ebXML Registry and Repository Part 1: Business Domain

ebXML Working Draft issue date May 11, 2000

Version: 1.0

Project Team Lead: Scott Nieman, Norstan Consulting, USA
Editor: Joe Dalman, TIE Commerce, USA
Contributors: Scott Hinkelman, IBM, USA; Terry Allen, CommerceOne, USA; Lisa Carnahan, DOC/National Institute of Standards and Technology, USA; Eric Jarry, Software AG, France; Kunio Mizoguchi, Ecom Japan; Jong L Kim, Inno Digital Co., Ltd, Korea; Michael Rowley, Ph.D, eXcelon Corp, USA; Robert Miller, Global eXchange Services, USA; Sally Fuger, Ford Motor Company, USA; Yutaka Yoshida, Sun Microsystems, USA; Nagwa Abdelgfour, Sun Microsystems, USA; Gait Boxman, TIE Product Development, Holland

Abstract

This document is Part 1 of the Functional Requirement Specification for an ebXML-compliant registry and repository. Part 1 defines the scope of the workflow to interact with a registry and repository, the actors involved in these interactions and the overall Domain Architecture for the e-Business Requirements of the registry and the repository. The Domain Architecture provides the basis of the detailed workflow specifications as defined in *ebXML Registry and Repository Part 2: e-Business Requirements* as well as forming the basis of traceability between Part 1 and Part 2.

Status of this Document

This is the first public working draft of the *ebXML Registry and Repository Part 1: Business Domain*, issued by the ebXML Registry and Repository Project Team for public review and by members and project teams of ebXML.

This working draft incorporates the decisions of the ebXML Registry and Repository project team as of May 11, 2000. The first comment period opens May 15, 2000 and ends May 29, 2000. Comments with associated starting line number and ending line number, and proposed changes (required) should be emailed to joe.dalman@tiecommerceusa.com in DocBook, MS Word or plain text format. We will not accept this document edited with revisions on.

During the first comment period, the project team will review public comments while continuing its development of *ebXML Registry and Repository Part 2: e-Business Requirements*. Part 2 will focus on the detailed workflows that highlight the business-to-business (B2B) interchanges between users, a registry, and a repository.

Table of Contents

1 Introduction

This *ebXML Registry and Repository Part I: Business Domain* working draft defines the scope and functional requirements of an ebXML-compliant registry and repository.

Section 2 defines the actors that can interact with a registry and repository, and represents the relationships between these actors in an Actor Relationship diagram. This diagram is a hierarchical view that depicts actor inheritance and specialization as well as dependencies between actors.

Section 3 defines the scope of the registry and repository as reflected by a high-level business domain Use Case. The use cases define the usage viewpoints from the perspective of several actors, including that of an ebXML business application.

Section 4 defines the ebXML Registry and Repository overall architecture through the use of a Domain Package diagram. Each package categorizes general functionality that could potentially become the software components, but not necessarily, since some functionality may be embedded within another service.

1.1 Purpose

The purpose of this working document is to define the overall system boundary as defined in the UN/CEFACT/TMWG UML Profile and Methodology. It is to define the scope of the overall system and provide a reasonable understanding of the size of the project.

The methodology covers four main workflows as shown below, as this working document is the first of four documents.

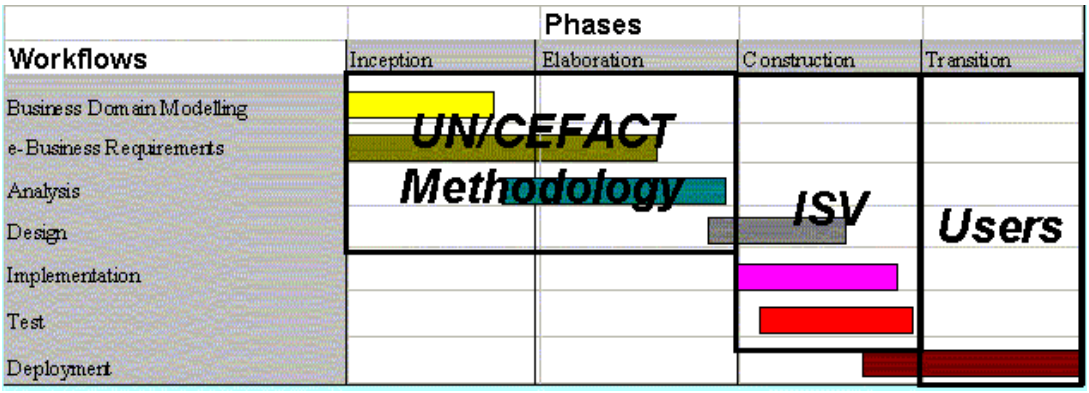

Figure 1. Phases and Workflows

Ultimately, the result of the methodology will enable the ebXML Registry and Repository Project Team to apply Production Rules to generate the XML interfaces for both the Registry and the Repository. Software vendors can use the XML interfaces specifications as well as the UML models to understand the static and dynamic behavior of the system in order to construct the systems. With these specifications, conformance testing can be more clearly defined, since software components must conform to the XML interfaces, and the interactions to the system should be according to the UML activity and sequence diagrams.

It should be understood that this working document defines the scope for a network of registries and a network of repositories. This approach fits the model of world wide Internet, incorporating a vast number of interdependencies between companies, vertical domains relying on other vertical domains (finance, logistics, purchasing) and specifications relying on other specifications. It is envisioned that existing EDI directories could become repositories, and their traditional references to outside associations that maintain code lists (i.e., phone numbers, mailing addresses) could be replaced by ebXML based links from one repository to another.
While some organizations advocate the philosophy of a single global repository, ebXML Registry and Repository disagrees with that notion for the following reasons:

1) The approach will not scale
2) It does not fit the World Wide Web model
3) User associations, consortiums and standards organizations do not trust the notion of one single organization maintaining such a repository, deciding what should be in the repository and what should not be in the repository.

1.2 Dependencies on other Specifications

The Unified Modeling Language version 1.3, as issued by the Object Management Group, is used to define the entire ebXML Registry and Repository specification. The intent is a top-down model driven design approach versus a bottom-up approach based on data models or hand developed DTDs and brute force software development. This allows the business needs to drive the technology usage, provides a technology neutral representation of the specifications, and allows for the auto-generation of software code including XML specifications.

The UN/CEFACT/TMWG UML Profile and Methodology, which defines how and which UML artifacts need to be produced, is being utilized. It is adapted from the Rational Unified Process and specifically focused on e-Business Process modeling, including business-to-business (B2B) transactions. While its intent is to provide a protocol-neutral method of defining e-Business Processes, it also includes sample methods to auto-generate XML interfaces from the UML model by applying Production Rules. These XML interfaces follow a request/response model for real-time interaction with a registry and repository. Therefore the XML interfaces are intended to invoke object methods by passing XML requests and returning XML document responses.

EbXML Registry and Repository project team will consider any W3C Recommendations in the design or implementation of a ebXML-compliant registry and repository. W3C specifications in progress such as Xlink and Xpointer are key specifications that will be monitored. Such specifications should be helpful in enabling internetworking of registries to repositories and repositories to repositories.

The Registry and Repository project team is considering transforming the Registry and Repository UML model to XML as according to the Object Management Group XML Metadata Interchange (XMI) v1.1 specification.

1.3 Conventions and Terminology

Many of the terms and definitions can be found in:

- ISO 11179
- OMG UML Specification
- TMWG Glossary
- OMG XMI Specification

They are not repeated here.

2 Actor Relationships

The complete Actor Relationship class diagram depicts the hierarchical relationships, the unidirectional associations, and the dependencies between the actors. The actor relationships are not intended to specify role base authorization. The registration authority shall define the security policy.

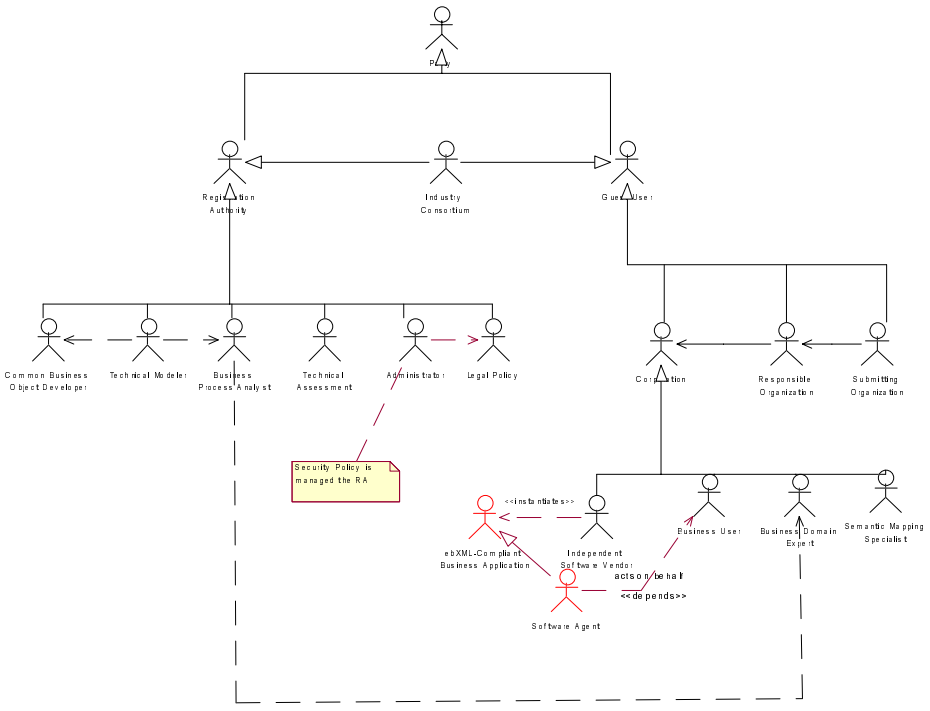

Figure 2.

The main areas in this hierarchy are elaborated upon below.

2.1 Guest User

The Guest User is an actor that inherits from the superclass actor Party. The Guest User is an organization or individual that acts on behalf of the organization.

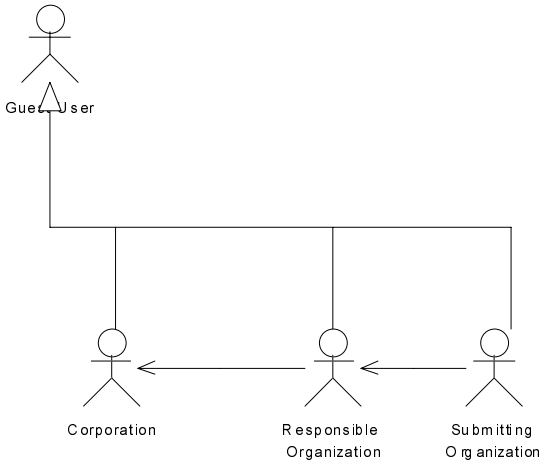

Figure 3.

Specialized actor classes of Guest User include the Corporation who is generally the end user of the registry and repository, the Responsible Organization and the Submitting Organization as defined by ISO 11179. Unidirectional associations are drawn between these organizations; e.g., the Submitting Organization acts on behalf of the Responsible Organization who in turns acts on behalf of the Corporation.

2.2 Corporation

The Corporation hierarchy does not have any other associations other than inheritance. Shown are four types of actors, three of which could reside in a typical corporate business environment,; specifically, the Business Domain Expert, Business User and the Semantic Mapping Specialist.

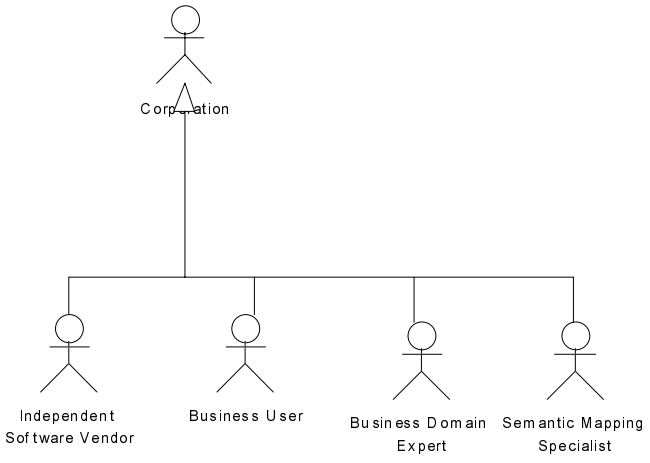

Figure 4.

The Business User could be from a large, medium or small corporate enterprise. The Business Domain Expert is an actor that represents the corporate needs in the development

of B2B specifications. The Semantic Mapping Specialist is an actor that reviews available B2B standards and specifications and maps them to the corporate internal business processes and application metadata representations. The Semantic Mapping Specialist traditionally supports EDI transactions or internal application-to-application (A2A) interfaces.

The Independent Software Vendor is a special type of Corporation that produces software that meets the needs of the Business Domain Expert. The business needs of the Business Domain Expert could imply that there is a unidirectional association between the Business Domain Expert and the Independent Software Vendor, however those needs are communicated through other channels, specifically through Technical Modelers (see Registration Authority).

2.3 Registration Authority

The Registration Authority (RA) is shown with several actors that could actually be part of the RA itself, subcontracted resources, or in some cases volunteers.

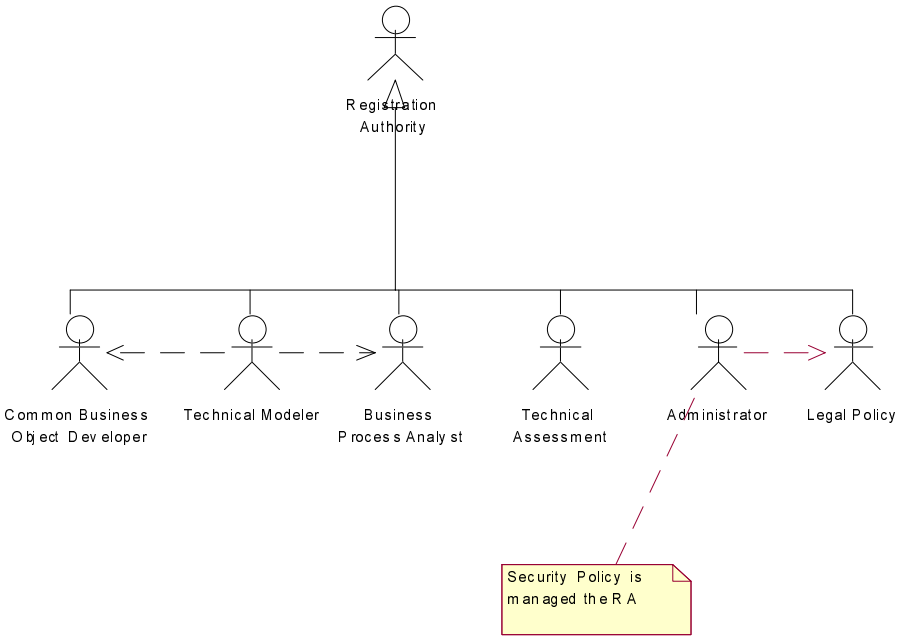

Figure 5.

Shown are six examples of roles within the Registration Authority. Three actors support the knowledge extraction process to define the business and functional requirements of B2B transactions: the Business Process Analyst, the Technical Modeler, and somewhat indirectly, the Common Business Object Developer. These role players are found in the UN/CEFACT Methodology for e-Business Process Modeling. The Business Process Analyst is a facilitator with the skills to ask questions of the Business Domain Expert and is vertical domain "neutral" in order to challenge the Business Domain Expert's stereotypical views of processes, and to identify business requirements including requirements that are not so obvious to the Business Domain Expert but have been experienced in past by the Business Process Analyst.

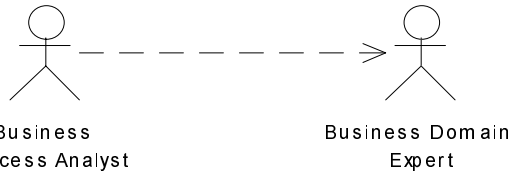

Figure 6.

The Technical Modeler is familiar with the selected UML modeling tool and knows how to interact with the repository for model development and to search for existing patterns. The Technical Modeler is reliant on the Business Process Analyst skills in extracting knowledge from the Business Domain Expert. The Common Business Object Developer is skilled in reviewing UML models and detecting "patterns" in both business process models (activity diagrams and sequence diagrams) and data / design patterns (class diagrams). The Technical Modeler is also dependent on the Common Business Object Developer, who has the ability to purchase reusable common business objects for inclusion into the models in development.

The Technical Assessment actor is responsible for UML model consistency and clarity. One task of the Technical Assessment actor is to identify semantic overlap, where equivalent semantic information is represented in different ways. Technical Assessment works with the Common Business Object Developer to resolve such issues. The Technical Assessment analyst is responsible for reviewing an industry submission of a model, identifying the submission's relevance, uncovering any semantic overlaps with other specifications in the repository, assigning model integration project plans, and issuing recommendations for the final approval of a proposed specification. Model integration is a step that harmonizes multiple submitted models that are indirectly inter-dependent to each other. This occurs primarily between vertical domains, in which each vertical domain names each semantic unit differently (ref: UN Layout Key).

The Administrator actor is responsible for the day-to-day operation of the ebXML Registry and / or Repository. This includes networking, server scalability, and usage statistics. The administrator has a thorough understanding of the overall *ebXML Architecture*, *ebXML Transport, Packaging, and Routing specifications*, and the APIs to the Registry and Repository as defined in *ebXML Registry and Repository Part 4: Design* (to track the number of API requests).

The legal/policy actor is responsible for the business arrangements, legal policies, and operational policies of the registry and repository.

2.4 *Industry Consortium*

The Industry Consortium can be a Registration Authority as well as a Guest User therefore multiple inheritance is shown. In reality, the inheritance to Registration Authority has optional cardinality, as the Industry Consortium may not choose to host its own ebXML-compliant Registry and Repository.

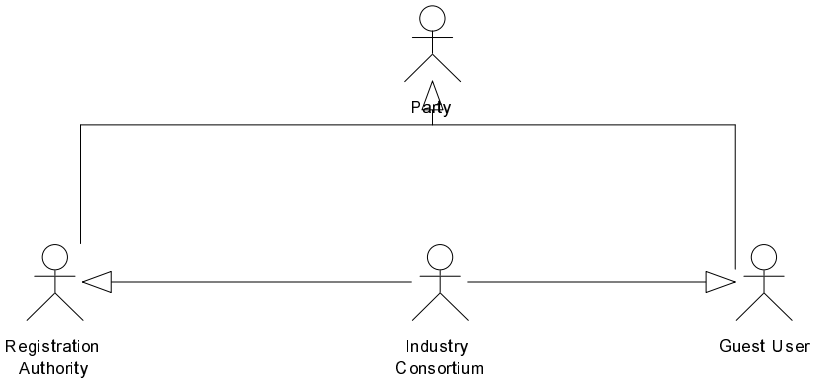

Figure 7.

2.5 ebXML Business Application

An ebXML Business Application is an application that knows how to communicate with the ebXML-compliant registry. An independent software vendor may develop such an application or it may be developed internally by a corporate development team using the ebXML specifications for a given problem domain. This is shown by the "**instantiates**" relationship between the Independent Software Vendor and the ebXML-compliant Business Application.

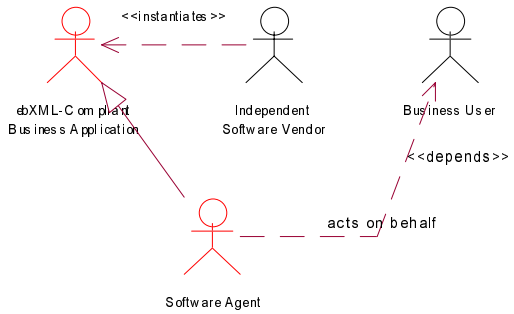

Figure 8.

A specialized actor class of the ebXML-compliant Business Application is the Software Agent. This is shown by an inheritance relationship to the ebXML-compliant Business Application actor. Software agents are based on artificial intelligence technology and are emerging in the industry at a rapid pace. A software agent acts on behalf of the Business User actor who configures the agent by specifying a unique profile that classifies a problem domain. This is noted by the "dependency" association between the Software Agent and the Business User. The unique behavior of the software agent is that it typically runs in an autonomous mode without initiation by the Business User, and presents to the Business User information that conforms to the profile. Software agents could monitor B2B transactions for anomalies, detect trends, find new trading partners, and configure B2B integration systems.

Software agents will be able to interact with an ebXML-compliant registry, and request specifications for e-Business Processes and any referenced data interchange specifications in the e-Business Process.

3 Domain Use Case

The overall scope of the registry and repository is the submission, classification, and storage of specifications and proposals, while allowing the creation of new specifications. The Domain Use Case provides an overall, high-level context of the business problem that doesn't delve into how the system performs the tasks on a functional basis. It provides a boundary to the scope of the functionality, and provides traceability to the actors involved as defined in the Actor Relationships hierarchical diagram.

All actors are human actors with the exception of the ebXML Business Application. (It does not appear as a different UML icon as suggested by other modeling conventions.) It is important to refer back to the Actor Relationship diagram to understand the various associations between the actors in the context of their roles. This approach enables a simple Domain Use Case diagram as shown below.

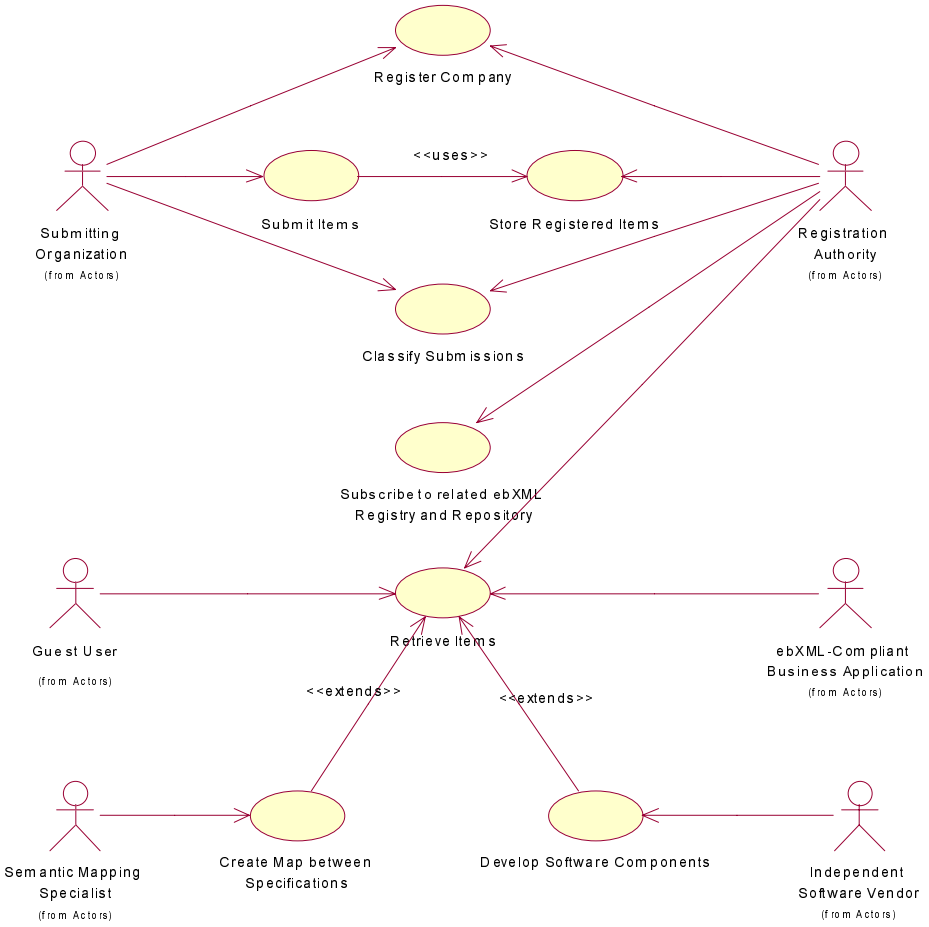

Figure 9.

Detailed discussion of each actor's viewpoint follows.

3.1 Submitting Organization

The Submitting Organization (SO) is responsible for the submission of items that are to be included into an ebXML Repository and referenced in an ebXML Registry. The SO must first register itself and be approved to submit items as noted by the Register Company use case. After the SO has been approved, appropriate access rights are granted that allow the SO to

maintain its submissions and their metadata based on the different status codes specified by *ebXML Registry and Repository Part 3:Analysis*.

The Submit Items use case provides the ability to submit a package. The Submit Items use case always "uses" the Store Registered Items use case to enter the submission into the repository. A record is created in the ebXML Registry (hence the term "register"), and a submission object is created in the repository as a container of the submission, which may be a collection of various documents, models, DTDs and other related information. Items within the collection may also be registered in the registry, at the discretion of the SO. The RA may at a later date create another collection, and relate that collection to the SO collection.

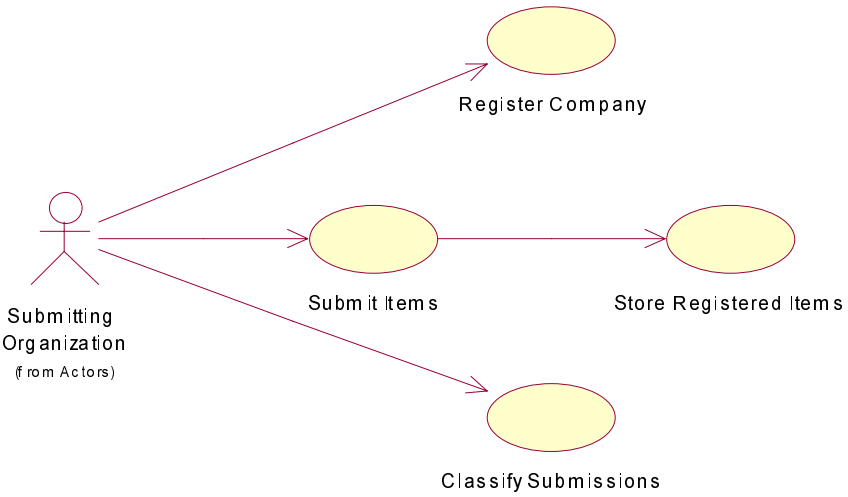

Figure 10

The SO can attempt to classify its submission, however, it is only a recommendation for the classification as the Registration Authority may change or add classification information after researching the submission.

3.2 Guest User

A Guest User is only granted read access. The Retrieve Items use case is abstract and includes the ability to directly retrieve data from the Repository or to search the ebXML Registry metadata in many ways, such as through unique identification, classification schemes, browsing, or complex search mechanisms. In the latter instance, a request may be sent from the Registry to the Repository to obtain a copy of the requested item.

Figure 11.

There is an implied trusted security relationship between the Registry and Repository that should be transparent to the Guest User. The Guest User may need to authenticate to the Registry.

3.3 Independent Software Vendor (ISV)

The Independent Software Vendor (ISV) actor uses the ebXML Registry and Repository to retrieve B2B specifications, in order to Develop Software Components. The Develop Software Components use case is extended by the Retrieve Items use case since at some point the specifications must be downloaded or viewed on-line. It is not required at all times (optional) throughout the software development life cycle, therefore the <<extends>> stereotype is used versus the <<uses>> stereotype.

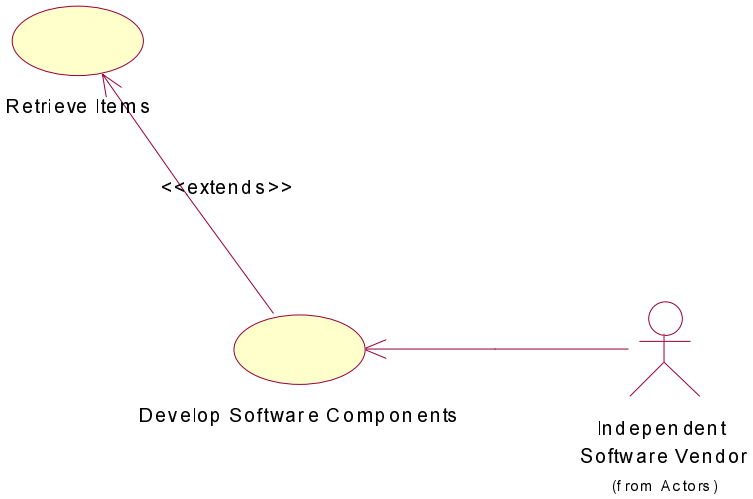

Figure 12.

Since ISV is a special type of Corporation, the software components that are constructed, tested and certified to be compliant to the retrieved specification(s), the ISV can, through its association to the SO, register and submit the software component for download by the Business User.

In addition, since the ISV is a special type of Guest User, the ISV may browse the registry and repository in a read-only mode to determine whether there are any good specifications that may be of interest. This allows the ISV to determine if the Develop Software Components use case should be entered.

The work product from the ISV could become a revenue source for both the ISV as well as the RA to cover administration costs.

3.4 Registration Authority (RA)

The Registration Authority (RA) is the owner of the ebXML registry and an ebXML Repository. The RA authorizes the SO's registration request that was initiated in the Register Company use case. This may include a complete background check of the SO in question.

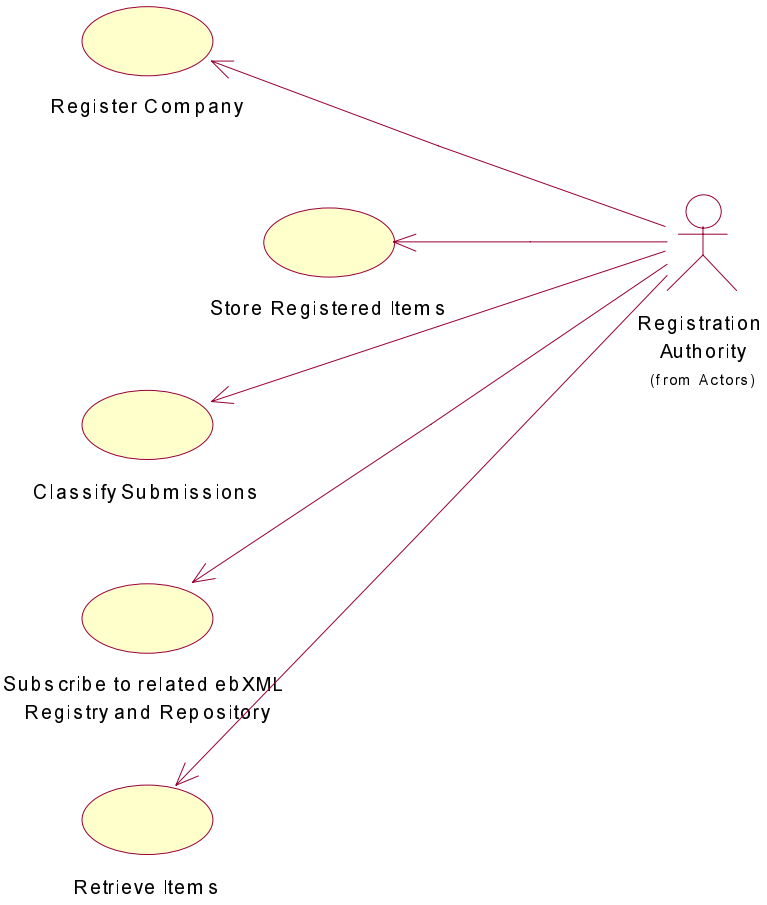

Figure 13.

The Store Items use case allows the RA the ability to grant public access to any items that the SO may have submitted. These items are in a queue until this grant occurs. As the trust between the SO and RA improves over time, the RA may certify the SO; i.e., allow the SO to Store Items without prior review by the RA. Factors that may go into certification criteria include the number of submissions, accuracy of the SO's classifications, and ability for an ISV to produce software components based on the SO's specification quality.

3.5 ebXML-compliant Business Application (ebAppl)

The ebXML-compliant Business Application imposes run-time requirements on the ebXML Registry and Repository, causing response time to become a key factor in the Retrieve Items use case. If the ebAppl is smart (software agent), the search and retrieval of items can be more complex, and collections of items could be returned. Therefore, collections of items are in scope for Retrieve Items use case.

Figure 14.

An example of an ebXML-compliant Business Application interacting with an ebXML Registry is to discover new trading partners that participate in a given marketplace, or provide certain products and services. This type of data is supported by the ebXML Business Process Metamodel, which is similar to eCo's Type Registry specifications. Conformance to eCo Types is out of scope for this specification, and is referenced only to suggest that the eCo's Type Registry can be implemented through an ebXML Registry.

3.6 Semantic Mapping Specialist

The usage of the ebXML Registry and Repository by the Semantic Mapping Specialist actor is very similar in scope to that of the ISV, except that the creation of maps includes the retrieval of two or more specifications. The interaction is the same in that the Create Map between Specifications use case is extended by the Retrieve Items use case. Specifications can be downloaded and mappings can be created off-line.

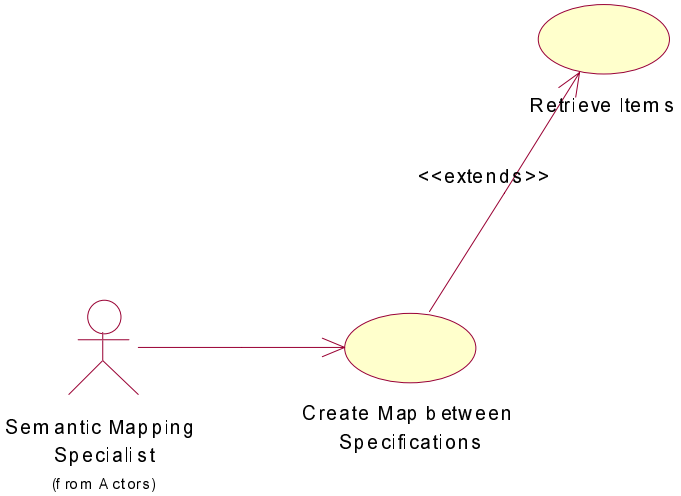

Figure 15.

The type of mappings could be "bridges" beyond two or more specifications, or integration mappings to commercial software in which the vendor does not provide an integration interface. Again, the Semantic Mapping Specialist is a special type of 1) Guest User for read-only search and retrieval usage, and 2) Corporation which can submit its work through its association to the SO.

The work product from the Semantic Mapping Specialist could become a revenue source for both the Semantic Mapping Specialist and the RA to cover administration costs.

4 ebXML Registry and Repository Architecture (Domain Package Diagram)

The following Domain Package Diagram details the various "subsystems" which carry out the responsibilities for the domain use cases. A specific use case may be carried out by a subsystem or an interaction between subsystems. The partitioning of the responsibilities for each use cases into packages or subsystems provides for an architecture that is intended to scale with size and increased volume of usage over time.

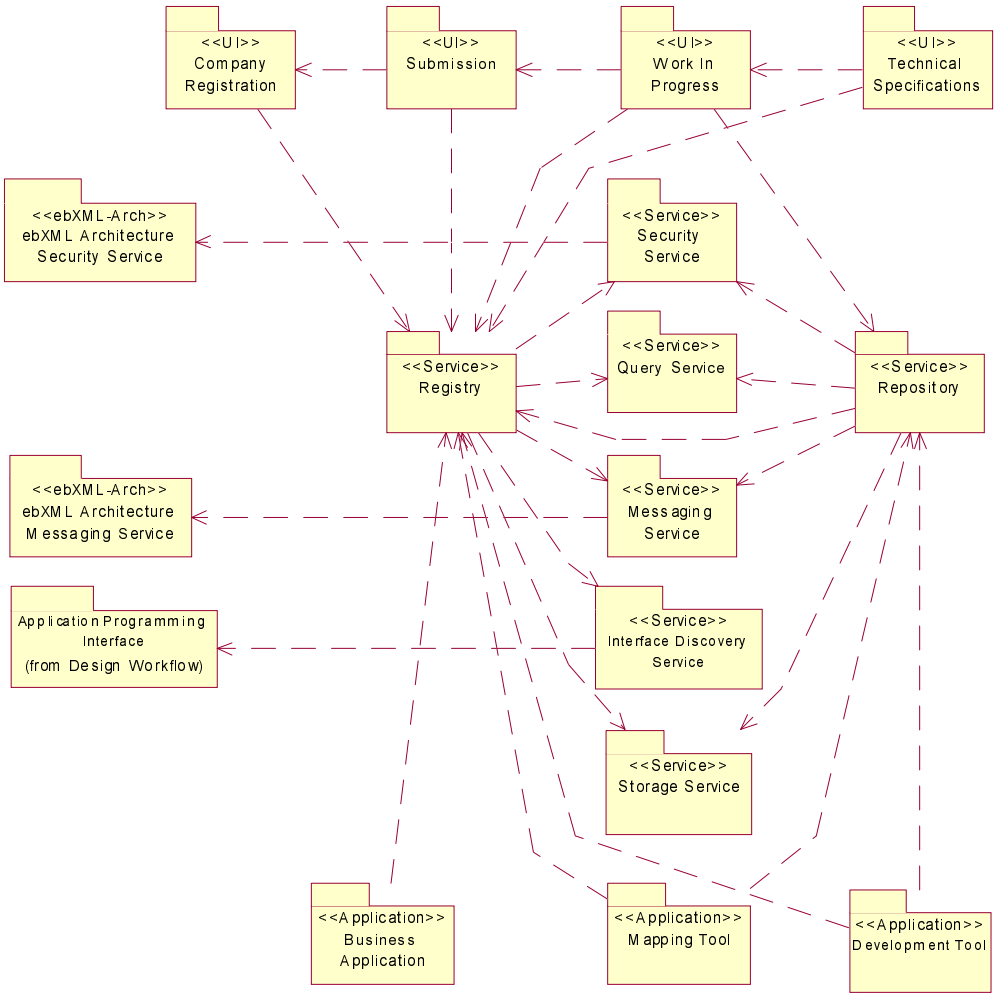

Figure 16.

It is IMPORTANT to note that there are optional <<Service>> packages **within** the Registry and Repository <<Service>> packages that cover additional capabilities that may be performed by the Repository. Each of these services will have its own API. These services are noted as optional or mandatory, and reflect whether a repository can be implemented by a file-based system, which is the minimum condition for implementation.

This document discusses the high level statement of the scope of each package and the interrelationships. The document *ebXML Registry and Repository Part 2: e-Business Requirements* provides more detail. Specifically, each package will have one use case diagram, and each use case will have a detailed activity diagram that describes the workflow.

4.1 Interrelationships between <<UI>>

The <<UI>> stereotype provides the scope for the User Interfaces and the workflow required for companies and individuals to interact with the registry and retrieve information from the repository. There are four main areas that comprise the registry and repository as shown in figure 17.

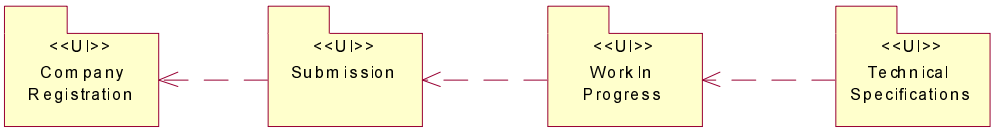

Figure 17.

This shows how to submit an item or package of items to a repository, allow people to be aware that it has been submitted, and if desired, allow the submission to progress into an ebXML-compliant specification. The Registration package is traceable to the Register Company use case. It will provide the use case diagram and detailed workflows for each use case in the diagram.

After an organization is registered with the RA and becomes an SO, it can submit items to the Registry and store them in the Repository. The Submission package is traceable to the Submit Items, Store Items, Classify Submissions, and Retrieve Items use cases. Unique classification schemes may be needed for each type of submission, since some submissions may have different important characteristics or attributes to describe them. Classifications will be detailed in *ebXML Registry and Repository Part 3: Analysis,* which will define the class diagrams for the registry.

If the SO intends for the submission to become an ebXML Specification to cover a business process, the submission may need to be enhanced. The Work In Progress package describes the workflow steps to become a Technical Specification, including library control functions such as checkin/checkout and versioning. Types of workflow steps may include enhancement of a submitted item to ensure that it meets the requirements of the ebXML Business Process Metamodel, harmonization with existing common business objects (Core Components) residing in the repository, and integration with other submitted items that may have overlapping semantics. Overlaps may occur for different vertical domains that have dependencies to each other. If a submission is not intended to be an ebXML Specification, but perhaps only a corporate specification, it still may reside in the repository but classified accordingly. In that case, there is no work to be performed, and the submission would pass rapidly through the Work In Progress package and flow into the Technical Specifications package.

The Technical Specifications package allows the submission to become a specification that can be viewed by **the public**. Work in progress is not available to the public unless in alpha or beta form, and classified as such. The Technical Specifications package provides the ability to browse the registry based on various classification schemes and to create simple and complex searches to find specifications.

4.2 Interrelationships between <<UI>> and <<Services>>

In order for the <<UI>> packages to complete each step of the workflow, each package must interact with the Registry package. The Work in Progress package may interface directly with the Repository package to add modified copies when a library control service is provided with the repository. Since the registry maintains state information about the original submitted item, the original copy will always be retained.

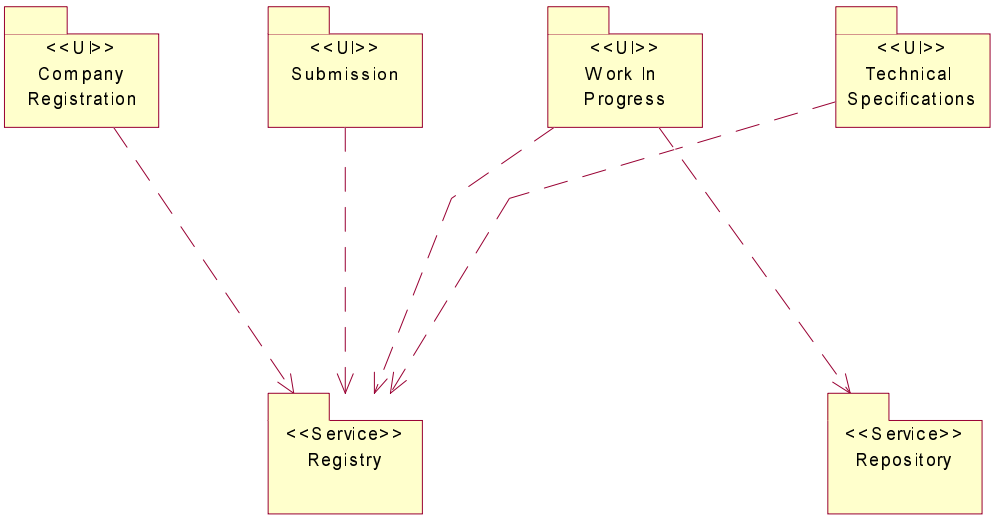

Figure 18.

All interactions via the <<UI>> are done through the Registry, and not through the Repository. This allows a more scaleable architecture, especially when some of the Registry contents are cached in memory.

4.3 Interrelationships between <<Applications>> and <<Services>>

The Registry and Repository also provide the ability for various development and business applications to interact with them.

As in typical software development life cycle, application development tools interact directly with the Repository. The two <<Application>> packages that are shown are the Mapping Tool, and the Development Tool. While they are both development tools *per se*, the Mapping Tools involves two or more specifications that need to be mapped at any one time. The Development Tool is a typical Integrated Development Environment including commercial Java, C++ and Smalltalk environments (VisualAge, Visual C++, Visual Café are examples). These IDEs require various features of the repository including check-out/check-in services, versioning services, transformation services for code generation as well as other features.

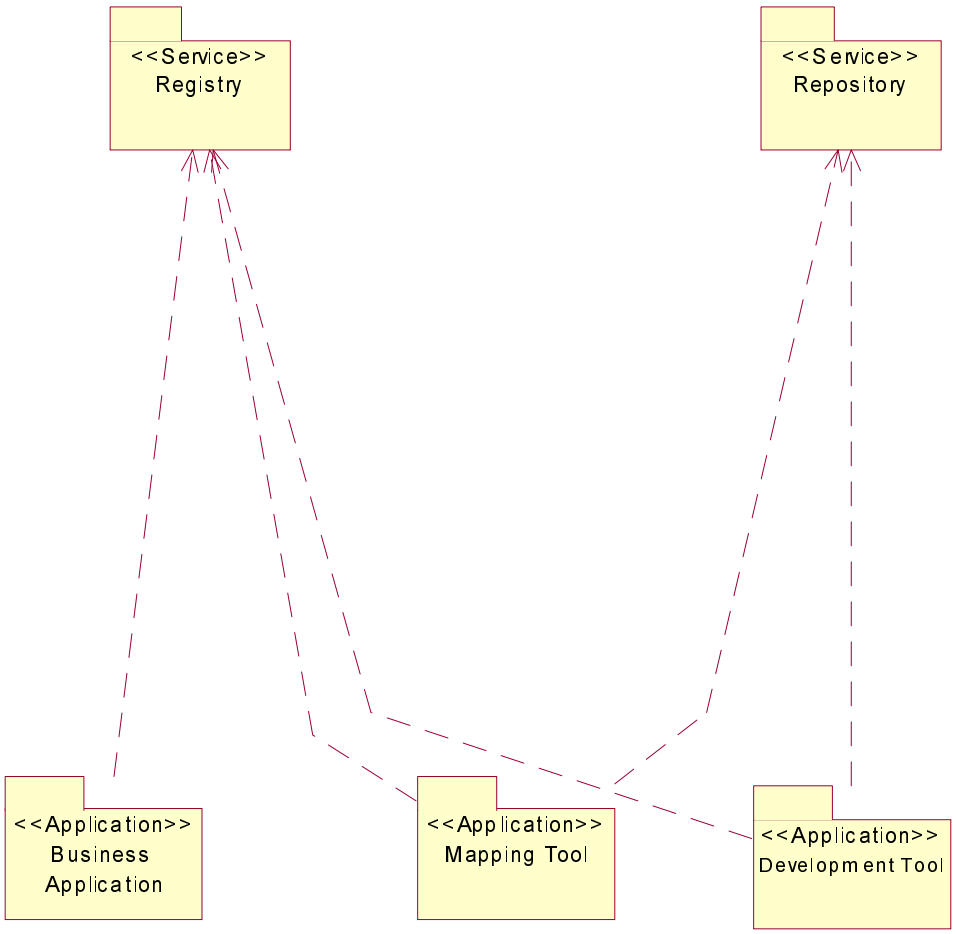

Figure 19.

The Business Application <<Application>> can only interface directly with the Registry, and has no need what for the repository services including check-out/check-in. This is traceable to the Retrieve Item use case in which the ebXML-compliant Business Application interacts with that use case.

4.4 Interrelationships between <<ebXML-Arch>> and <<Services>>

The Security and the Message services are related to the overall ebXML Architecture for security services and message services in conformance with the ebXML Transport, Package, and Routing specification. Thus the authentication, certificate, and encryption techniques are common across all ebXML-compliant Business Applications. Since the ebXML Registry and Repository is also a type of ebXML-compliant Business Application, intercommunication between the registry and repository is done using the ebXML TRP messaging service.

Figure 20.

4.5 Interrelationships between <<Services>>

As previously stated, the Registry and Repository both depend on a common security and messaging service, as formerly illustrated.

For the simple case of retrieval of a registered item by reference to a unique identifier over HTTP it is desirable to support "THTTP" as specified by RFC-2483 and revised by the OASIS registry and repository Technical Committee as well as TRP.

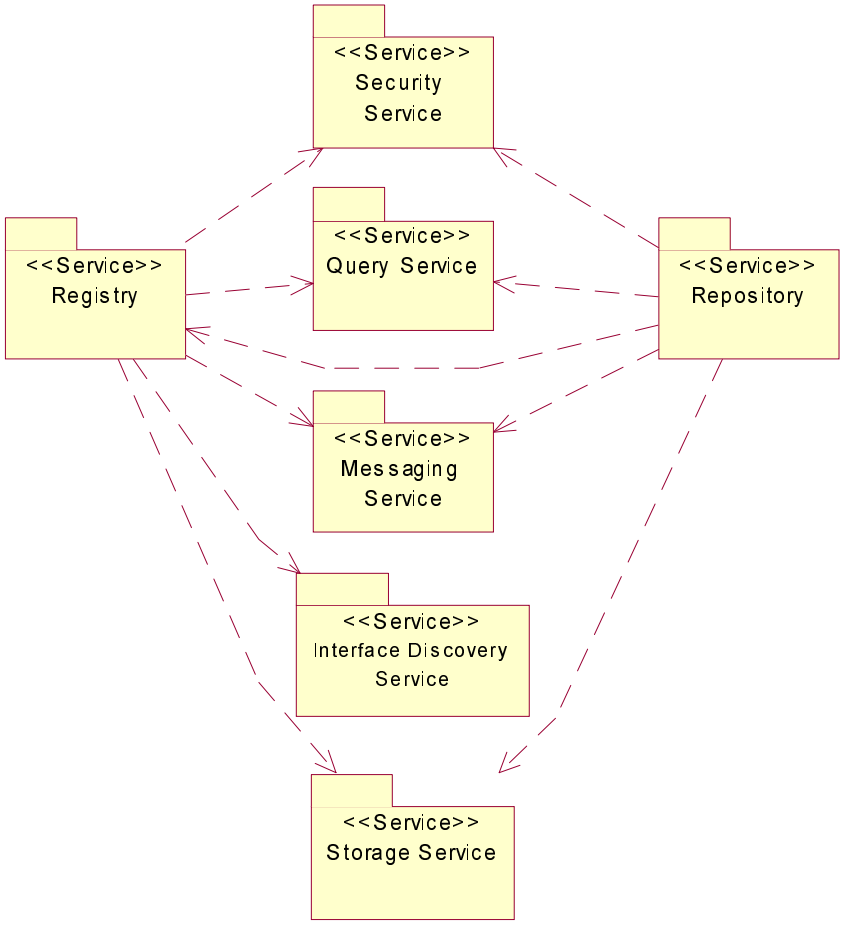

Figure 21

The dependencies between the Registry and Repository show a number of different features. First, the Registry's dependency on the Repository shows a publish and subscribe mechanism, in which the Registry receives metadata from a Repository particularly when a specification reaches a state in which the information needs to become public, or a new version is issued. The Registry may subscribe to more than one Repository besides its Primary Repository. This dependency is traceable to the Subscribe to Related Repository use case.

A Primary Repository is an ebXML-compliant Repository maintained by the RA to which the Registry has write authorization. This is shown by the Repository's dependency on the Registry to receive its content and establish a URI link to the content. This URI link must be maintained by the Registry.

The dependencies on the query service package show that metadata in the registry can be searched to find information regarding contents of the repository. This metadata describes the contents in the repository including its URI for the ability to retrieve a copy of the item. In addition, based on the RA's policies or types of information stored, the repository content could be searched. The types of searches our defined in *ebXML Registry and Repository Part 3:Analysis*.

The dependencies on the storage service shows that registered items in the repository must be stored. In addition, the classification schemes and metadata are stored in the registry. The physical storage mechanisms are not specified, but could be as simple as a file system.

Figure 22

The Interface Discovery service package describe a mandatory service that must be implemented by the RA. The RA chooses which optional services they will provide. Interface Discovery service is the mechanism that advertises which interfaces are available. Based on the security policies set up by the RA, these interfaces may be restricted.

The application programming interface will be defined in *ebXML Registry and Repository Part 4:Design*.

4.6 Optional Services within the <<Service>> Registry package

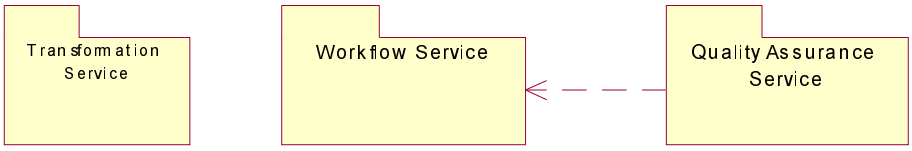

Figure 23.

4.6.1 <<Service>> Transformation Service

The Transformation Service is used to transform objects into another form. (e.g., IDEF-1X to XMI, XMI to XML Schema). *This is going to be defined by ebXML Registry and Repository Part 2:e-Business Requirements.*

4.6.2 <<Service>> Workflow Service

The Workflow Service carries out the workflows that are defined by *ebXML Registry and Repository Part 2:e-Business Requirements.*

4.6.3 <<Service>> Quality Assurance Service

The Quality Assurance Service is used to validate content based on its classification and is only invoked on certain workflow events; e.g., registration of ebXML business process model. *This is going to be defined by ebXML Registry and Repository Part 2:e-Business Requirements.*

4.7 Optional Services within the <<Service>> Repository package

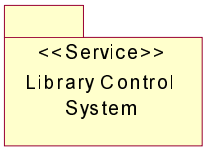

Figure 24

4.7.1 <<Service>> Library Control Service

The library control system package is an optional service that supports development, e.g., the work in progress package. *This is going to be defined by ebXML Registry and Repository Part 2:e-Business Requirements.*

Appendix

- OASIS Registry and Repository Technical Committee
- UN/CEFACT UML Profile and Methodology
- OMG Meta Object Facility (MOF)
- OMG Unified Modeling Language version 1.3 includes UML, OCL and XMI (located on Rational's web site)
- ISO 11179