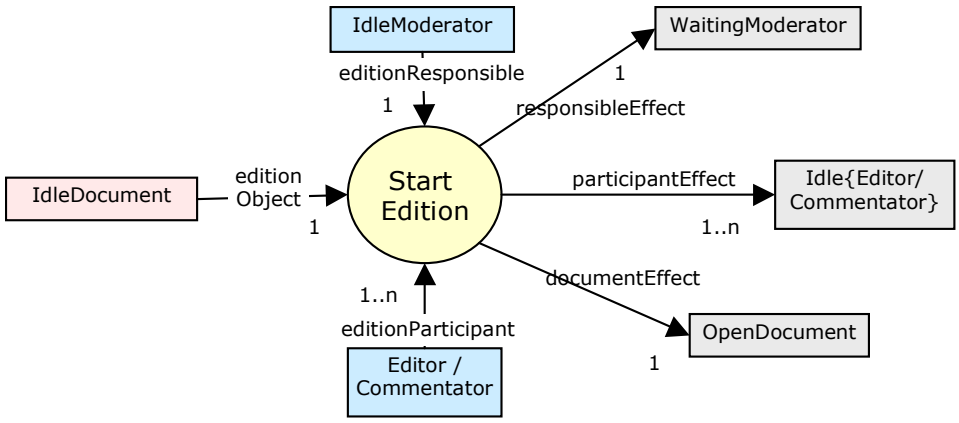


Figure 56. The Start Edition Process

A fragment of the serialisation of the Start Edition process is shown in Figure 57.

```
<daml:Class rdf:ID="StartEdition">
       <rdfs:subClassOf rdf:resource="&process;#AtomicProcess"/>
       <rdfs:subClassOf>
              <daml:Restriction daml:cardinality="1">
                     <daml:onProperty rdf:resource="#editionObject"/>
              </daml:Restriction>
       </rdfs:subClassOf>
       …
</daml:Class>
<rdf:Property rdf:ID="editionObject">
       <rdfs:subPropertyOf rdf:resource="&process;#input"/>
       <rdfs:domain rdf:resource="#StartEdition"/>
       <rdfs:range rdf:resource="#IdleDocument"/>
</rdf:Property>
…
```

Figure 57. Fragment of Start Edition serialisation with DAML-S

9.2.3.3 Functional model

User entities in collaborative editing functional model are viewer, commentator and editor. We have grouped commentator and editor for clarity, because they receive and perform the similar operation in the calls represented by the arrows [LLOR03a].

One of the service entities inside this model is the moderator. He is in charge of controlling the development of the service instance, taking into account the turns requested by commentators and editors (the users allowed to modify the document). The other service entity represented is the service instance, which encapsulates access to metadata and document entities.

Finally, the server entity corresponds to the collaborative editing server that is in charge of maintaining the information for the whole system. It also manages service instances creating and destroying them as moderator orders it.

Figure 58 shows graphically the collaborative editing functional model.

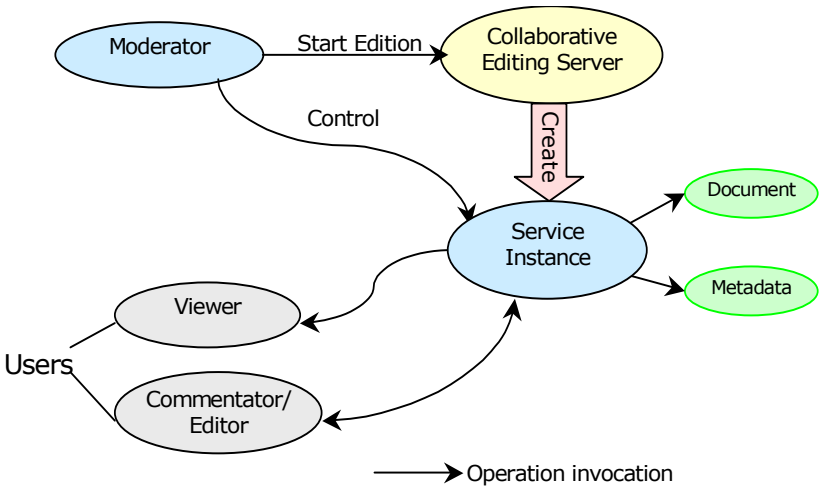


Figure 58. Collaborative editing functional model

9.2.3.4 Control and information flow

We have already mentioned throughout this section that moderator controls the flow of information in the service. He decides which participant will be the next to access to the document, based on the token requests made by the participants. The number of participants is unknown and also the sequence of steps that will take place in order to reach with the final objective of the service, the edition of a document thanks to the contribution of several participants.

Aspects related to the storage of the document information and workflow control information inside collaborative editing are specially addressed in the next sections.

9.2.3.4.1 DOCi storage

From the point of view of the storage of the documentation information, we can have the following possibilities:

- Centralised in a unique remote location. All users access remotely to the document.
- Local copies that are transmitted from user to user together with the token. The user with the access to the document makes modifications over it locally.

9.2.3.4.2 WFCi storage

In the collaborative editing service, only one entity, the moderator, has the control over the development of a specific instance of the service. In this way, the workflow control information can be locally stored at the moderator location.

Conclusions and future lines

10 Conclusions

Sections 2 and 3 of this thesis work, *Electronic commerce of services*, have been dedicated to illustrate the state of the art in several working areas related to this work. These areas are:

- Process description initiatives
- Metadata schemas initiatives

Inside process description initiatives, we have presented six different approaches for the definition of processes using XML-derived languages. The initiatives presented correspond to international organisations, international consortiums, private companies liaisons and other public or private initiatives in the field.

Moreover, three different metadata schemas initiatives have been presented with the common feature that they can be represented with XML-derived languages.

We studied several process description initiatives and metadata schemas initiatives using XML-derived languages with the objective of describing the structure of services offered by electronic means and its associated information. From the initiatives studied, we only used an initiative corresponding to process description, DAML-S/OWL, because we found that it was not possible to describe the metadata present inside electronic commerce of services using any of the schemas studied, as the information we wanted to represent was not described inside them.

In section 4 we have done an introduction to the main contributions of this research work. It has been described in a general way the contribution done.

The main contributions of this work are related with electronic commerce of services, e-services. In section 5 we have done a general introduction to electronic commerce, explaining the different kinds of electronic commerce we can have and also introducing the definition of e-service. In section 6, we have described the classification of e-services depending on the different features analysed.

After defining e-services classifications, we have described how workflow is present inside electronic commerce. This description can be found in section 7.

Based on the e-services classifications and workflow we have proposed a methodology for the development of systems offering services in an electronic way, following an electronic commerce scheme. This methodology has been discussed in section 8.

In sections 9.1 and 9.2, we have applied the methodology proposed to two completely different kinds of services: Legal and administrative services and Collaborative editing service. These services have been further explained in Appendix A and Appendix B, respectively.

The above contributions have been presented in different peer-reviewed international conferences and have been published in the corresponding proceedings books, all of them with ISBN from relevant publishers. This shows the relevance and applicability of the contributions presented. The five published papers group several of the above themes as we describe next.

Publication 1: Legal and administrative services through electronic commerce

A possible way for offering legal and administrative services in an electronic way was presented in the following publication, corresponding to an international conference:

7th International Conference on Intelligence in Services and Networks (IS&N 2000).

Llorente S., Delgado J.

Legal and administrative services through Electronic Commerce

February 2000, Athens (Greece)

Lecture Notes in Computer Science, Vol. 1774. Springer Verlag, 2000. ISBN 3-540-67152-8.

In this publication, the system implemented inside the TRials in the Domain of Electronic Commerce (TRADE, ACTS 328) European project was presented [TRAD04b]. This system allowed lawyers and administrative consultants to offer their services in an electronic way. In this publication, the business and architectural model of these services were presented, together with the flow of information and control needed for supplying them.

This conference was the 7th of a series of conferences devoted to the definition of services addressed to provide new solutions in the telecommunications sector. Intelligence in Services and Networks (IS&N) [ISN] was also the name given to a collection of projects within the European Commission collaborative research and development programme on Advanced Communications Technologies and Services (ACTS). The organisation of conferences was a key issue inside these projects as stated by the ACTS programme.

The relevance of this paper inside the conference was that it presented a novel approach for providing services by electronic means, electronic commerce of legal and administrative services.

Publication 2: A methodology for the development of workflow-based distributed systems

The fundamentals of the methodology described in section 8 were presented in the following publication, also corresponding to an international conference:

IEEE International Conference on Telecommunications (ICT 2001).

Llorente S., Delgado J., Polo J.

A methodology for the development of workflow-based distributed systems

June 2001, Bucharest (Romania)

IEEE ICT 2001 Proceedings, IEEE 2001. ISBN 973-99995-1-4.

The basic components of the methodology described in section 8 were presented in this publication, describing each of the initially considered components: Metadata associated to e-service and e-service workflow, definition of e-service workflow using the XML language, description of the control and information flow of an e-service and finally, the entities present in the functional model together with their operations.

This conference was organised by IEEE, the Institute of Electrical and Electronics Engineers, a leading authority in technical areas ranging from computer engineering, biomedical technology and telecommunications, to electric power, aerospace and consumer electronics, among others. ICT 2001 was the 8th of a series of conferences and it covered a variety of challenging telecommunication topics ranging from background fields like signals, traffic, coding, communication basics up to large communication systems and networks, fixed, mobile and

integrated. Applications, services, system and network management issues also received significant attention.

The paper presented was relevant to the applications and services areas covered by this conference, as it presented a different approach for the representation of services to be offered by distributed systems.

Publication 3: Using workflow-based systems for e-services provision

The methodology was further refined in the following publication, corresponding to an international conference, where new possibilities for the definition of e-services were studied and presented:

The Second IFIP Conference of E-Commerce, E-Business and E-Government (I3E 2002)

Llorente S., Delgado J.

Using workflow-based systems for e-services provision

October 2002, Lisbon (Portugal)

Towards the Knowledge Society. Kluwer Academic Publishers, 2002. ISBN 1-4020-7239-2.

In this publication, some of the concepts related to e-service classification were presented together with new alternatives for the definition of e-service workflow using DAML-S and refined functional model entities. We also presented the application of the methodology to the implementation of a system offering legal and administrative services in an electronic way inside the auspices of the Área 2000 project [AREA00a].

This conference was the 2nd of a series of conferences organised together by several working groups of IFIP, the International Federation for Information Processing (IFIP). IFIP is a non-governmental, non-profit umbrella organization for national societies working in the field of information processing. It was established in 1960 under the auspices of UNESCO as an aftermath of the first World Computer Congress held in Paris in 1959. Today, IFIP has several types of Members and maintains friendly connections to specialized agencies of the United Nations system and non-governmental organizations. Technical work, which is the heart of IFIP's activity, is managed by a series of Technical Committees which have several Working Groups. The I3E 2002 conference provided a forum for users, engineers, and scientists in academia, industry, and government to present their latest findings in e-commerce, e-business, or e-government applications and the underlying technology to support those applications.

The relevance of the paper presented focused on the possibility of describing e-commerce, e-business and e-government applications in a similar way, by using a workflow-based approach.

Publication 4: Dynamic e-services in collaborative applications

The next publication, corresponding again to an international conference, laid the foundations of the application of our methodology to a different kind of service, the collaborative editing service:

23rd International Conference on Distributed Computing Systems – Workshops (ICDCS 2003 Workshops)

Llorente S., Delgado J.

Dynamic e-services in collaborative applications

May 2003, Providence, Rhode Island, (United States)

23rd International Conference on Distributed Computing Systems – Workshops (ICDCS 2003 Workshops) Proceedings. IEEE, 2003. ISBN 0-7695-1921-0.

In this publication, the classification of e-services presented in section 6 was further refined, with the presentation of dynamic and static e-services. Moreover, the methodology was applied to collaborative editing service, describing each of their components for this service, as presented in section 9.2 and Appendix B.

ICDCS (International Conference on Distributed Computing Systems) is the premier conference in distributed computing systems and is sponsored by the IEEE Computer Society. ICDCS has a long history (this year they celebrate the 24th conference) of significant achievements and worldwide visibility. The conference provides a forum for engineers and scientists in academia, industry, and government to present and discuss their latest research findings. The coverage of topics is broad, thus bringing together researchers and developers with complementary backgrounds and areas of expertise.

The relevance of the paper presented was mainly focused on the presentation of a dynamic approach for providing collaborative services.

Publication 5: Use of metadata in a collaborative editing service

The last publication presented defines in more detail the collaborative editing service, applying it to the special case of edition of multimedia documents. This publication also corresponds to an international conference. The study of the edition of multimedia documents opens new horizons for the continuation of this work, in the field of document fragment expression. This is a problem currently not solved, as we will see in section 11, Future research lines. Publication details are:

7th International Conference on Electronic Publishing (ELPUB 2003)

Llorente S., Delgado J.

Use of metadata in a collaborative editing service

June 2003, Guimaraẽs (Portugal)

From Information to Knowledge (ELPUB2003), Univesidade do Minho, 2003. ISBN 972-98921-2-1

Apart from applying the methodology, several metadata description models were studied for this paper. They correspond to some of the ones presented in section 3 of the state of the art, and they represent one of the research lines we could follow after this work.

ELPUB2003 was the 7th in a series of annual international conferences on Electronic Publishing. The objective of ELPUB2003 is to bring together researchers, managers, developers, and users working on the issues related to electronic publishing for public, scientific and commercial applications.

The relevance of the paper presented in this conference comes from the application of an electronic publishing oriented approach to the collaborative editing service.

Final conclusion

As a final conclusion of this thesis work, we should mention that it is a contribution to the field of the electronic commerce of services, e-services, especially for the definition of a methodology for

their description. This description should help in the implementation of e-services systems, based on the different components of the methodology. The thesis results have been evaluated in several international and national projects as detailed. These results have also produced several publications in different peer-reviewed international conferences with very different characteristics, as it can be seen from their descriptions, which can give an idea of the applicability of the contributions presented in different research fields.

11 Future research lines

The methodology presented in this work is a first step to the full provision of services using an electronic commerce scheme. Elements coming from several external initiatives can be found in some of the components of our methodology. We have presented the ones we chose together with the justification of this decision. However, we have gone much forward to provide a consistent methodology that we have validated in real cases.

Nevertheless, as usual, we could not say that the work is finished, as we could continue in the use of the methodology for the definition of more different kinds of services, for instance, medical services or e-government applications, which have many common points with the legal and administrative services presented. We could even think in applying the methodology to e-learning services, where a combination of the approaches presented for legal services and collaborative editing service could be used. The definition and later implementation of these services is the first research line opened by this work.

Another research line, very related with the previous one is to complete the description of the self-learning mechanism for the definition of new e-services. This will be a key issue in the application of the methodology to different services, as the construction of the workflow definition is the most difficult task for the provision of electronic services.

We have also opened a different research line by applying metadata to the definition of services. Some of the existing metadata schemes could be used and mapped to the metadata used inside e-services. This will permit the construction of other kind of applications based on e-services, for instance, an advanced search of e-services based on metadata, using a similar scheme as the one described in [TOUS03a].

There is another field where research can be conducted from this work and it is the document fragment description issue. This is an old issue, as it was already outlined and described in the Open Document Architecture standard [ITUT95a], but it was a specific solution that has not been considered in XML-based documents. Nevertheless, this issue is also being discussed by the corresponding working groups of another set of standards, MPEG-7 and MPEG-21, describing what they call locators for accessing to different parts of their documents and specifications. The new research line in this area comes from the edition of multimedia documents described in section 9.2 and also in [LLOR03b]. The edition of multimedia documents by means of a collaborative editing service will allow the refinement of this service while, at the same time, could permit the connection with the description of document fragments. It is still an open issue, as the best option has to be considered, and contributions to the standards MPEG-7 and MPEG-21 could be done after this work.

Another research field is the application of the methodology presented in this work to the area of mobile applications. Currently, due to the bandwidth of the wireless networks and restrictions of the terminals, not many services are implemented. Based on the concepts described in this work, it would be possible to describe mobile services, adding the features specific to this field, if any. This will be particularly important in the multimedia arena, as new application scenarios are currently appearing, for instance, inside the MPEG-21 standard [MPEG02a].

Finally, since most of the e-services to develop will imply production and distribution of documents that could have associated digital rights, a natural field for continuation of work is Digital Rights Management (DRM). In fact, we have already started some work in this area, that can be summarised in the following three publications, presented to the corresponding international conferences:

Workshop on Regulatory ontologies and the modelling of complaint regulations (WORM 2003) (Part of the International Federated Conferences [OTM'03])

Delgado J., Gallego I., Llorente S., García R.

Regulatory Ontologies: An Intellectual Property Rights approach

November 2003, Catania, Sicily (Italy)

Lecture Notes in Computer Science, Vol. 2889. Springer Verlag, 2003. ISBN 3-540-20494-6.

The 16th Annual Conference on Legal Knowledge and Information Systems (JURIX-03)

Delgado J., Gallego I., Llorente S., García R.

IPROnto: An Ontology for Digital Rights Management

December 2003, Utrecht (The Netherlands)

Legal Knowledge and Information Systems. JURIX 2003: The Sixteenth Annual Conference. IOS Press, 2003. ISBN 1-58603-398-0.

4th International Conference on Web Delivering of Music (WEDELMUSIC 2004)

Rodríguez E., Llorente S., Delgado J.

Use of rights expression languages for protecting multimedia information

September 2004, Barcelona (Spain)

Conference proceedings to be published by IEEE Computer Society.

References

Bibliographic references

[ACTS04a] Advanced Communications Technologies & Services (ACTS) web site, www.cordis.lu/infowin/acts/home.html

[ALCA97a] Alcaraz E., Hughes B., Diccionario de términos jurídicos, April 1997, Ariel, ISBN 84-344-0509-1

[AREA00a] Proyecto Área 2000, dmag.upf.edu

[BPEL03a] Web Services Addressing (WS-Addressing), BEA, IBM and Microsoft, March 2003, www-106.ibm.com/developerworks/webservices/library/ws-add/

[BPEL03b] Business Process Execution Language for Web Services Version 1.1, May 2003, www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[CEN04a] European Committee for Standardization (CEN), 2004, www.cenorm.be/cenorm/index.htm

[CROSS01a] CrossFlow team, Final Report (Project Deliverable D16), January 2001, www.crossflow.org

[CROSS04a] CrossFlow project web site, 2004, www.crossflow.org

[CWA03a] CWA 14860, Dublin Core eGovernment Application Profiles, November 2003, ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/MMI-DC/cwa14860-00-2003-Nov.pdf

[CWA03b] CWA 14859, Guidance on the use of Metadata in eGovernment, November 2003, ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/MMI-DC/cwa14859-00-2003-Nov.pdf

[CWA03c] CWA 14855, Dublin Core Application Profile Guidelines, November 2003, ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/MMI-DC/cwa14855-00-2003-Nov.pdf

[CWA03d] CWA 14857, Mapping between Dublin Core and ISO 19115, *Geographic Information - Metadata*, November 2003, ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/MMI-DC/cwa14857-00-2003-Nov.pdf

[CWA03e] CWA 14856, Guidance material for mapping between Dublin Core and ISO in the Geographic Information Domain, November 2003, ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/MMI-DC/cwa14856-00-2003-Nov.pdf

[CWA03f] CWA 14858, Dublin Core Spatial Application Profile, November 2003, ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/MMI-DC/cwa14858-00-2003-Nov.pdf

[CWA04a] Workshop on Dublin Core Metadata, 2004, www.cenorm.be/cenorm/businessdomains/businessdomains/informationsocietystandardizationsystem/applying+technologies/meta-data+(dublin+core)++workshop/index.asp

[DAML01a] Joint US/EU ad hoc Agent Markup Language Committee, DAML+OIL, March 2001, www.daml.org/2001/03/daml+oil-index.html

[DAML03a] DAML-S Coalition, Describing Web Services using OWL-S and WSDL, October 2003, www.daml.org/services/owl-s/1.0/owl-s-wsdl.html

[DAML03b] OWL Services Coalition, OWL-S: Semantic Markup for Web Services, November 2003, www.daml.org/services/owl-s/1.0/owl-s.html

[DAML04a] DARPA Agent Markup Language, 2004, www.daml.org

[DAML04b] OWL-S 1.0 Release, 2004, www.daml.org/services/owl-s/1.0/

[DCMI03a] Guidelines for implementing Dublin Core in XML, April 2003, dublincore.org/documents/2003/04/02/dc-xml-guidelines/

[DCMI03b] Dublin Core Metadata Element Set, Version 1.1: Reference Description, June 2003, dublincore.org/documents/dces/

[DCMI03c] Using Dublin Core - Dublin Core Qualifiers, August 2003, dublincore.org/documents/usageguide/qualifiers.shtml

[DCMI03d] DCMI Metadata Terms, November 2003, dublincore.org/documents/dcmi-terms/

[DCMI04a] Dublin Core Metadata Initiative, 2004, dublincore.org

[EBXML01a] ebXML, ebXML Technical Architecture Specification (version 1.04), February 2001, www.ebxml.org/specs/ebTA.pdf

[EBXML01b] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), The Organization for the Advancement of Structured Information Standards (OASIS), ebXML Business Process Specification Schema (Version 1.01), May 2001, www.ebxml.org/specs/ebBPSS.pdf

[EBXML01c] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), The Organization for the Advancement of Structured Information Standards (OASIS), ebXML Core Components Dictionary (Version 1.04), May 2001, www.ebxml.org/specs/ccCTLG.pdf

[EBXML01d] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), The Organization for the Advancement of Structured Information Standards (OASIS), ebXML Naming Convention for Core Components (Version 1.04), May 2001, www.ebxml.org/specs/ebCCNAM.pdf

[EBXML01e] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), The Organization for the Advancement of Structured Information Standards (OASIS), ebXML Collaboration-Protocol Profile and Agreement Specification (Version 1.0), May 2001, www.ebxml.org/specs/ebCCP.pdf

[EBXML01f] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), The Organization for the Advancement of Structured Information Standards (OASIS), ebXML Business Process and Business Information Analysis Overview (Version 0.7), May 2001, www.ebxml.org/specs/bpOVER.pdf

[EBXML01g] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), The Organization for the Advancement of Structured Information Standards (OASIS), ebXML Business Process Analysis Worksheets & Guidelines (Version 0.10), May 2001, www.ebxml.org/specs/bpWS.pdf

[EBXML01h] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), The Organization for the Advancement of Structured Information Standards (OASIS), ebXML E-Commerce Patterns, (Version 0.99), May 2001, www.ebxml.org/specs/bpPATT.pdf

[EBXML01i] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), The Organization for the Advancement of Structured Information Standards (OASIS), ebXML Catalog of Common Business Processes (Version 0.99), May 2001, www.ebxml.org/specs/bpPROC.pdf

[EBXML02a] The Organization for the Advancement of Structured Information Standards (OASIS), ebXML Message Service Specification (Version 0.99), April 2002, www.ebxml.org/specs/ebMS2.pdf

[EBXML03a] United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), UN/CEFACT Modeling Methodology (UMM) User Guide, 2003, www.iprismglobal.com/prism/ufiles/tsimon/umm.pdf

[EBXML04a] OASIS web site, 2004, www.oasis-open.org/home/index.php

[GALL01a] Gallego I., Modelos para comercio electrónico basados en Sistemas Intermediarios, PhD. Thesis work, May 2001

[HTTP99a] Internet Engineering Task Force (IETF), Hypertext Transfer Protocol – HTTP/1.1, June 1999, www.ietf.org/rfc/rfc2616.txt

[ISO96a] JTC 1/SC 22, ISO/IEC 11404:1996 Information technology - Programming languages, their environments and system software interfaces - Language-independent datatypes, 1996, www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=19346&ICS1=35

[ISO02a] ISO Standard 639.2, Codes for the Representation of Names of Languages Part 2: Alpha-3 Code, March 2002, www.loc.gov/standards/iso639-2/langhome.html

[ISO03a] ISO Standard 15836-2003, February 2003, www.niso.org/international/SC4/n515.pdf

[ISN] Intelligence in Services and Networks, www.cordis.lu/infowin/acts/analysys/concertation/isn/

[ITUT95a] ITU-T Study Group 8, ISO/IEC Joint Technical Committee 1, ITU-T Recommendation T.422, ISO/IEC 8613-12 Information Technology – Open Document Architecture (ODA) and Interchange Format – Identification of fragments, 1995

[LLOR00a] Llorente S., Delgado J., Legal and administrative services through electronic commerce, 7th International Conference in Intelligence in Services and Networks, ISN&N 2000, February 2000, Springer, ISBN 3-540-00021-6

[LLOR01a] Llorente S., Delgado J., Polo J., A methodology for the development of workflow-based distributed systems, IEEE International Conference on Telecommunications, ICT 2001, June 2001, IEEE ICT 2001 Proceedings, ISBN 973-99995-1-4

[LLOR02a] Llorente S., Delgado J., Using workflow-based systems for e-service provision, 2nd IFIP Conference on e-commerce, e-business and e-government, October 2002, Kluwer Academic Publishers, ISBN 1-4020-7239-2

[LLOR03a] Llorente S., Delgado J., Dynamic e-services for collaborative applications, 23rd International Conference on Distributed Computing Systems – Workshops (ICDCS 2003 Workshops), May 2003, IEEE, ISBN 0-7695-1921-0

[LLOR03b] Llorente S., Delgado J., Use of metadata in a collaborative editing service, Electronic Publishing 2003, June 2003, From Information to knowledge, Electronic Publishing 2003 (ElPub2003) Proceedings, ISBN 972-98921-2-1

[LOM] 1484.12.2 Standard for ISO/IEC 11404 binding for Learning Object Metadata data model

[LOM02a] 1484.12.1-2002 IEEE Standard for Learning Object Metadata, June 2002, ISBN 0-7381-3297-7

[LOM03a] IEEE 1484.12.3/D1 Draft Standard for eXtensible Markup Language (XML) Binding for Learning Object Metadata Data Model, February 2003, www.cs.kuleuven.ac.be/~erikd/LOM/20030214/ballot.pdf

[LOM04a] WG12: Learning Object Metadata, 2004, ltsc.ieee.org/wg12/index.html

[LOM04b] Learning Object Metadata XML Schema, April 2004, ltsc.ieee.org/xsd/lomv1.0/lom.xsd

[MIME96a] Internet Engineering Task Force (IETF), Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, November 1996, www.ietf.org/rfc/rfc2045.txt

[MPEG02a] ISO/IEC JTC1/SC29/WG11, MPEG-21 overview, October 2002, www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm

[NEWT03a] Proyecto Área 2000: NewTrade, 2003, dmag.upf.edu

[NISO01a] NISO Standard Z39.85-2001, September 2001, www.niso.org/standards/resources/Z39-85.pdf

[OWLW04a] W3C, OWL Web Ontology Language, February 2004. www.w3.org/TR/owl-ref/

[RDF04a] W3C Resource Description Framework (RDF), www.w3.org/RDF/

[RFC01a] Alvestrand H., Tags for the Identification of Languages, January 2001, www.faqs.org/rfcs/rfc3066.html

[RFC02a] Rosenberg J. et al., SIP: Session initiation protocol, June 2002, www.faqs.org/rfcs/rfc3261.html

[SOAP02a] W3c Note, SOAP Version 1.2 Email Binding, June 2002, www.w3.org/TR/2002/NOTE-soap12-email-20020626

[SOAP03a] W3c Recommendation, SOAP Version 1.2 Part 0: Primer, June 2003, www.w3.org/TR/soap12-part0/

[SOAP03b] W3c Recommendation, SOAP Version 1.2 Part 1: Messaging Framework, June 2003, www.w3.org/TR/soap12-part1/

[SOAP03c] W3c Recommendation, SOAP Version 1.2 Part 2: Adjuncts, June 2003, www.w3.org/TR/soap12-part2/

[TOUS03a] Tous R., Delgado J., Interoperability adaptors for distributed information search on the web, Electronic Publishing 2003, June 2003, From Information to knowledge, Electronic Publishing 2003 (ElPub2003) Proceedings, ISBN 972-98921-2-1

[TRAD99a] TRials in the Domain of Electronic Commerce (TRADE) demo web site, 1999, dmag.upf.edu/legal/TRADE/PaginasDemo/Eng/DemoMainPage.html

[TRAD04a] TRials in the Domain of Electronic Commerce (TRADE) information page, dmag.upf.edu

[TRAD04b] ACTS 328, TRials in the Domain of Electronic Commerce (TRADE) information page, www.cordis.lu/infowin/acts/rus/projects/ac328.htm

[UDDI00a] UDDI Overview presentation, June 2000, www.uddi.org/pubs/UDDI_Overview_Presentation.ppt

[UDDI00b] uddi.org, UDDI Technical White Paper, Setember 2000, www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[UDDI01a] uddi.org, UDDI Executive White Paper, November 2001, www.uddi.org/pubs/UDDI_Executive_White_Paper.pdf

[UDDI02a] uddi.org, UDDI Version 3.0 features list, July 2002, www.uddi.org/pubs/uddi_v3_features.htm

[UDDI02b] uddi.org, UDDI Version 3.0, July 2002, uddi.org/pubs/uddi-v3.00-published-20020719.htm

[UDDI04a] uddi.org, OASIS member section, 2004, www.uddi.org

[UNCE04a] UN/CEFACT United Nations Centre for Trade Facilitation and Electronic Business, 2004, www.unece.org/cefact/

[VCAR01a] Ianella R., W3C Note Representing vCard Objects in RDF/XML, February 2001, www.w3.org/TR/vcard-rdf

[W3C04a] World Wide Web Consortium (W3C), 2004, www.w3.org

[WEBO04a] Web-Ontology (WebOnt) Working Group, 2004. www.w3.org/2001/sw/WebOnt/

[WFMC98a] Conformance White Paper, June 1998, www.wfmc.org/standards/docs/conformance.pdf

[WFMC98b] Workflow Management Application Programming Interface (Interface 2&3) Specification, July 1998, www.wfmc.org/standards/docs/if2v20.pdf

[WFMC98c] Workflow Management Facility, July 1998, www.wfmc.org/standards/docs/jointflow.pdf

[WFMC98d] Workflow Management Coalition Audit Data Specification, September 1998, www.wfmc.org/standards/docs/TC-1015_v11_1998.pdf

[WFMC99a] Workflow Management Coalition - Terminology & Glossary, February 1999, www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

[WFMC99b] Workflow Management Coalition Interface 1: Process Definition Interchange Process Model, October 1999, www.wfmc.org/standards/docs/TC-1016-P_v11_IF1_Process_definition_Interchange.pdf

[WFMC99c] Workflow Management Coalition Workflow Standard - Interoperability Abstract Specification, November 1999, www.wfmc.org/standards/docs/TC-1012_Nov_99.pdf

[WFMC00a] Workflow Management Coalition Workflow Standard - Interoperability Internet e-mail MIME Binding, January 2000, www.wfmc.org/standards/docs/tc018v12.pdf

[WFMC00b] Workflow Standard - Interoperability XML-HTTP binding, February 2000, www.wfmc.org/standards/docs/http_binding_proposal.pdf

[WFMC02a] AWSP, Asynchronous Web Services Protocol, February 2002, www.wfmc.org/standards/docs/AWSP_20020213.zip

[WFMC02b] Workflow Management Coalition Workflow Standard, Workflow Process Definition Interface - XML Process Definition Language, October 2002, www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf

[WFMC02c] XML Schema for XML Process Definition Language (XPDL), www.wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd

[WFMC03a] Wf-XML 2.0, XML Based Protocol for Run-Time Integration of Process Engines, October 2003, www.wfmc.org/standards/docs/ASAP_WfXML_2003_10.zip

[WFMC04a] Workflow Management Coalition's (*WfMC*) web site, www.wfmc.org

[WSDL01a] W3c Note, March 2001, www.w3.org/TR/wsdl

[WSDL03a] Web Services Description Language (WSDL) Version 1.2 Part 3: Bindings, June 2003, www.w3.org/TR/wsdl12-bindings/

[WSDL03b] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, November 2003, www.w3.org/TR/wsdl20/

[WSDL03c] Web Services Description Language (WSDL) Version 2.0 Part 2: Message Patterns, November 2003, www.w3.org/TR/wsdl20-patterns/

[XML04a] W3c Recommendation, Extensible Markup Language (XML) 1.0 (Third Edition), February 2004, www.w3.org/TR/2004/REC-xml-20040204/

[XMLS01a] W3c Recommendation, XML Schema 1.0, May 2001, www.w3.org/XML/Schema

[XPAT99a] W3c Recommendation, XML Path Language (XPath) Version 1.0, November 1999, www.w3.org/TR/xpath

Thesis research work publications

LLORENTE, Silvia; DELGADO, Jaime. "Legal and administrative services through electronic commerce", 7th International Conference on Intelligence in Services and Networks, IS&N'2000, Athens (Greece), February 2000. Telecommunications and IT Convergence Towards Service E-volution, Lecture Notes in Computer Science, Vol. 1774, Springer, ISBN 3-540-67152-8

LLORENTE, Silvia; DELGADO, Jaime; POLO, José. "A methodology for the development of workflow based distributed systems", 8th IEEE International Conference on Telecommunications, IEEE ICT 2001, Bucharest (Romania), June 2001. IEEE ICT 2001 Proceedings, Vol. 2, GEOMA, ISBN 973-99995-1-4

LLORENTE, Silvia; DELGADO Jaime. "Using workflow-based systems for e-services provision", The Second IFIP Conference on E-Commerce, E-Business, E-Government, I3E 2002, Lisbon (Portugal), October 2002. Towards the knowledge society - eCommerce, eBusiness and eGovernment, Kluwer Academic Publishers, ISBN 1-4020-7239-2

LLORENTE, Silvia; DELGADO, Jaime. "Dynamic e-Services for Collaborative Applications", 3rd International Workshop on Distributed Auto-adaptive and Reconfigurable Systems, DARES, in conjunction with 23rd International Conference on Distributed Computing Systems, ICDCS 2003, Providence, Rhode Island (United States), May 2003. 23rd International Conference on Distributed Computing Systems – Workshops (ICDCS 2003 Workshops), IEEE, ISBN 0-7695-1921-0

LLORENTE, Silvia; DELGADO, Jaime. "Use of metadata in a collaborative editing service", ICCC/IFIP 7_{th} International Conference on Electronic Publishing, Guimarães (Portugal), June 2003. Publisher, ISBN 972-98921-2-1

Appendix

Appendix A. Legal e-service

11.1 Introduction

In this appendix it is described in detail the structure of a legal e-service, divorce, presented in section 9.1, Legal and administrative e-services. The structure includes the steps of a divorce, together with their relationships, which represent the divorce general workflow. For each step, it is also described its representation as a DAML-S process, where inputs, outputs, preconditions and effects representing the participants, documents interchanged and conditions to be accomplished are shown.

11.2 Divorce general workflow diagram

Figure 59 shows the general diagram for a divorce according to Spanish law. In this diagram it is not considered the contracting part of the legal service, corresponding to preliminary phase presented in section 9.1.4, Workflow in legal services. The contracting or preliminary phase occurs before the steps shown in this diagram.

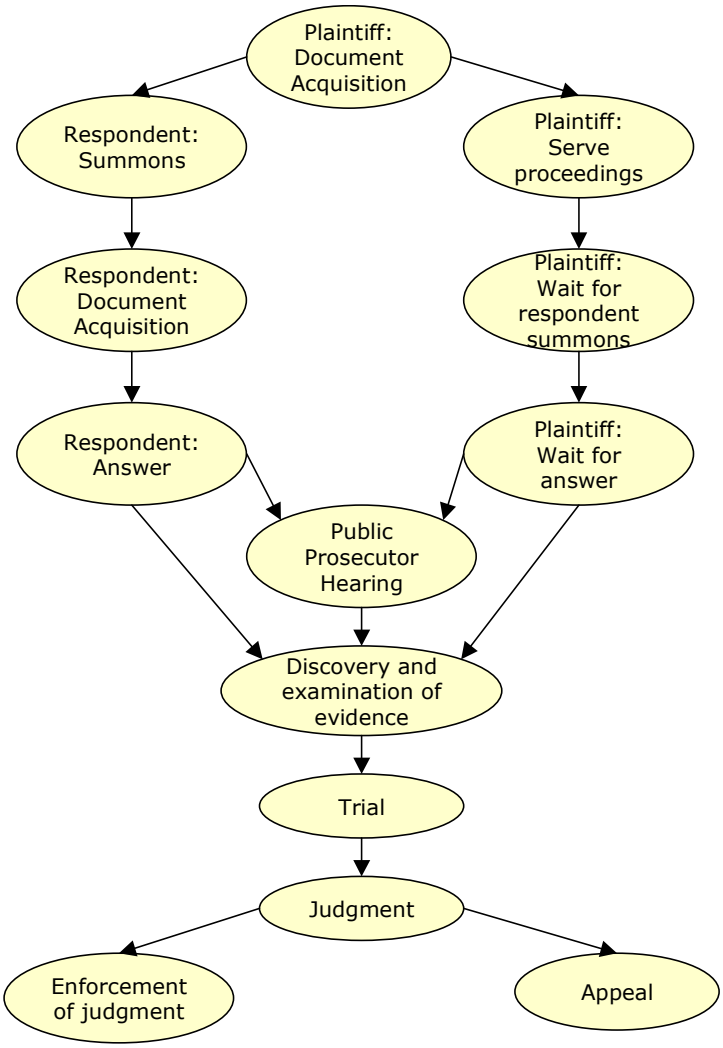


Figure 59. Divorce general workflow

The diagram shown in Figure 59 is organised in the actions or steps in which divorce service is decomposed. Arrows represent the relationship among actions, but two consecutive actions can be active at the same time [ALCA97a].

The participants in the divorce legal service are: Client, Lawyer, Notary, Procurator and Court. These are the base participants, but we could add more if it is needed. There is a part of the divorce workflow that is separated into plaintiff and defendant. The reason for this is that a divorce is different from the point of view of each of the two parties. In practice, for one divorce, we have two different e-services, one for the plaintiff part and the other for the respondent part. For each of them we have its corresponding client, lawyer or lawyers, notary and procurator. The only common participant on both e-services should be court (unless the two divorcing parties contract the same lawyer or go to the same notary), which is not very common.

The plaintiff party has the initiative, starting the legal service with the *Document Acquisition* action. On it, the plaintiff client sends all needed documentation to his lawyer. After this, plaintiff party *serves legal proceedings*, and the respondent receives *summons*. This is really the point where the respondent case starts, as he or she receives notification that the plaintiff party has presented a lawsuit against him or her.

The *Respondent: Summons* action does not really occur on-line, but we have included it for completeness. The first on-line action for the respondent party is *Respondent: Document Acquisition*. This action is equivalent to *Plaintiff: Document Acquisition*, and the respondent client sends all needed documentation to the lawyer he or she has contracted. After *Document Acquisition*, the respondent party sends to Court the *Answer* to the lawsuit presented by plaintiff party.

During the respondent actions, *Summons, Document Acquisition* and *Answer*, the plaintiff part is in a waiting state. First, he or she has to wait for respondent summons. After this, he or she waits for respondent answer. In these two actions, the plaintiff receives the respondent party documents through Court.

The next action, *Public Prosecutor Hearing*, is the first one common to plaintiff and respondent parties. This is an optional action, only needed if plaintiff and respondent parties have children in common. When there are no children, the first common action is *Discovery and examination of evidence*. In this action, both respondent and plaintiff parties present evidences suitable for their claims. Court decides which ones are accepted and carried out.

Next action in the divorce legal service is the *Trial*. In this action, both parties go to Court to explain their respective points of view. This is the first "physical" contact among plaintiff and respondent parties and Court. Until now, all contact is done by means of documents and information. When Court decides that *Trial* is finished, the judge is ready to deliver *Judgment*, the next action in the divorce service. The *Judgment* action only has one document, the *Judgment*, which is sent from Court to Client by means of Procurator and Lawyer.

Finally, two things may happen. The first one is that one or both parties disagree in the *Judgment* and they *Appeal* it, starting a new legal case in front of a different Court. The second thing that may happen is that any of the parties agrees with the *Judgment*. In this case, if the other party does not accomplish with what the *Judgment* says, an *Enforcement of Judgment* is done to force the other party to accomplish the *Judgment*.

11.3 Representation of the divorce legal service processes using DAML-S

In this section, the different processes inside a divorce are represented by means of DAML-S IOPEs. Some of them are decomposed into several subprocesses, as they represent a complex interchange of documents among several parties

11.3.1 Plaintiff: Document Acquisition

Figure 60 shows the representation of *Plaintiff: Document Acquisition* step from a divorce using DAML-S IOPEs (Input/Output/Precondition/Effect). This step represents the sending of information from the client to his lawyer(s) when a divorce process has been started.

Figure 61 shows the sending of a special document, *Official Authorisation*. This document has to be done by a notary and gives a lawyer or lawyers the **legal ability** for acting in name of the client. It is separated into two subprocesses, as we can have two possibilities: One is that the notary gives the official authorisation to client and, afterwards, client gives it to lawyer. The other one is that the notary directly gives the official authorisation to the lawyer or lawyers.

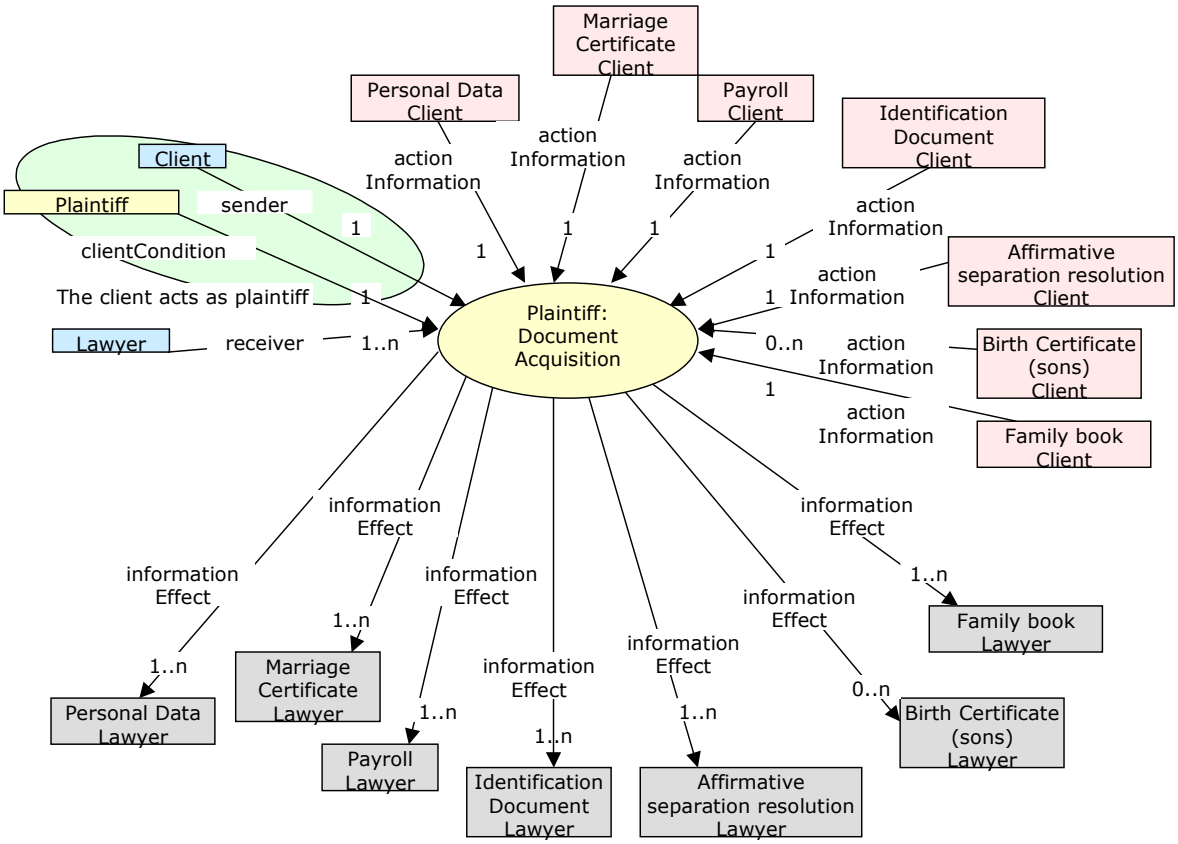


Figure 60. Document Acquisition

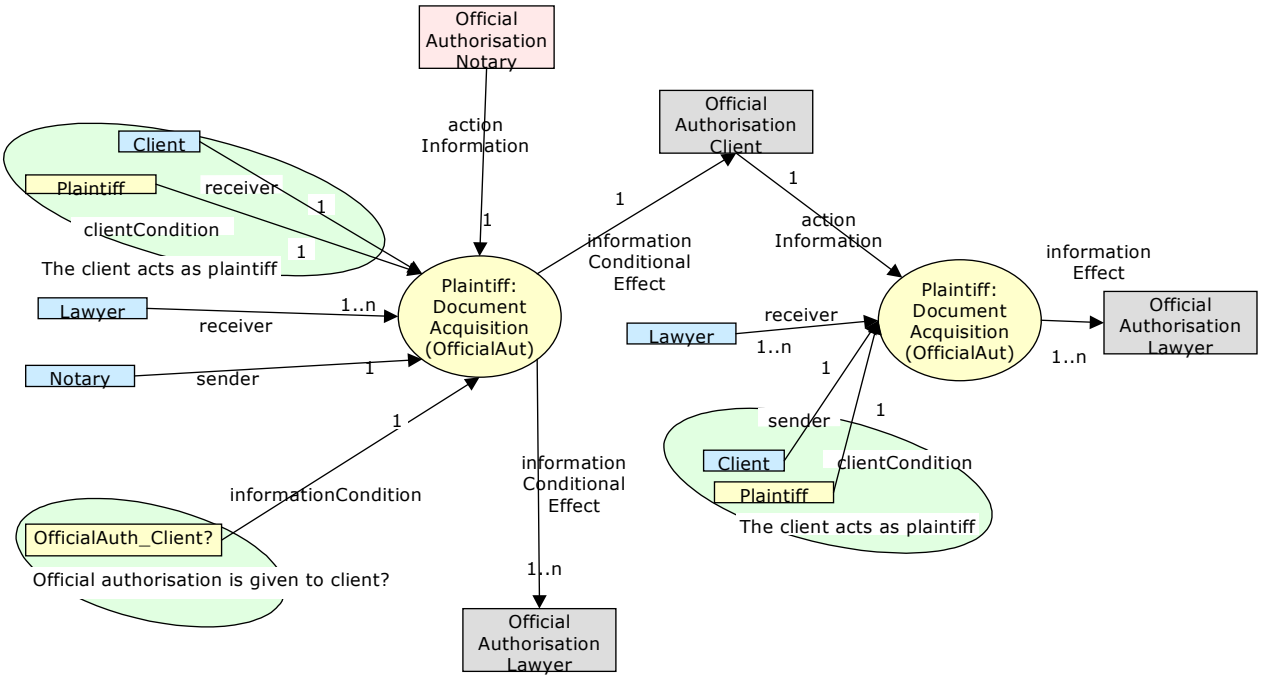


Figure 61. Document Acquisition of the Official Authorisation

11.3.2 Plaintiff: Serve proceedings

Figure 62 and Figure 63 show the representation of *Plaintiff: Serve proceedings* step from a divorce. This step represents the start of the legal proceedings taking the appropriate information to court. It is divided into two subprocesses because the lawyer has to give all information to procurator. Afterwards, procurator gives this information to court. The most relevant document in this action is the lawsuit, that lawyer prepares based on the documentation presented by his or her client.

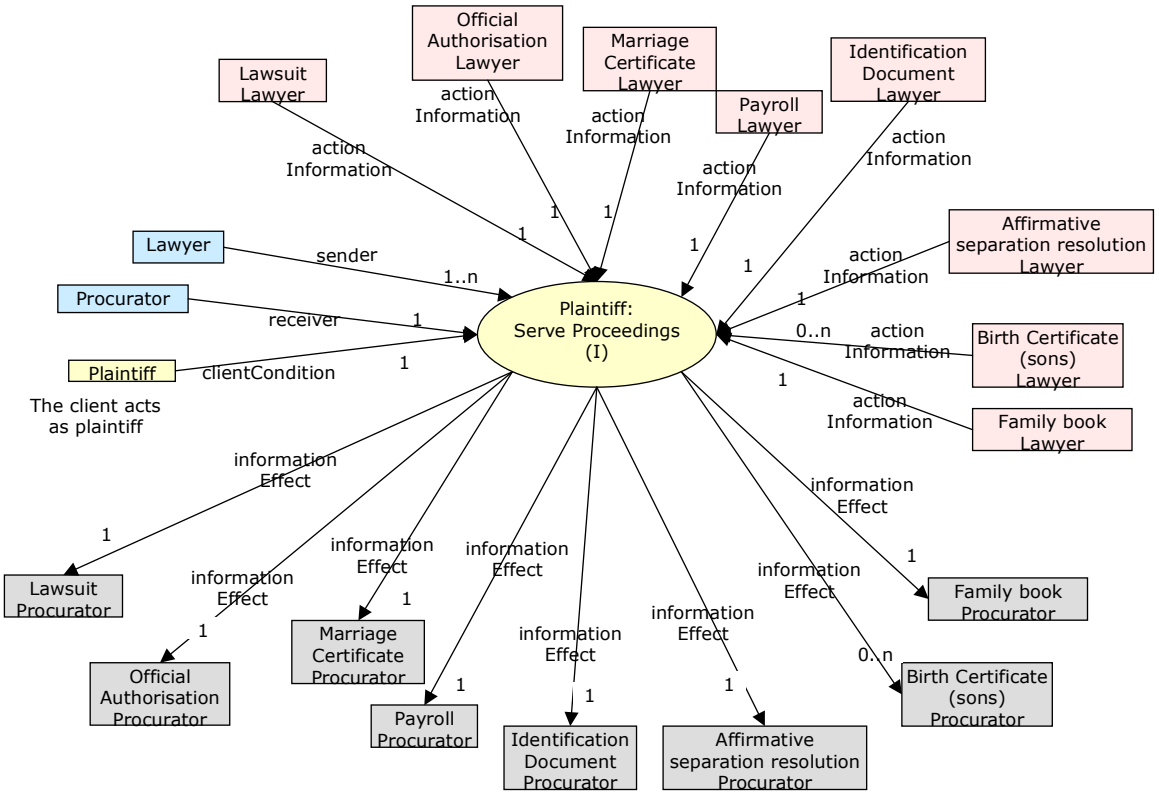


Figure 62. Serve proceedings (first part)

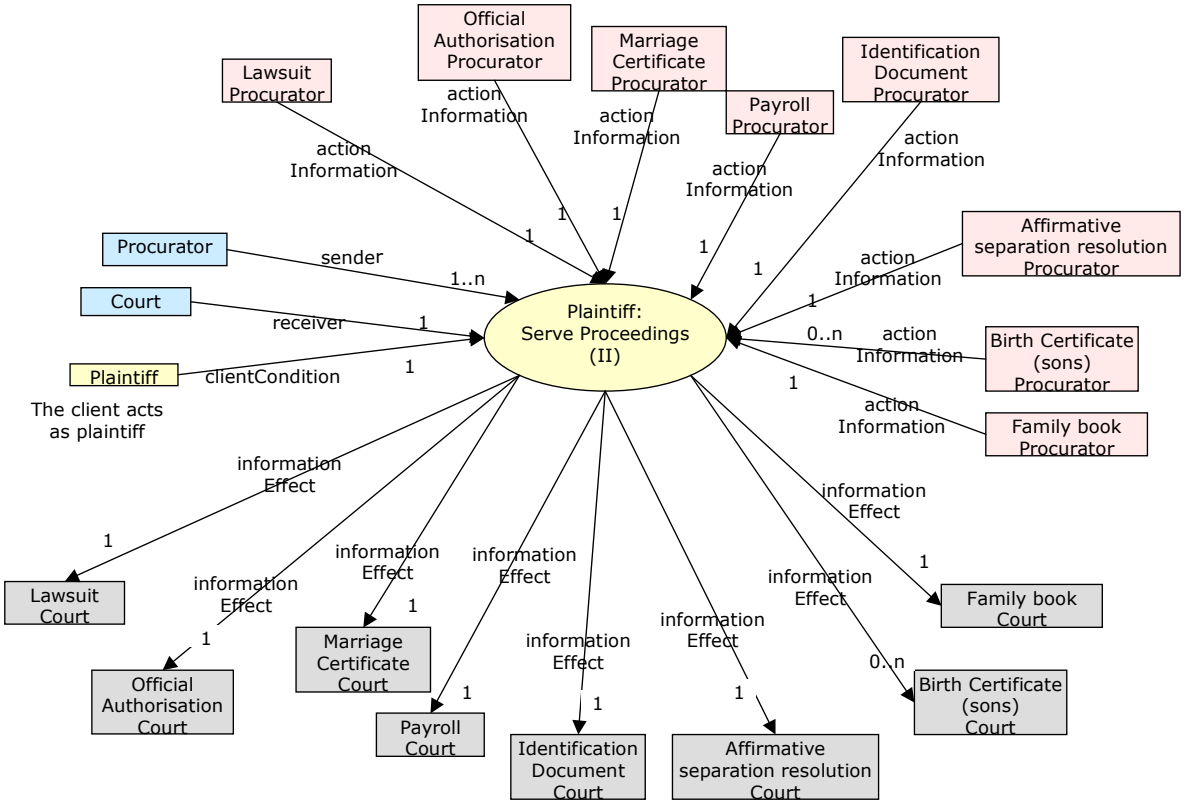


Figure 63. Serve proceedings (second part)

11.3.3 Respondent: Summons

Figure 64 and Figure 65 show the representation of *Respondent: Summons* step from a divorce. This step represents the start of the divorce from the respondent part point of view. The first part of this step shown in Figure 64 does not take place in the application offering legal and administrative services, as it is done directly from court to client and the client does not have a lawyer for the moment. In this first part, the documents that the respondent party receives are the ones sent by the plaintiff party to court in order to start a lawsuit against the respondent party. The second part of this step, the one shown in Figure 65, is where the respondent contracts a lawyer that provides the service and therefore the service can be started. The documents that respondent party sends to lawyer are the ones he or she has received from the plaintiff party.

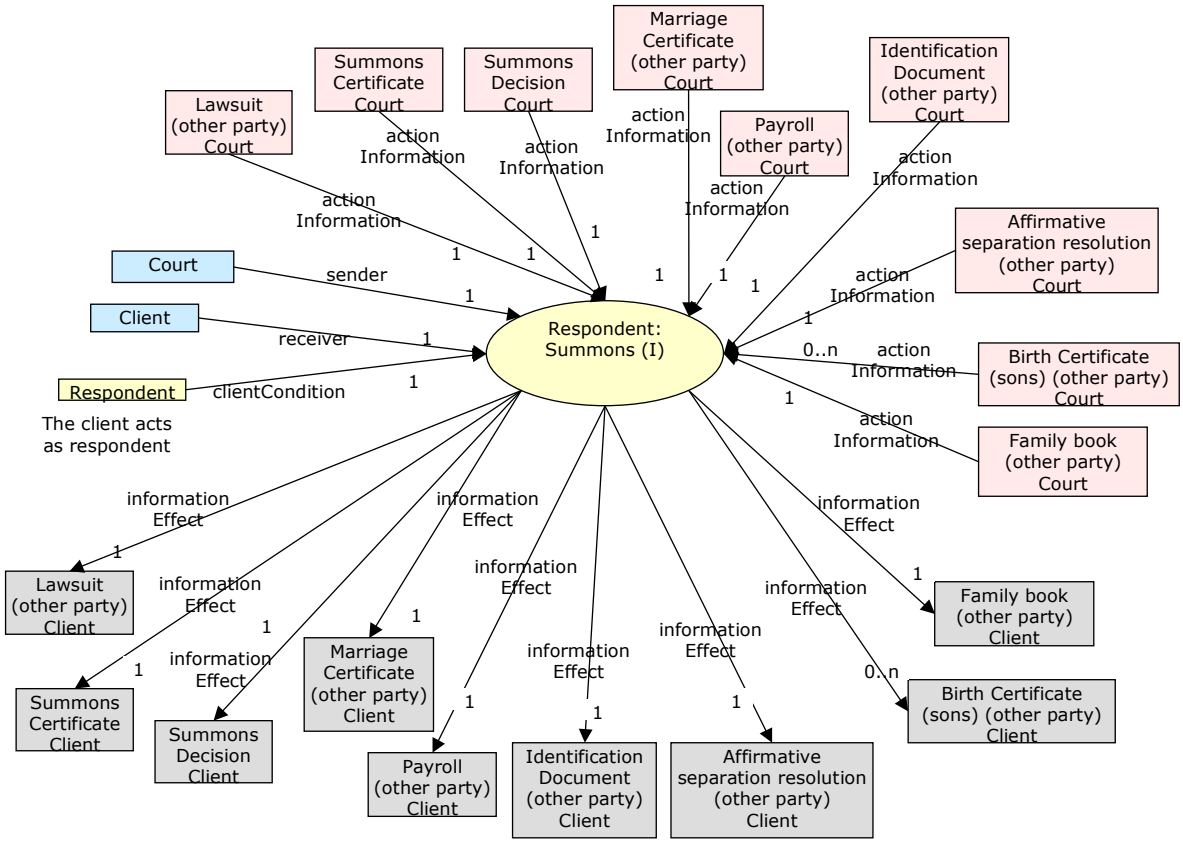


Figure 64. Summons (first part)

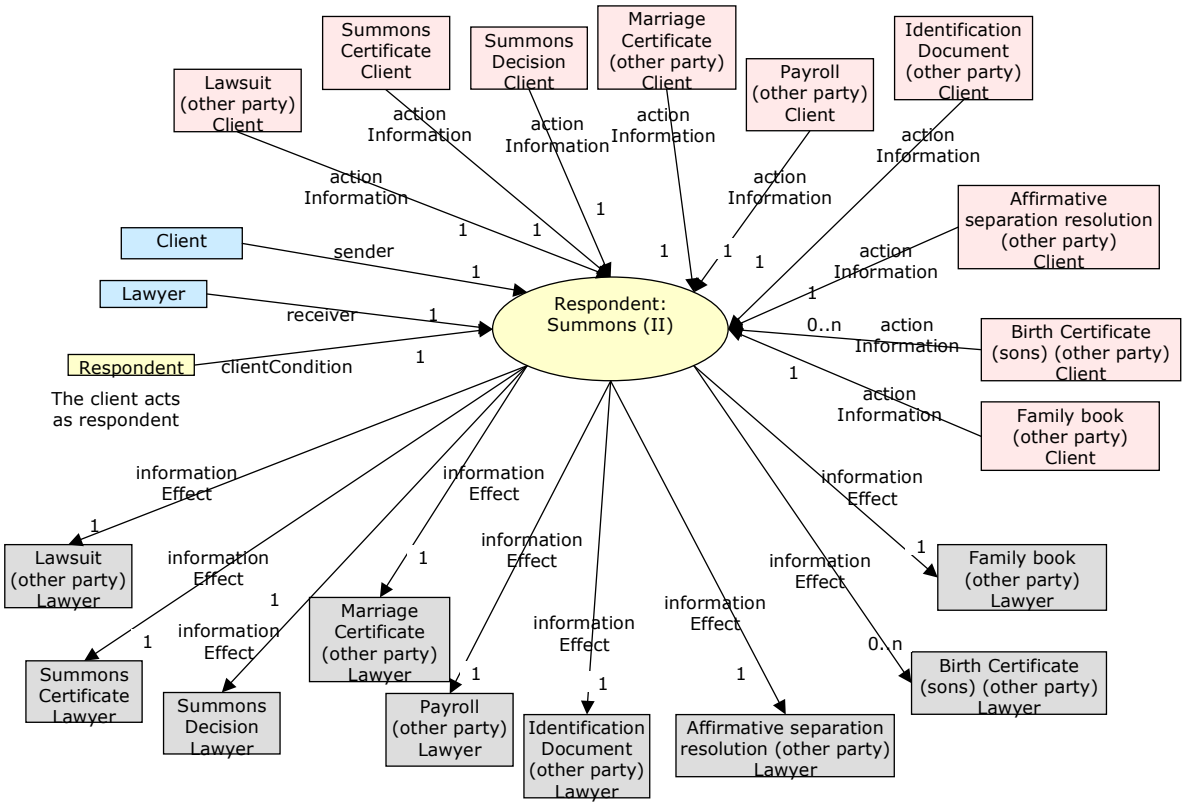


Figure 65. Summons (second part)

11.3.4 Respondent: Document Acquisition

Figure 66 shows the representation of *Respondent: Document Acquisition* step from a divorce. This step represents the sending of information from the client to his lawyer(s) when he or she has received a divorce lawsuit.

Figure 67 shows the sending of a special document, *Official Authorisation*. This document has to be done by a notary and gives a lawyer or lawyers the **legal ability** for acting in name of the client. It is separated into two subprocesses, as we can have two possibilities: One is that the notary gives the official authorisation to client and, afterwards, client gives it to lawyer. The other one is that the notary directly gives the official authorisation to the lawyer or lawyers.

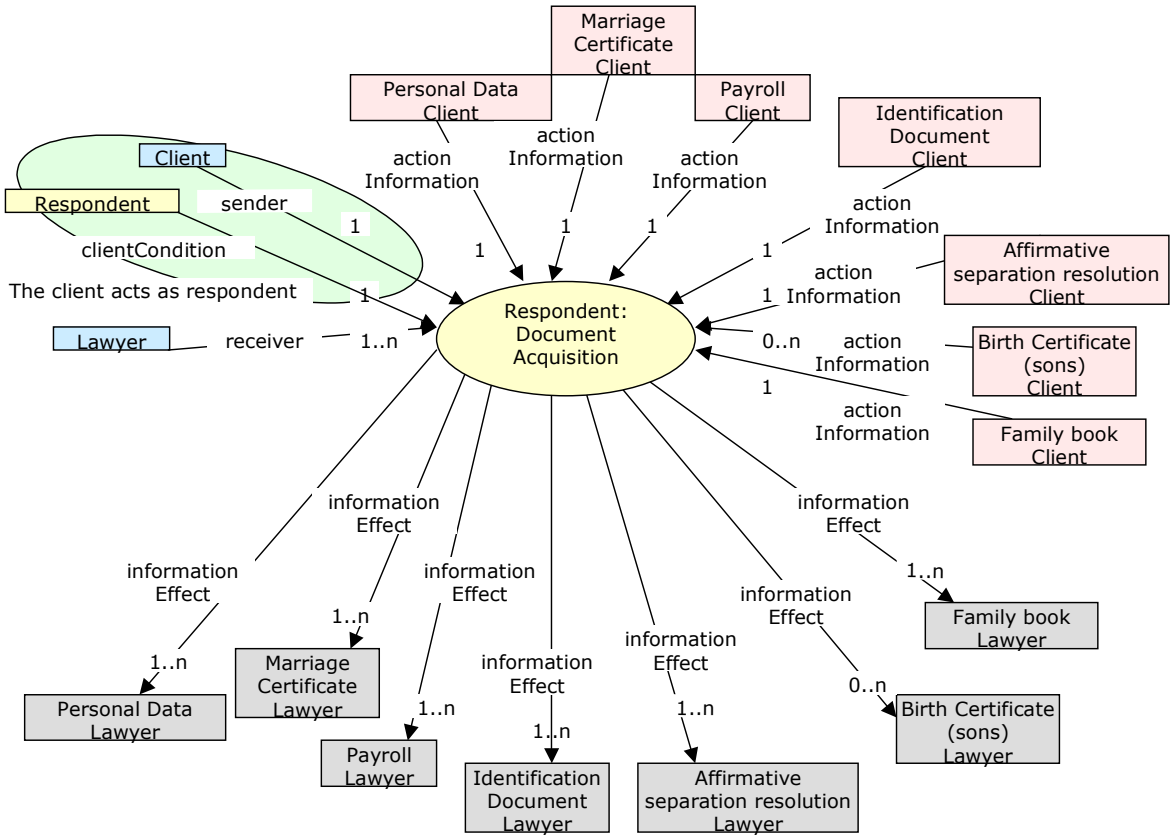


Figure 66. Respondent Document Acquisition

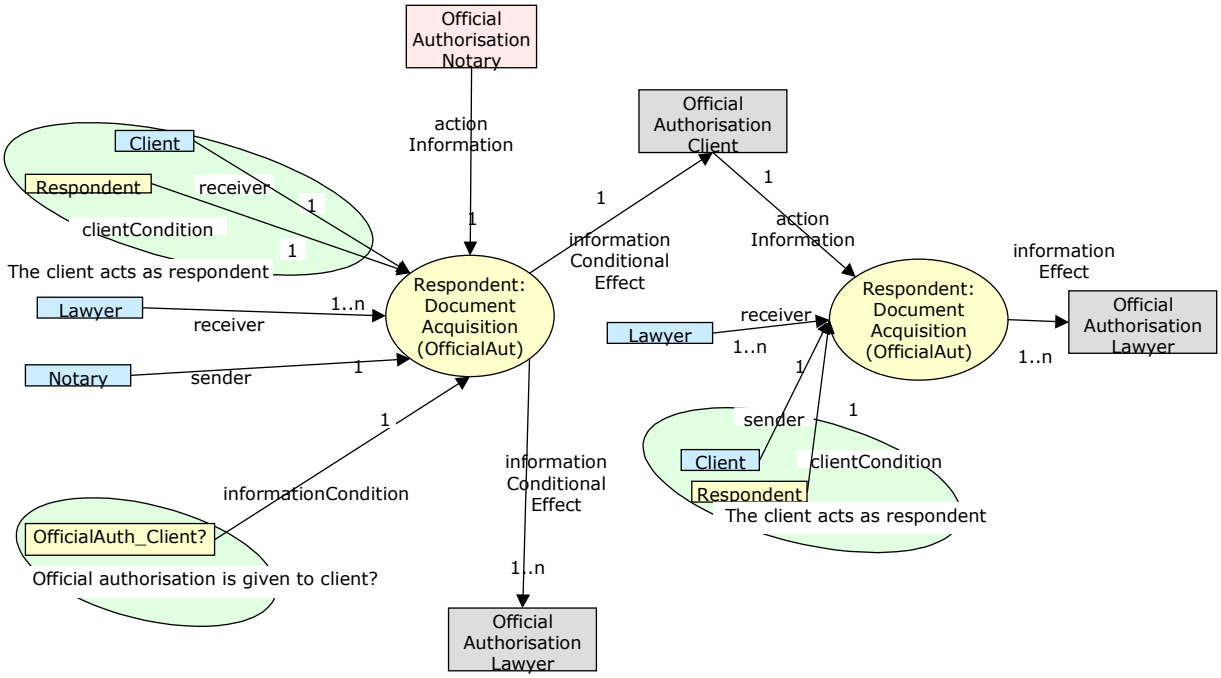


Figure 67. Respondent Official Authorisation Document Acquisition

11.3.5 Plaintiff: Wait for respondent summons

Figure 68 shows the representation of *Plaintiff: Wait for respondent summons* step from a divorce. In this step, the plaintiff client waits for the respondent client to be informed of the divorce lawsuit presented against him or her. It is separated into two subprocesses because the documents come from court to procurator and then procurator has to give the Summons decision document to the lawyer.

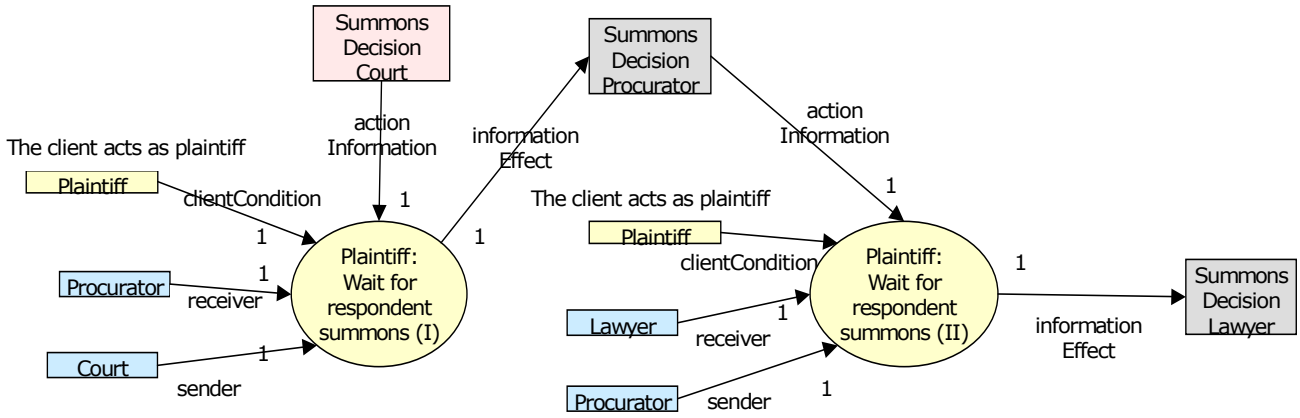


Figure 68. Wait for respondent summons

11.3.6 Respondent: Answer

Figure 69 shows the representation of *Respondent: Answer* step from a divorce. In this step, the respondent client presents the answer to the divorce lawsuit presented against him or her. It is separated because the documents go from lawyer to procurator and then procurator has to give the documents to court.

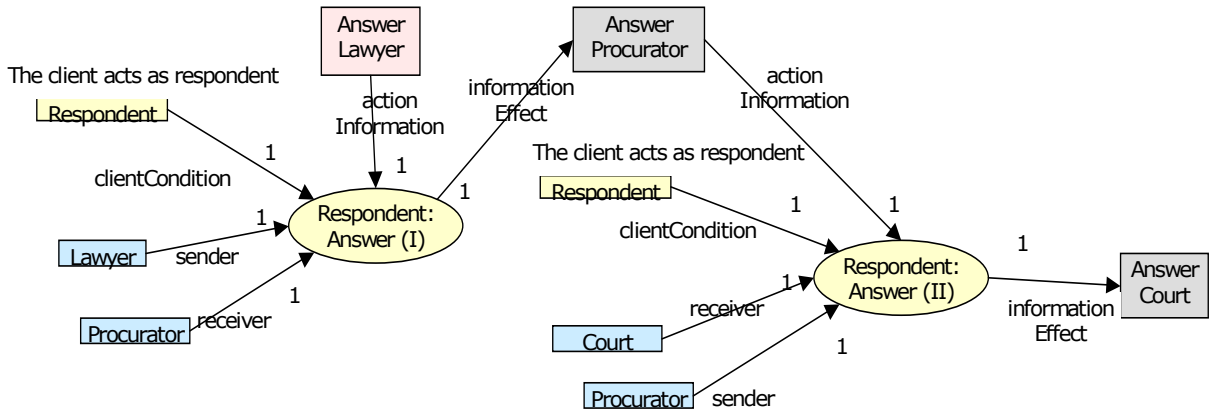


Figure 69. Answer

11.3.7 Plaintiff: Wait for answer

Figure 70 shows the representation of *Plaintiff: Wait for answer* step from a divorce. In this step, the plaintiff client waits for the answer of the respondent client to the lawsuit presented against him

or her. It is separated because the documents come from court to procurator and then procurator has to give the documents to the lawyer.

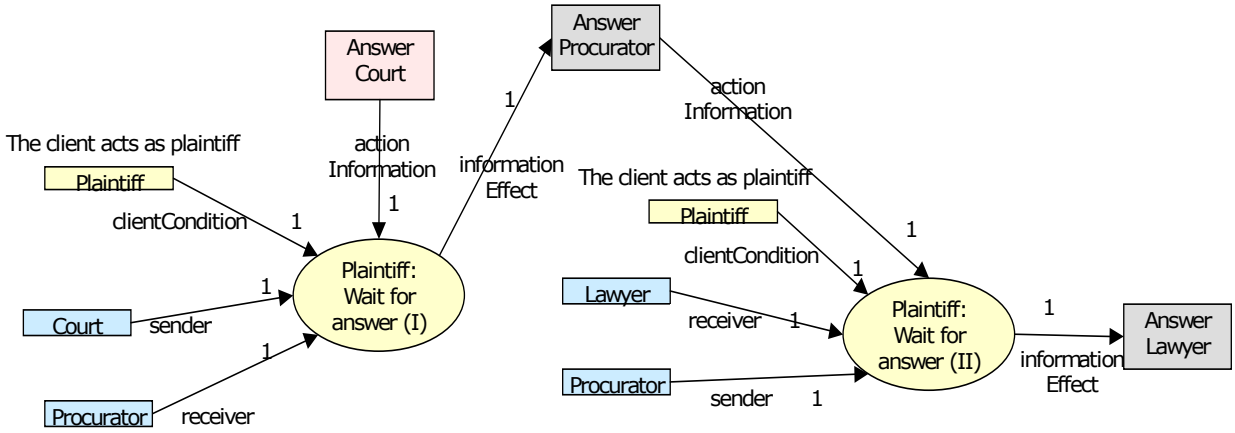


Figure 70. Wait for answer

11.3.8 Public prosecutor hearing

This is the first common action for plaintiff and defendant. It is optional, and the divorce could jump directly to the next action, *Discovery and examination of evidence*, from the previous actions, *Respondent: Answer* and *Plaintiff: Wait for answer*. In case it takes place, many steps have to be taken, in the form of document interchanges. These interchanges among court, lawyer, procurator and public prosecutor are shown in Figure 71 and Figure 72.

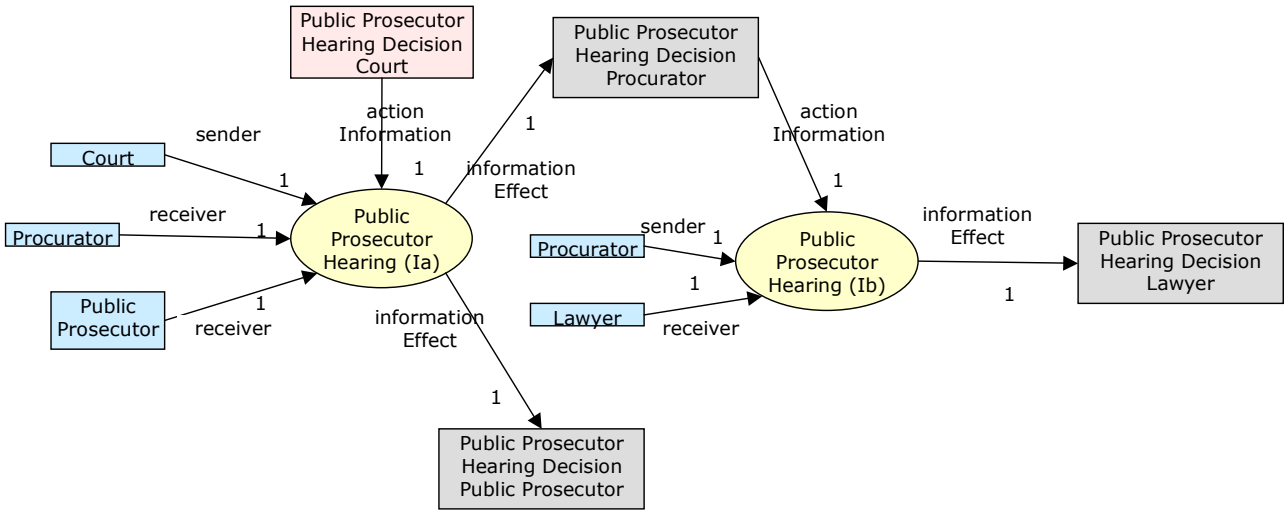


Figure 71. Public prosecutor hearing (first part)

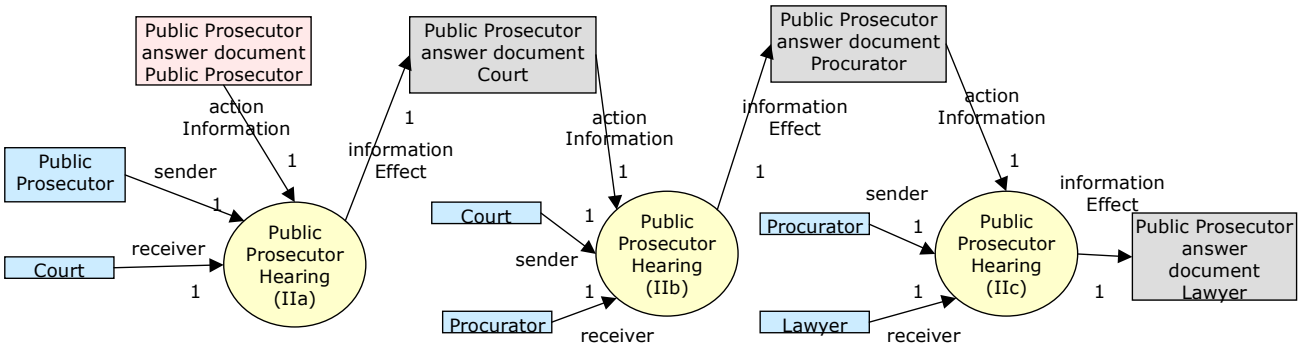


Figure 72. Public prosecutor hearing (second part)

11.3.9 Discovery and examination of evidence

If the *Public prosecutor hearing* action is not needed in a divorce, the first common action for plaintiff and respondent is *Discovery and examination of evidence*. In this action, the plaintiff and respondent parties, present the evidences they consider that are the most suitable for defending their claims in the divorce case. Court decides which ones are accepted and taken into account for the correct development of the case. Figure 73, Figure 74, Figure 75, Figure 76 and Figure 77 show the different subprocesses present inside this action.

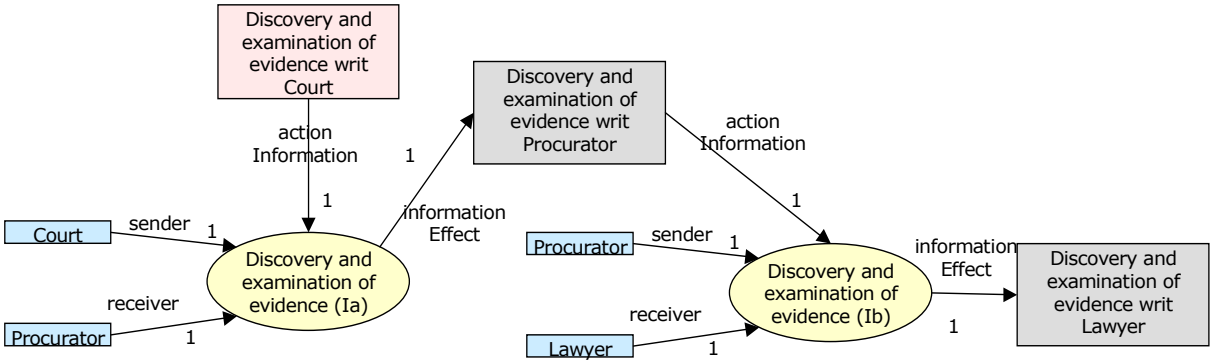


Figure 73. Discovery and examination of evidence (first part)

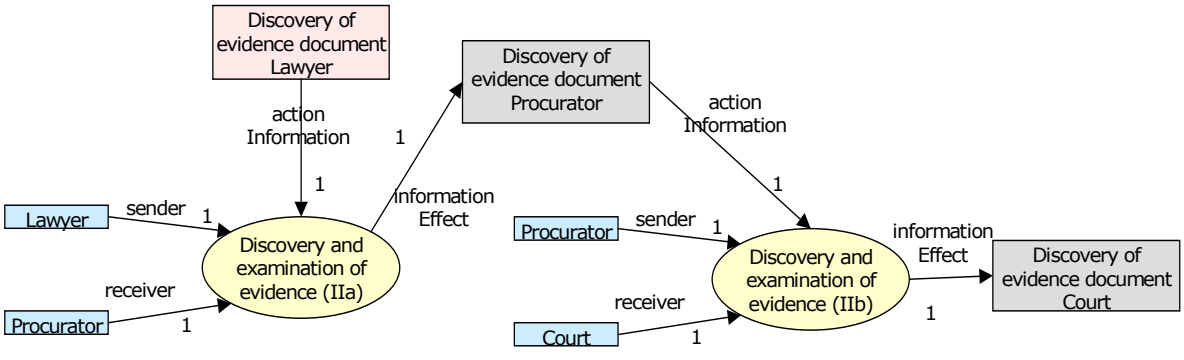


Figure 74. Discovery and examination of evidence (second part)

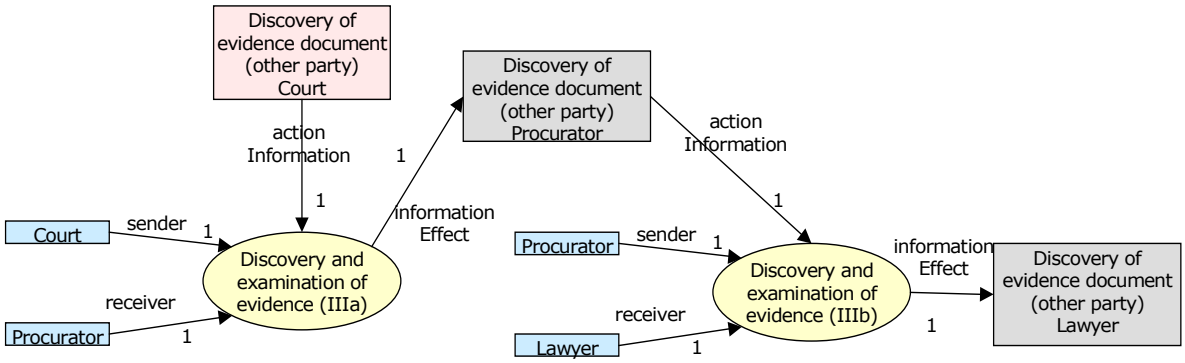


Figure 75. Discovery and examination of evidence (third part)

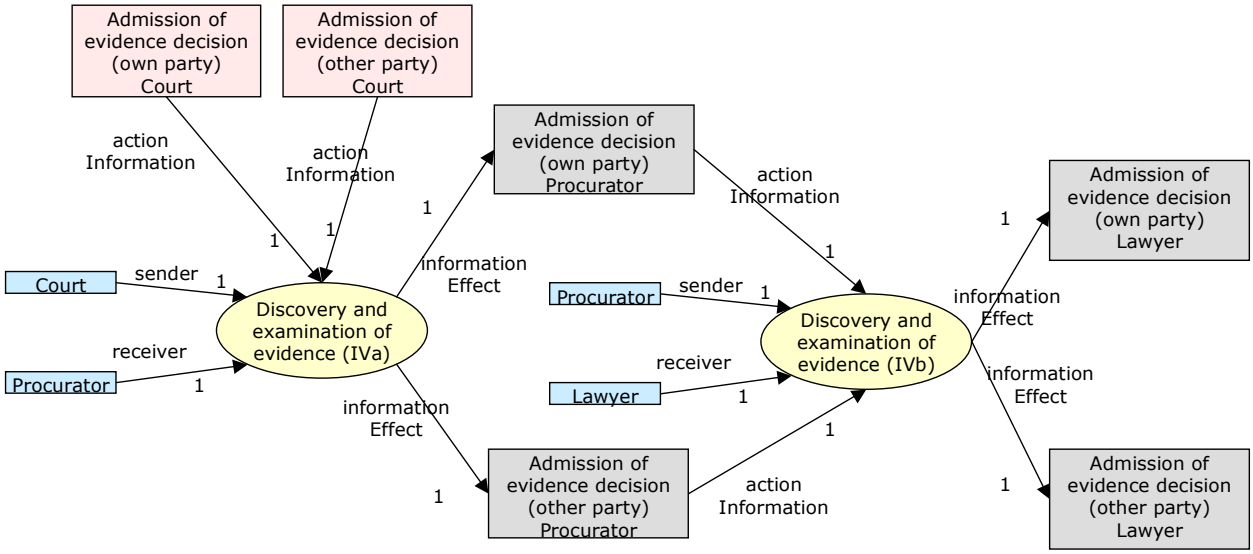


Figure 76. Discovery and examination of evidence (fourth part)

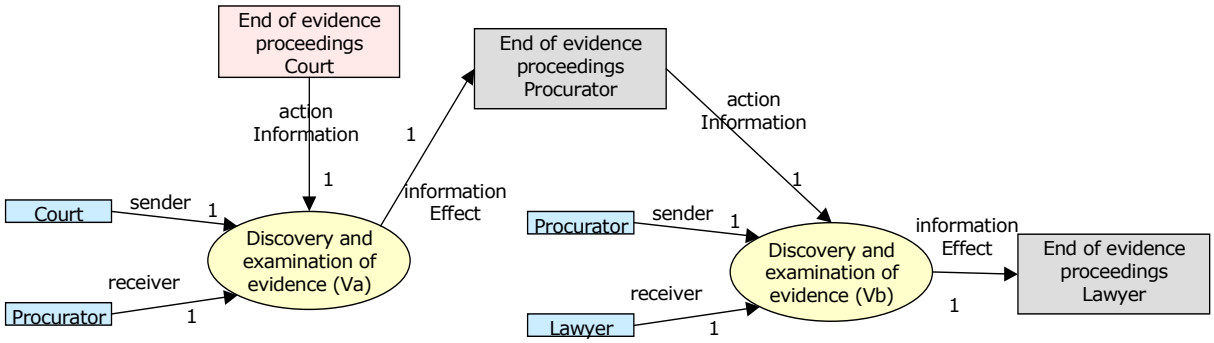


Figure 77. Discovery and examination of evidence (fifth part)

11.3.10 Trial

In the *Trial* action, the plaintiff and respondent parties interchange several documents with court. Nevertheless, the most relevant part of this action cannot take place electronically or by means of document interchanges because the parties have to go to court of law to present their different

positions in order to decide the divorce case. Figure 78, Figure 79, Figure 80, Figure 81, Figure 82 and Figure 83 show the different subprocesses inside this action.

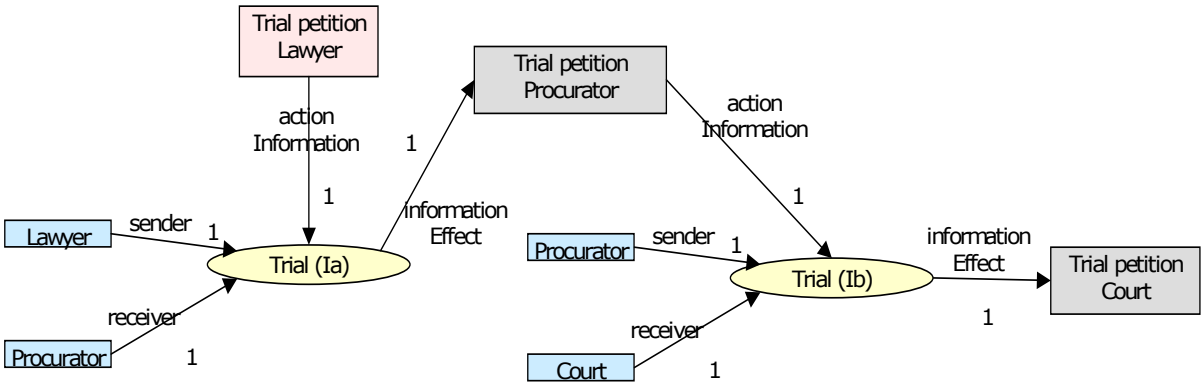


Figure 78. Trial (first part)

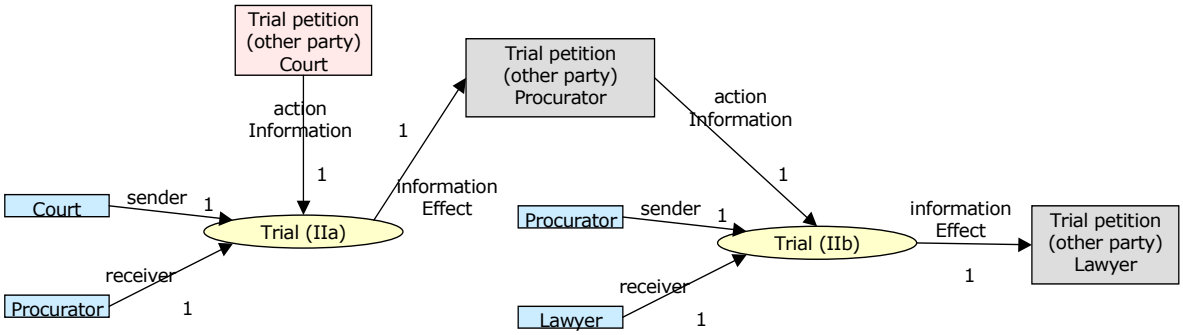


Figure 79. Trial (second part)

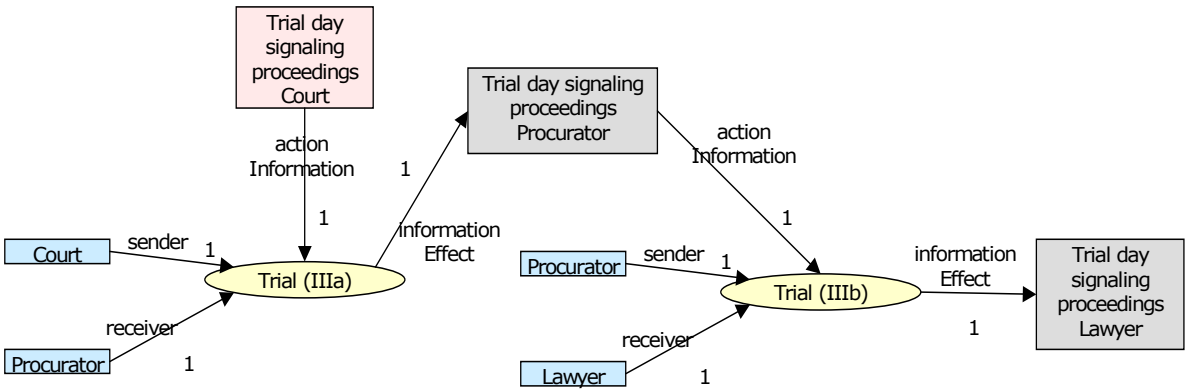


Figure 80. Trial (third part)

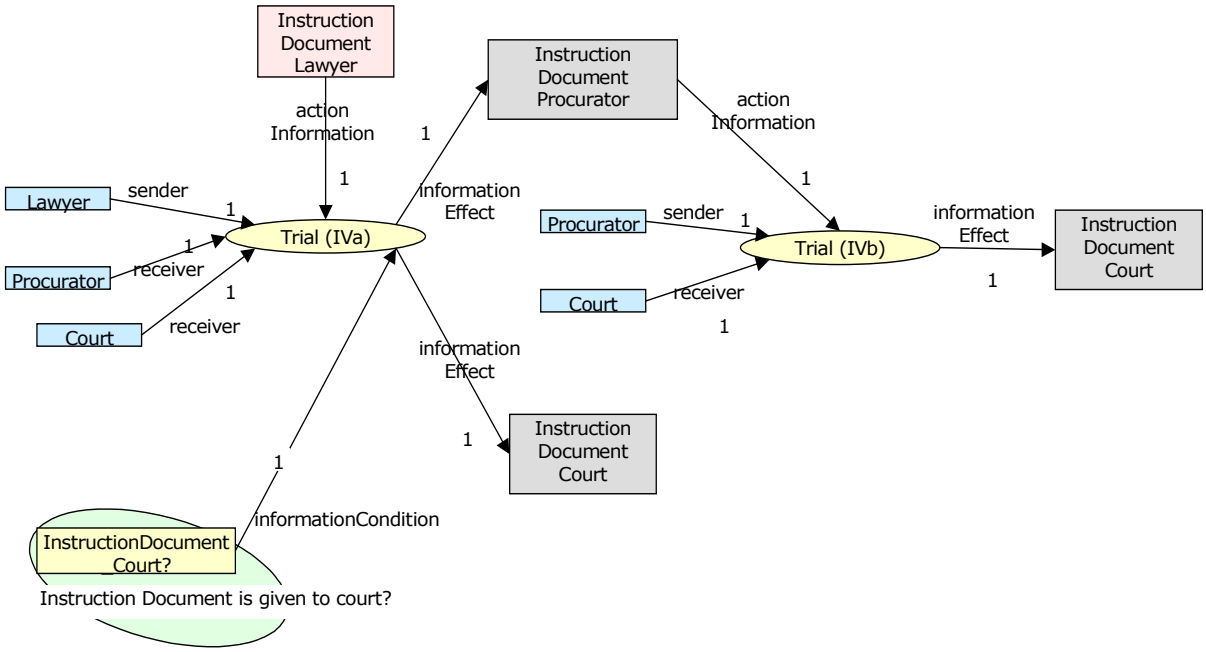


Figure 81. Trial (fourth part)

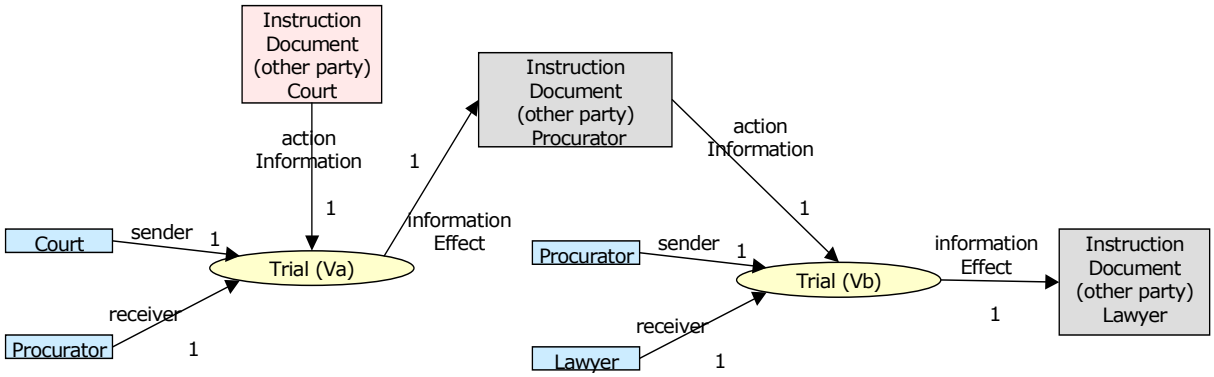


Figure 82. Trial (fifth part)

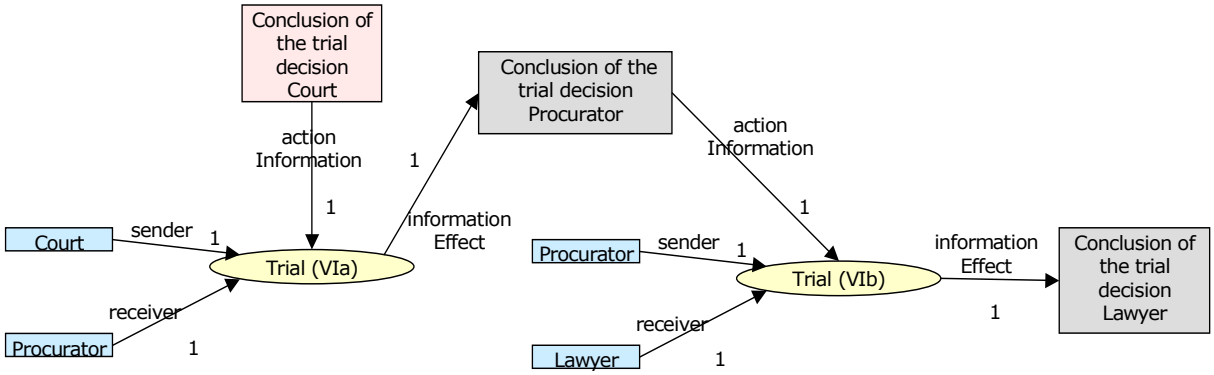


Figure 83. Trial (sixth part)

11.3.11 Judgment

In the *Judgment* action, the Court party delivers the judgment to the plaintiff and respondent parties. After this, two actions are possible, the *Enforcement of judgment* or the *Appeal* if one of the parties or both do not agree on the judgment. This is a quite simple action, because only the Judgment document is given from Court to Client, passing through Procurator and Lawyer. Figure 84 shows the subprocesses present in this action.

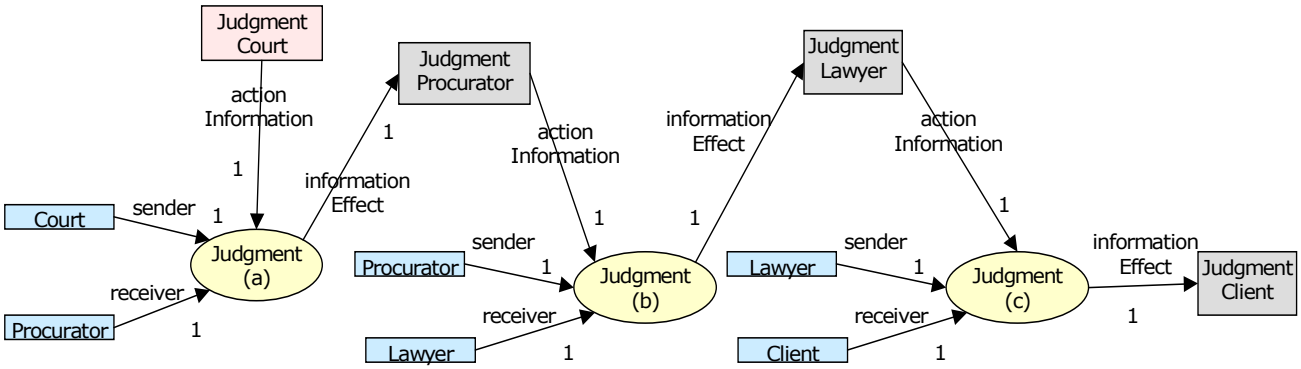


Figure 84. Judgment

11.3.12 Enforcement of Judgment

If any of the parties agrees with the *Judgment*, but the other one does not accomplish with what the *Judgment* says, an *Enforcement of Judgment* is done to force the other party to accomplish the *Judgment*. Lawyer of the agreeing party sends a document requesting to Court the *Enforcement of Judgment*. Figure 85 and Figure 86 show the subprocesses present in this action.

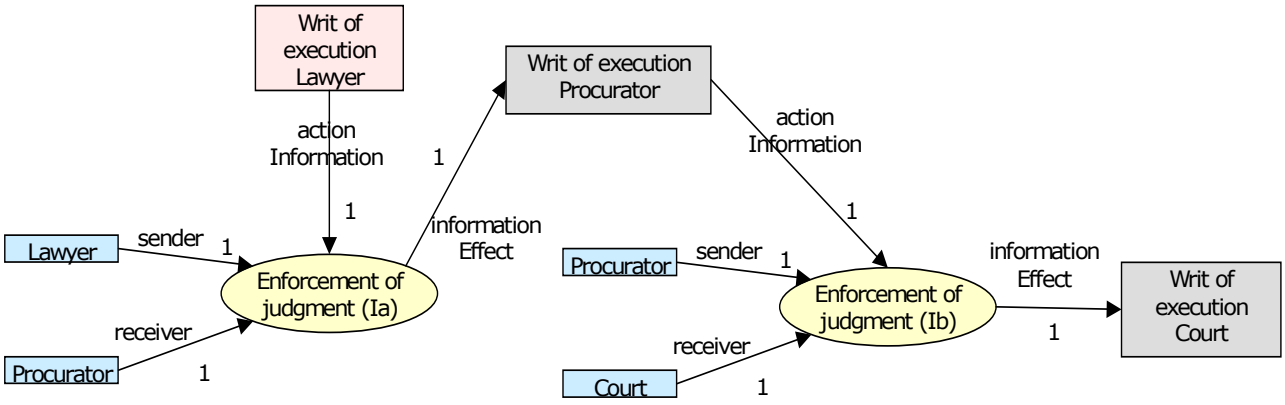


Figure 85. Enforcement of judgment (first part)

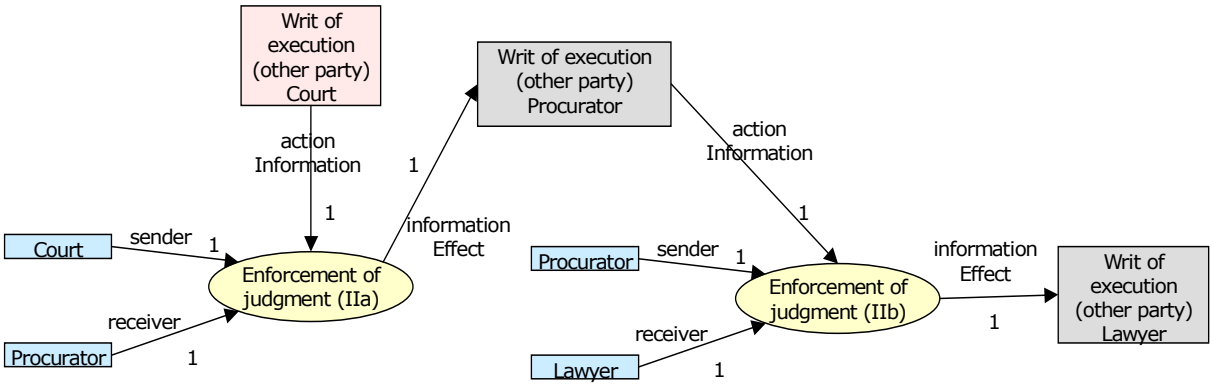


Figure 86. Enforcement of judgment (second part)

11.3.13 Appeal

If one or both parties disagree in the *Judgment*, they *Appeal* it, starting a new legal case in front of a different Court. To do so, the party or parties not agreeing present a document for requesting the judicial review. This document is sent to the other party and a new legal case may be started after this, but this is out of the scope of this specific divorce service. Figure 87 and Figure 88 show the subprocesses of this action.

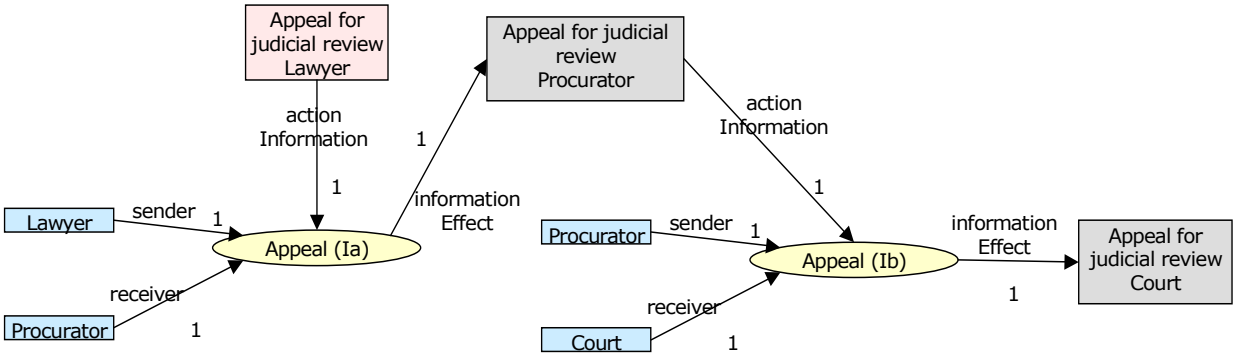


Figure 87. Appeal (first part)

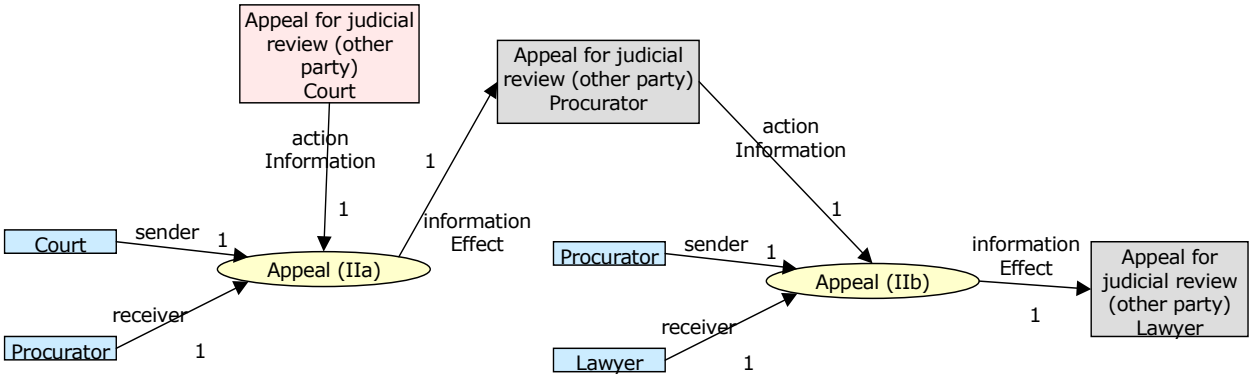


Figure 88. Appeal (second part)

Appendix B. Collaborative editing e-service

11.4 Introduction

In this appendix it is described in detail each of the collaborative editing e-service state diagrams appearing on section 9.2, Collaborative editing e-service. They depend on the different user roles we have inside collaborative editing service: moderator, editor, commentator and viewer. We can determine the operations or processes based on the transitions present on each state diagram. These processes are described by means of DAML-S, indicating inputs, outputs, preconditions and effects that represent what happens with the participants and documents inside collaborative editing e-service. The document state diagram can be extracted from the state diagrams of the different user roles, so it is also presented.

11.5 State diagram for moderator

Moderator user role can be in three different states inside collaborative editing:

- Idle, when the edition has not been started.
- Waiting for turn requests from commentators and editors.
- Editing the document when a commentator or editor has the turn for edition.

The transitions between these states are shown in Figure 59.

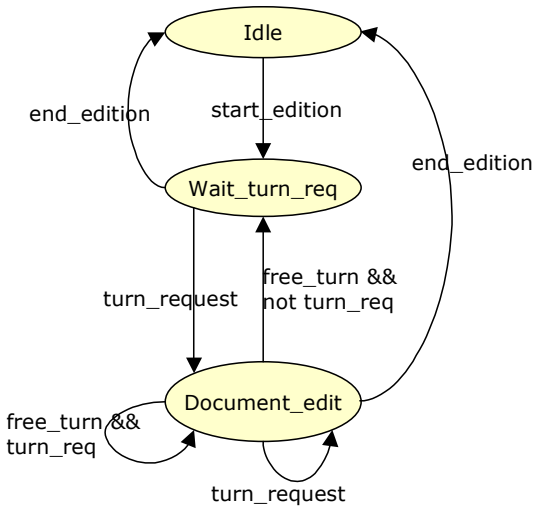


Figure 89. State diagram for moderator inside collaborative editing

Table 37 shows the transitions defined in the state diagram for moderator.

Table 37. Transitions for moderator

Initial state	End state	Event	Condition
Idle	Wait_turn_req	start_edition	
Wait_turn_req	Document_edit	turn_request	
Wait_turn_req	Idle	end_edition	
Document_edit	Document_edit	turn_request	
Document_edit	Document_edit	free_turn	turn_req
Document_edit	Wait_turn_req	free_turn	not turn_req
Document_edit	Idle	end_edition	

11.6 State diagram for editor and commentator

Editor and commentator user roles can be in four different states inside collaborative editing:

- Idle, when the edition has not been started.
- Viewing edition, when they have not requested a token for editing.
- Waiting for token, when they have done a turn request.
- Editing the document when they have the turn for edition.

The transitions between these states are shown in Figure 90.

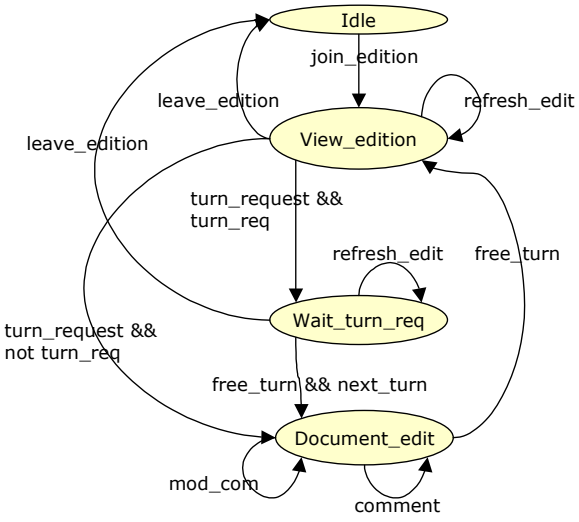


Figure 90. State diagram for editor and commentator in collaborative editing

Table 38 shows the transitions defined in the state diagram for editor and commentator user roles.

Table 38. Transitions for editor and commentator

Initial state	End state	Event	Condition
Idle	View_edition	join_edition	
View_edition	View_edition	refresh_edit	
View_edition	Wait_turn_req	turn_request	turn_req
View_edition	Document_edit	turn_request	not turn_req
View_edition	Idle	leave_edition	
Wait_turn_req	Wait_turn_req	refresh_edit	
Wait_turn_req	Document_edit	free_turn	next_turn
Wait_turn_req	Idle	leave_edition	
Document_edit	Document_edit	modification, comment (mod_com)	
Document_edit	Document_edit	comment	
Document_edit	View_edition	free_turn	

11.7 State diagram for viewer

Viewer user role can be in two different states inside collaborative editing:

- Idle, when the edition has not been started.
- Viewing edition, when they have joined the edition.

The transitions between these states are shown in Figure 91.

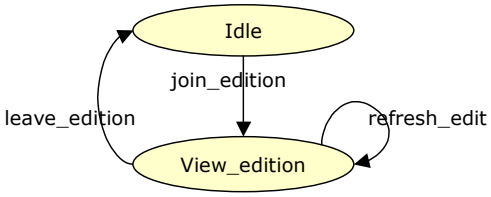


Figure 91. State diagram for viewer in collaborative editing

Table 39 shows the transitions defined in the state diagram for viewer user role.

Table 39. Transitions for viewer

Initial state	End state	Event
Idle	View_edition	join_edition
View_edition	View_edition	refresh_edit
View_edition	Idle	leave_edition

11.8 Operations

From the transitions presented in previous sections we can extract the operations or processes that each user role can perform. They are explained in Table 40.

Table 40. Operations by user role

User	Operation	Meaning
Moderator	Start_edition	The edition of the document starts
	End_edition	The edition of the document ends
	Turn_request	The turn is given to a user that previously requested turn, if he is still in the collaborative editing service
	Free_turn	The turn is freed. If there are pending turns, the turn is given to the next user in the queue. If not, moderator goes to a waiting state.
Editor	Join_edition	The user participates in the collaborative editing service
	Leave_edition	The user leaves the collaborative editing service
	Turn_request	The user requests turn for editing or commenting the document
	Modification	The user modifies the document
	Comment	The user makes a comment over the document
	Refresh_edition	The view of the document is refreshed, to reflect the last changes over the document
Commentator	Join_edition	The user participates in the collaborative editing service
	Leave_edition	The user leaves the collaborative editing service
	Turn_request	The user requests turn for editing or commenting the document
	Comment	The user makes a comment over the document
	Refresh_edition	The view of the document is refreshed, to reflect the last changes over the document
Viewer	Join_edition	The user participates in the collaborative editing service
	Leave_edition	The user leaves the collaborative editing service
	Refresh_edition	The view of the document is refreshed, to reflect the last changes over the document

11.9 Representation of the collaborative editing service processes using DAML-S

We can represent each of the operations or processes inside collaborative editing using DAML-S IOPEs (Input/Output/Precondition/Effect). Figures from 91 to 103 are the graphical representation of each of these processes. These processes are also listed on Table 43.

Figure 92 shows the *Start Edition* process. On it, the moderator user role starts the edition. As a consequence of this, the document is opened, the edition participants (Editors and commentators) go to idle state and the moderator goes to a waiting state, where he waits for receiving operation invocations from the edition participants.

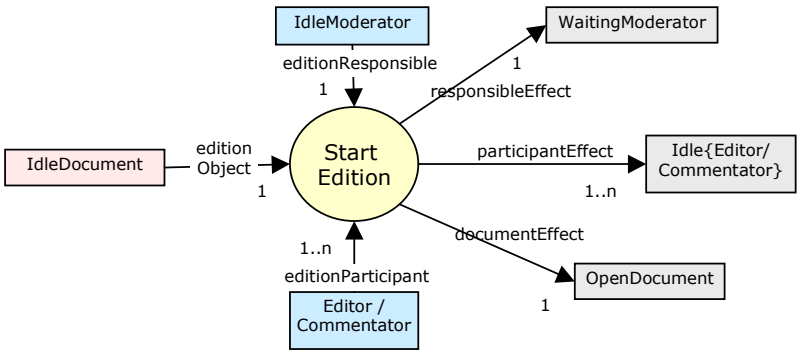


Figure 92. Start Edition process

Figure 93 shows the *End Edition* process. On it, the moderator user role ends edition. As a consequence of this, the document is closed, passing to idle state and the different user roles go to idle state, including moderator.

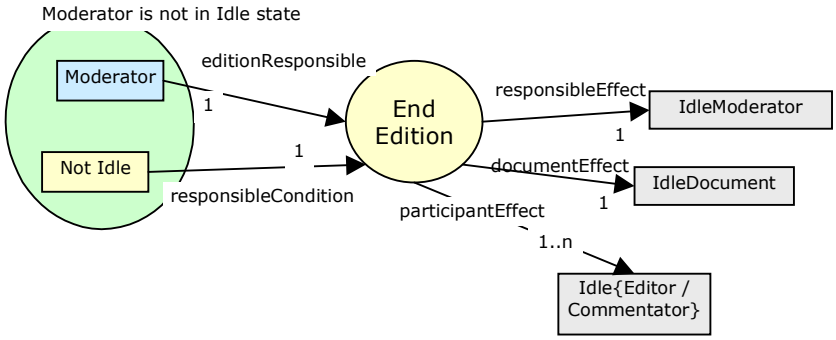


Figure 93. End Edition process

Figure 94 shows the *Turn Request* process when nobody is editing the document. In this case, the moderator gives the turn to the user that has requested it. The result of this is that the document, the moderator and the user requesting turn go to the edit state and the modifications or comments over the document can be done.

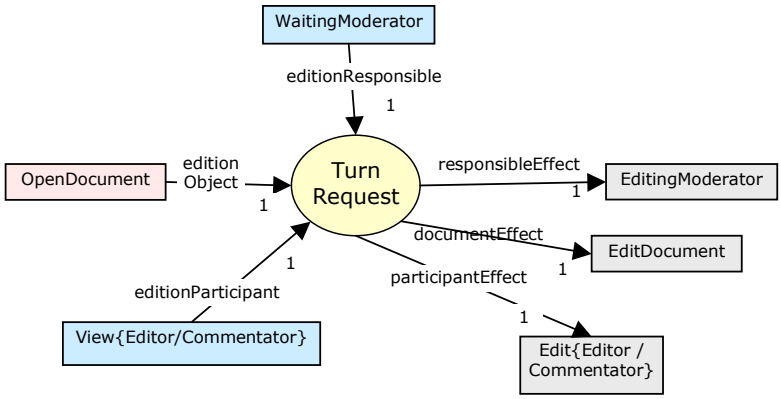


Figure 94. Turn request process when nobody is editing

Figure 95 shows the *Turn Request* process when somebody is editing the document. In this case, the moderator takes note that the user has requested turn. The document and the rest of users remain in the same state, but the user requesting turn goes to a waiting for turn state.

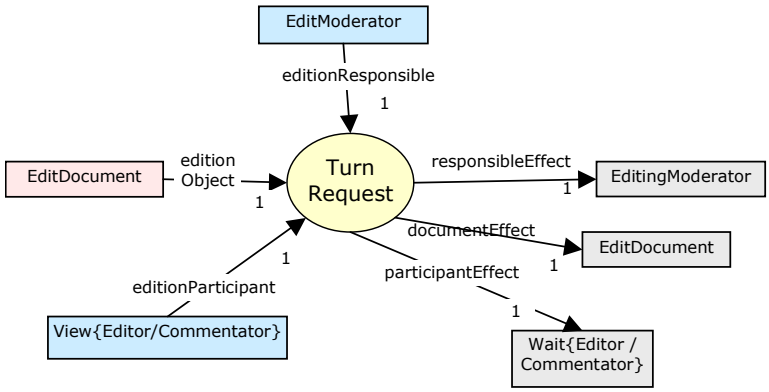


Figure 95. Turn request process when there is somebody editing

Figure 96 shows the *Free Turn* process when there are pending turns. In this case, the moderator takes the first turn request from his list for giving turn to the corresponding user. If the user has left edition, the turn is given to the next one in the list. If all users in the turn request list have left the edition, then the process that should be invoked is the one represented by Figure 97, where no more turns are requested after a free turn. If the turn is given to a user that requested it, the effect of this process is that moderator and document remain in the edit state, the user that freed the turn goes to the view state and the next user in the turns request list goes to the editing state, being able to make modifications or comments over document (depending on his/her user role).

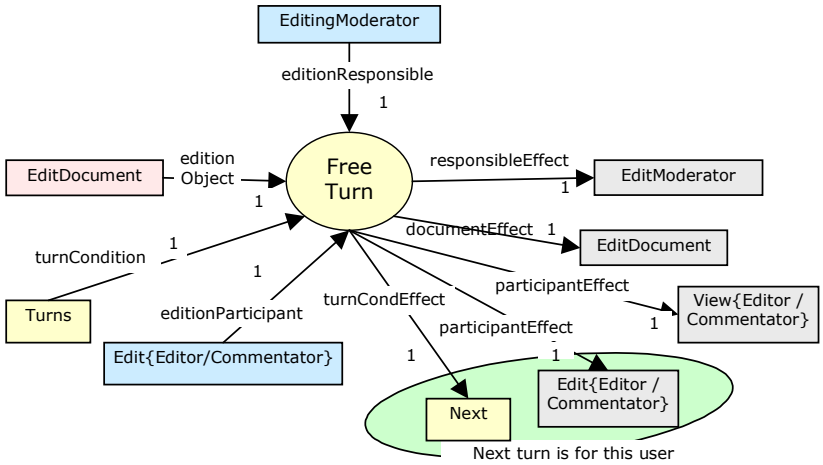


Figure 96. Free turn process with turns requested

Figure 97 shows the *Free Turn* process when no more turns are requested. In this case, the document goes to its open state, the user that freed the turn goes to the view state and the moderator enters in the waiting state.

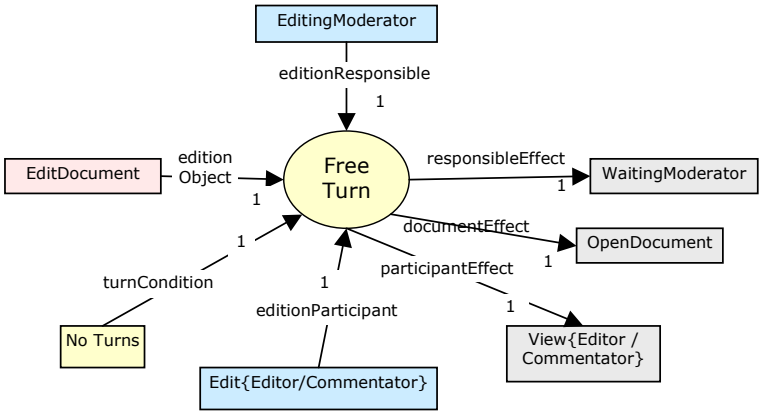


Figure 97. Free turn process with no more turns requested

Figure 98 shows the *Join Edition* process. This process can occur after the *Start edition* process has been invoked. The user (editor, commentator or viewer) that invokes this operation, enter into the view state.

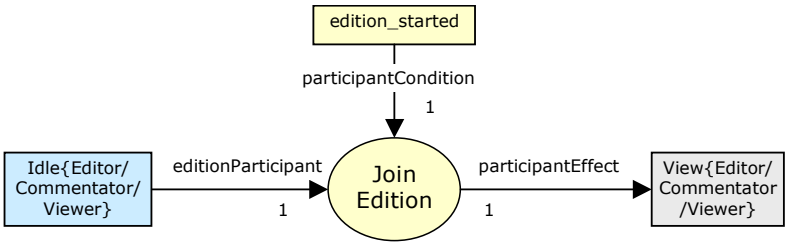


Figure 98. Join edition process

Figure 99 shows the *Leave Edition* process. The effect of this process is that the user invoking it returns to the idle state.

Figure 99. Leave edition process

Figure 100 shows the *Modification* process. This operation can only be invoked by the editor user role. The result of this operation is that the document is modified, but the document, moderator or user states remain equal.

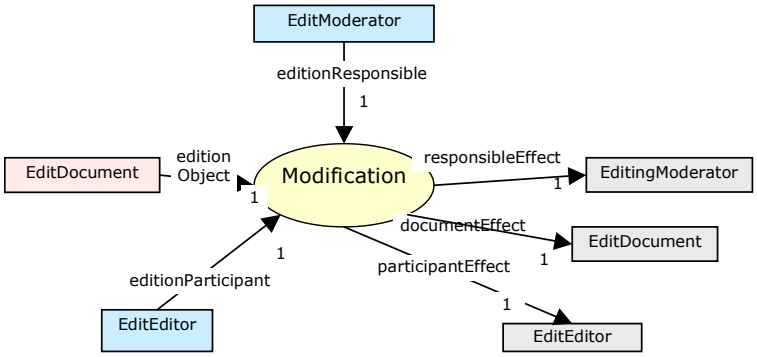


Figure 100. Modification process

The comment event involves several processes:

- Make comment: A commentator or editor makes a comment over the document.
- Accept comment: An editor accepts a previous comment made by another user.
- Reject comment: An editor rejects a previous comment made by another user.

Figure 101 shows the *Make Comment* process. This operation can be invoked by the editor and commentator user roles. The result of this operation is that the document has a new comment, but the document, moderator or user states remain equal.

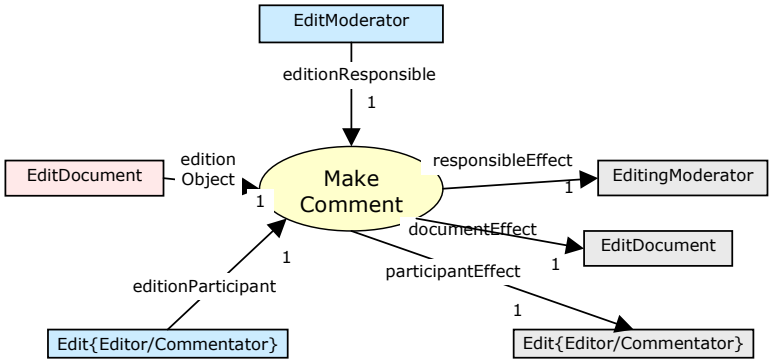


Figure 101. Make comment process

Figure 102 shows the *Accept Comment* process. This operation can be invoked only by the editor user role. The result of this operation is that a comment is accepted and possibly added as a modification of the document, but the document, moderator or user states remain equal.

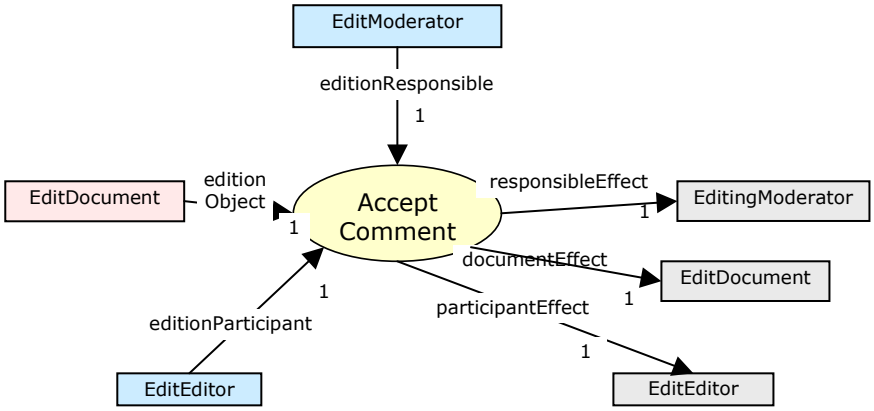


Figure 102. Accept comment process

Figure 103 shows the *Reject Comment* process. This operation can be invoked only by the editor user role. The result of this operation is that a comment is rejected and deleted from the document, but the document, moderator or user states remain equal.

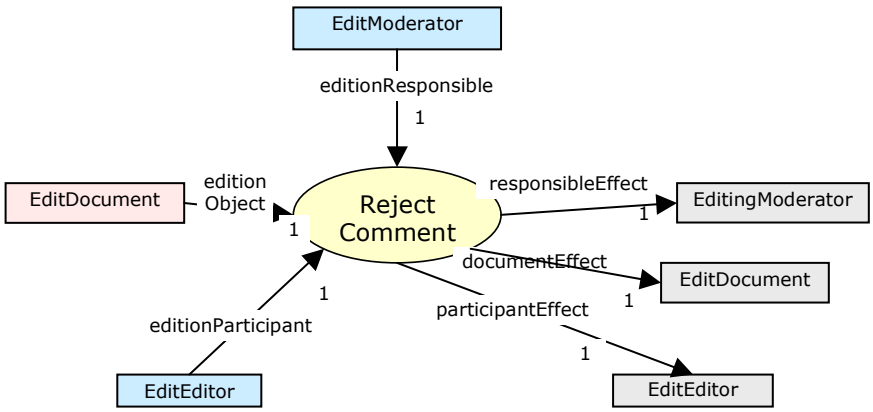


Figure 103. Reject comment process

Figure 104 shows the *Refresh Edition* process. This result of this operation is that the user invoking it has a fresh copy of the document being edited.

Figure 104. Refresh edition process

On Table 43 we have listed the collaborative editing processes. We have grouped them by user role, indicating which of them, moderator, commentator, editor or viewer starts the process or has the main responsibility. The meaning of each column is described next:

- Process: The name of the process being described.
- Invoked by: User that starts the process or has the main responsibility.
- Moderator State: State of the moderator user. We can have three different values, Idle, Editing and Waiting.
- Editor / Commentator / Viewer State: State of the editor/commentator/viewer user. We can have four different values, Idle, Viewing, Editing and Waiting.

- Document State: State of the document. We can have three different values, Idle, Open and Edit.
- Moderator Condition: Condition that has to be accomplished by moderator user in order to invoke this process.
- Editor / Commentator / Viewer Condition: Condition that has to be accomplished by editor, commentator or viewer users for this process to be invoked.
- Moderator Effect: Effect over the moderator user after invoking the process. This can be seen as a destination state and it can have three values, Idle, Waiting and Editing.
- Editor / Commentator / Viewer Effect: Effect over the moderator user after invoking the process. This can be seen as a destination state and it can have four different values, Idle, Viewing, Editing and Waiting.
- Document Effect: Effect over the document after invoking the process. This can be seen as a destination state and it can have three different values, Idle, Open and Edit.

The meaning of the states and effects for moderator are related with its state diagram, described in 11.5 State diagram for moderator. Table 41 shows this mapping.

Table 41. Mapping among moderator states in collaborative editing processes and moderator state diagram

Moderator State	State diagram state
Idle	Idle
Waiting	Wait_turn_req
Editing	Document_edit

The meaning of the states and effects for editor, commentator and viewer are related with their state diagrams, described in 11.6, State diagram for editor and commentator and 11.7, State diagram for viewer, respectively. Table 42 shows this mapping.

Table 42. Mapping among editor, commentator and viewer states in collaborative editing processes and their state diagrams

Editor / Commentator / Viewer State	State diagram state
Idle	Idle
Viewing	View_edition
Waiting	Wait_turn_req
Editing	Document_edit

From the processes described in Table 43, we can extract the state diagram for a document, that is described in 11.10 State diagram for document.

Table 43. Processes inside collaborative editing

Process	Invoked by	Moderator (M) State	Editor (E) Commentator (C) Viewer (V) State	Document State	M Condition	E/C/V Condition	Moderator Effect	E/C/V Effect	Document Effect
start_ edition	M	Idle	N/A	Idle	N/A	N/A	Waiting	Idle	Open
end_ edition	M	Waiting Editing	N/A	N/A	N/A	N/A	Idle	Idle	Idle
join_ edition	E	N/A	Idle	N/A	N/A	edition_ started	N/A	Viewing	N/A
leave_ edition	E	N/A	Viewing Waiting	N/A	N/A	N/A	N/A	Idle	N/A
turn_ request	E	Waiting	Viewing	Open	N/A	N/A	Editing	Editing	Edit
turn_ request	E	Editing	Viewing	Edit	N/A	N/A	Editing	Waiting	Edit
free_ turn	E	Editing	Editing	Edit	turn_req / next_ turn	N/A	Editing	Viewing / Editing	Edit
free_ turn	E	Editing	Editing	Edit	not turn_req	N/A	Waiting	Viewing	Open
modificat ion	E	Editing	Editing	Edit	N/A	N/A	Editing	Editing	Edit
make_ comment	E	Editing	Editing	Edit	N/A	N/A	Editing	Editing	Edit
accept_ comment	E	Editing	Editing	Edit	N/A	N/A	Editing	Editing	Edit
reject_ comment	E	Editing	Editing	Edit	N/A	N/A	Editing	Editing	Edit
refresh	E	N/A	Viewing	N/A	N/A	N/A	N/A	Viewing	N/A
join_ edition	C	N/A	Idle	N/A	N/A	edition_ started	N/A	Viewing	N/A
leave_ edition	C	N/A	Viewing Waiting	N/A	N/A	N/A	N/A	Idle	N/A
turn_ request	C	Waiting	Viewing	Open	N/A	N/A	Editing	Editing	Edit
turn_ request	C	Editing	Viewing	Edit	N/A	N/A	Editing	Waiting	Edit
free_ turn	C	Editing	Editing	Edit	turn_req / next_ turn	N/A	Editing	Viewing / Editing	Edit
free_ turn	C	Editing	Editing	Edit	not turn_req	N/A	Waiting	Viewing	Open
make_ comment	C	Editing	Editing	Edit	N/A	N/A	Editing	Editing	Edit
refresh	C	N/A	Viewing	N/A	N/A	N/A	N/A	Viewing	N/A
join_ edition	V	N/A	Idle	N/A	N/A	edition_ started	N/A	Viewing	N/A
leave_ edition	V	N/A	Viewing	N/A	N/A	N/A	N/A	Idle	N/A
refresh	V	N/A	Viewing	N/A	N/A	N/A	N/A	Viewing	N/A

11.10 State diagram for document

Based on the processes we have described for collaborative editing, we can define the state diagram for a document being produced in the collaborative editing service. Figure 105 shows this state diagram.

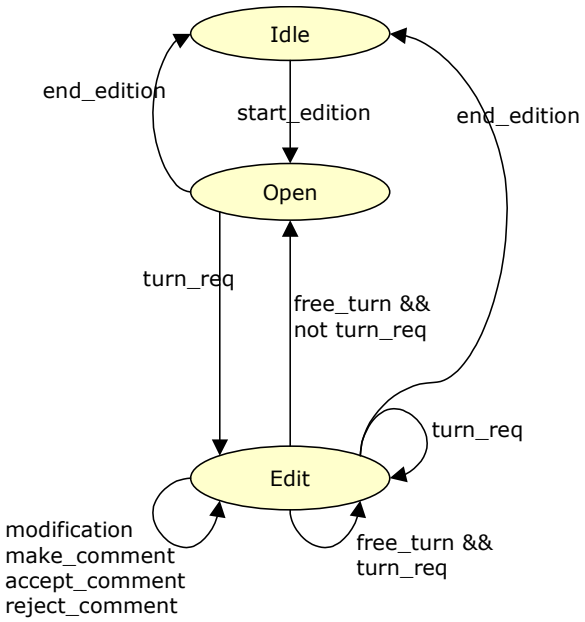


Figure 105. State diagram for document based on collaborative editing processes

The transitions present in Figure 105 diagram are summarised in Table 44.

Table 44. Transitions for document

Initial state	End state	Event	Condition
Idle	Open	start_edition	
Open	Idle	end_edition	
Open	Edit	turn_req	
Edit	Idle	end_edition	
Edit	Open	free_turn	turn_req
Edit	Edit	turn_req	
Edit	Edit	free_turn	not turn_req
Edit	Edit	modification	
Edit	Edit	make_comment	
Edit	Edit	accept_comment	
Edit	Edit	reject_comment	