FORM

IST-1999-10357

Engineering a Co-operative Inter-Enterprise Management Framework Supporting Dynamic Federated Organisations Management

Document Number:	IST-1999-10357/LMD/WP4/0522_AnnexE
Title of Deliverable:	Deliverable 11 Final Inter-Enterprise Management System Model **Annex E, BB Contract Schema Explained**
Deliverable Type: (P/R/L/I)*	P
Nature of the Deliverable: (P/R/S/T/O)**	R
Contractual Date of Delivery to the CEC:	28 February 2002
Actual Date of Delivery to the CEC:	28 February 2002

Workpackage responsible for the Deliverable:	WP3
Editor:	Brian Cullen, TCD and Henrik Røn, LMD
Contributor(s):	
Reviewer(s):	Stefan Penter, DLT and Vincent Wade, TCD

ABSTRACT

This document describes the XML Schema used for BB Contract Specification

KEYWORDS

XML, XSL, Building Block Contract specification, reuse

* Type: P:Public, R-Restricted, L-Limited, I-Internal

** Nature: P-Prototype, R-Report, S-Specification, T-Tool, O-Other

IST-1999-10357

FORM

Deliverable D11

Final Inter-Enterprise Management System Model

Annex E, BB Contract Schema Explained

Editor : Brian Cullen, TCD and Henrik Røn, LMD

Status – Version : Final

Date : **28/02/2002**

Distribution : Project

Code : IST-1999-10357/LMD/WP4/0522_AnnexE

The FORM Consortium consists of:

Atos Origin Intégration (France) – *Project Coordinator*

Broadcom Eireann Research Ltd. (Ireland)

DELTA Danish Electronics, Light & Acoustics (Denmark)

Fraunhofer FOKUS (Germany)

LM Ericsson A/S (Denmark)

TDC Tele Danmark A/S (Denmark)

Trinity College Dublin (Ireland)

UH Communications A/S (Denmark)

University College London (UK)

Waterford Institute of Technology (Ireland)

KPN Research (The Netherlands)

Table of Contents

1 Introduction

The purpose of this document is to explain the structure of the XML Schema [W3C XML Schema] [W3C XML Datatypes], produced by the Methodology Advisory Group (MAG), used to describe Building Block (BB) Contracts. This document is split into two sections. The first provides an explanation of the BB Contract Schema and how each of the elements defined within it should be used while the second details the guidelines that should be adhered to when bundling the contract descriptions and related documents (e.g. UML diagrams) together.

2 The Schema

This section of the document will introduce and explain the schema that will be used to describe the Building Block contracts. This section is itself broken down into two subsections. The first of these subsections presents an overview of the structure of a complete contract description while the second section discusses the individual tags in details.

2.1 Schema Overview

The next section will describe each of the elements of the schema in detail and it is, therefore, necessary to provide an overview of the entire schema to set the context for these descriptions. To this end the box below provides a summary version of a XML document using this schema. The purpose of this example is only to show where the elements fit together and as such attributes and character data are not shown.

```
<BBContract name="..." version="...">
        <description> ... </description>
         <supportedReferencePoints referenceArchitectureURI="…">
                <referencePointWithRole>
                        <referencePoint name="…"/> <role name="…"/>
                </referencePointWithRole>
        </supportedReferencePoints>
        <contractScope>
                <useCaseSet diagramURI="...">
                        <useCaseCollaboration diagramURI="..." relatedUseCase="..."/>
                </useCaseSet>
        </contractScope>
        <interfaceInteractions>
                 <interfaceInformation description="…" fileURI="…"/>
        </interfaceInteractions>
        <boundaryInformationModel>
              <informationModel modelURI=""/>
        </boundaryInformationModel>
        <technologyDescription>….</technologyDescription>
        <dependencies>
                <dependencyGroup>
                        <dependency BBcontract="..."/>
                        <collaborations> <collaboration diagramURI="..."/> </collaborations>
                </dependencyGroup>
        </dependencies>
</BBContract>
```

2.2 Schema Elements In Detail

This part of the document is structured so that each section represents a separate element or tag in the XML schema being described. It is hoped that this will allow contract designers to find information on the element they are currently working with quickly and efficiently. In effect this section is intended as a reference and would be difficult to read start-to-finish in one sitting.

Each subsection will discuss the element assigned to it under the following headings.

Heading	Contents
Overview	Provides a generic overview of the purpose of the element in question as well as an example of how it might look when used in a document.
Context	Specifies where in a document such an element should be found.
Information Representation	
→ *Attributes*	Explains the information represented by the attributes, if any, to the element.
→ *Content*	Explains the content model of the element in question.

2.2.1 The Root Element (BBContract)

2.2.1.1 Overview

As already mentioned this element is the root element of any XML document that wishes to completely describe a Building Block Contract Description. The purpose of this element is to provide the context in which the information relating to a contract can be expressed. This element also provides a mechanism for uniquely identifying BBContract, as will be discussed later in this section.

```
<BBContract
        name="…"
        date="…"
        contractSpecifier="…"
        version="…">
        …
</BBContract>
```

2.2.1.2 Context

This element should be the root element of any document describing a single BB Contract.

2.2.1.3 Information Representation

2.2.1.3.1 Attributes

Name	Type	Usage	Comments
name	anyURI	required	The name specified here is used to uniquely identify a particular contract.
contractSpecifier	String	optional	The name of the person or organisation that specified the contract.
date	Date	optional	The date the contract description was last updated to a degree that required a major version number change.
version	Decimal	required	The version of the contract description specified with a major and minor version number e.g. 1.1

As stated above the name used in specifying a contract must be globally unique to allow individual contracts to be identified. Guidelines for ensuring that all contracts are unique will be presented later in the document and readers are advised to ensure that they have these guidelines before using this element.

2.2.1.3.2 Content

The BBContract element only allows element content. Also it defines a specific order in which its children must appear and that each child can appear once at the most. The elements allowed are listed, in order, below. Note that the purpose of the tags are not discussed here but will be presented later in the document.

Element	Usage
description	required
supportedReferencePoints	required
contractScope	required
interfaceInteractions	required
boundaryInformationModel	required
technologyDescription	required
dependencies	optional

2.2.2 The description Element

2.2.2.1 Overview

The purpose of this element is to provide an overall description of the contract, it's purpose and other high level information that the contract designer feels merits inclusion. The usage of this element is illustrated below.

```
<description>
        …
        …
</description>
```

2.2.2.2 Context

This element should be a child of the BBContract element.

2.2.2.3 Information Representation

2.2.2.3.1 Attributes

This element contains no attributes.

2.2.2.3.2 Content

This element consists solely of free text, the format of which is at the discretion of the contract designer. However, as mentioned in the overview, it is expected that this text will include information on the purpose of the contract and any other high level information that would help a system designer decide whether or not the contract fits their needs.

2.2.3 The supportedReferencePoints Element

2.2.3.1 Overview

The purpose of this element is to provide a list of the reference points supported by the contract being described. The typical usage of this tag is shown below. On its own this tag provides no information but when taken with it's child elements, discussed later, it specifies the list of reference points that are supported by the contract in question.

```
<supportedReferencePoints
        referenceArchitectureURI="…">
        …
        …
</supportedReferencePoints>
```

2.2.3.2 Context

This element should be a child of the BBContract element.

2.2.3.3 Information Representation

2.2.3.3.1 Attributes

Name	Type	Usage	Comments
referenceArchitecture URI	anyURI	required	Uniquely specifies the reference point architecture from which the supported reference points are identified.

2.2.3.3.2 Content

This element allows only element content and in particular only one type of element, called referencePointWithRole. A contract may support zero or more referencePoints and thus this child element may be absent, if the contract does not explicitly support a reference point, but must appear once for each supported reference point.

2.2.4 The referencePointWithRole Element

2.2.4.1 Overview

The purpose of this element is to identify a single reference point with its role. The typical usage of this tag is shown below. In isolation this tag simply identifies a reference point and it is only when aggregated in the supportedReferencePoints element that it becomes useful.

```
<referencePointWithRole>
        …
        …
</referencePointWithRole>
```

2.2.4.2 Context

This element is a child of the supportedReferencePoints element.

2.2.4.3 Information Representation

2.2.4.3.1 Attributes

None

2.2.4.3.2 Content

This element has two child elements: referencePoint and role.

2.2.5 The referencePoint Element

2.2.5.1 Overview

The purpose of this element is to identify a single reference point with its role. The typical usage of this tag is shown below. In isolation this tag simply identifies a reference point and it is only when aggregated in the referencePointWithRole element that it becomes useful.

```
<referencePoint name="…"/>
```

2.2.5.2 Context

This element is a child of the referencePointWithRole element.

2.2.5.3 Information Representation

2.2.5.3.1 Attributes

Name	Type	Usage	Comments
name	ReferencePointID	Required	Uniquely specifies a particular reference point. The format of this name is specified as two or more uppercase letters, a hyphen and two or more uppercase letters e.g. IES-AS.

2.2.5.3.2 Content

This element has no content.

2.2.6 The role Element

2.2.6.1 Overview

The purpose of this element is to identify a single reference point with its role. The typical usage of this tag is shown below. In isolation this tag simply identifies a reference point and it is only when aggregated in the referencePointWithRole element that it becomes useful.

```
<role name="…"/>
```

2.2.6.2 Context

This element is a child of the referencePointWithRole element.

2.2.6.3 Information Representation

2.2.6.3.1 Attributes

Name	Type	Usage	Comments
name	roleID	Required	Uniquely specifies the role, which the contract acts as. The format of this name is a text string.

2.2.6.3.2 Content

This element has no content.

2.2.7 The contractScope Element

2.2.7.1 Overview

The purpose of this element is to provide a list of use case diagrams and their associated collaboration diagrams. The typical usage of this tag is shown below. As with the supportedReferencePoints element this tag provides no information on its own but, rather, is used to aggregate information that describes where the contract may be used.

```
<contractScope>
        …
</contractScope>
```

2.2.7.2 Context

This element should be a child of the BBContract element.

2.2.7.3 Information Representation

2.2.7.3.1 Attributes

This element contains no attributes.

2.2.7.3.2 Content

This element allows only element content and in particular only one type of element, called useCaseSet. This child element must appear at least once (i.e. contractScope cannot be empty) and may be repeated as often as desired.

2.2.8 The useCaseSet Element

2.2.8.1 Overview

The purpose of this element is to provide a reference to a UML use case diagram and a list of collaboration diagrams that elaborate one or more of the use cases within the diagram. The use cases summarised in the use case diagram cover the uses of the contract by one or more external entities, represented as use case actors. Each individual use case represents some use of the contract that provides benefit to one actor, called the Primary Actor for that use case. Each use case may also involve some interactions with actors other than the primary actor, these being called the Secondary Actors for that use case.

```
<useCaseSet diagramURI="…">
        …
</useCaseSet>
```

Design Note: One aim of allowing BBs to have multiple Contracts is that the different Contract types deal as much as possible with only one type of Contract user. Hence, a design goal is to have as many of the use cases in a contract share the same primary actor and reduce the interactions with secondary actors.

The typical usage of this tag is shown below.

2.2.8.2 Context

This element should be a child of the contractScope element.

2.2.8.3 Information Representation

2.2.8.3.1 Attributes

Name	Type	Usage	Comments
diagramURI	anyURI	required	Specifies a URI where a UML use case diagram can be retrieved/viewed. It is recommended that relative URIs be used to allow contracts to be exchanged easily.

2.2.8.3.2 Content

This element allows only element content and in particular only one type of element, called useCaseCollaboration. This child element must appear at least once (i.e. useCaseSet cannot be empty) and should appear once for each use case within the diagram.

2.2.9 The useCaseCollaboration Element

2.2.9.1 Overview

The purpose of this element is to provide a reference to an UML interaction diagram that elaborates on a named use case. An interaction diagram may either be a collaboration diagram or a sequence diagram. Each interaction diagram should contain at least the following:

- An object that represents the contract as a whole.
- An object that represents the primary actor of the referenced use case.

```
<useCaseCollaboration
        relatedUseCase="…"
        diagramURI="…"/>
```

In addition the interaction diagram may also include the following if the contract specifier considers showing them as important to the understanding of the contracts operation:

- Objects representing secondary actors from the referenced use case.
- Objects representing information object classes present in the Boundary Information Model.

2.2.9.2 Context

This element should be a child of the useCaseSet element.

2.2.9.3 Information Representation

2.2.9.3.1 Attributes

Name	Type	Usage	Comments
relatedUseCase	token	required	Specifies the use case that the collaboration diagram relates to. There is no way to automatically ensure that the reference use case actually exists and so it is up to the writer to ensure this is consistent.
diagramURI	anyURI	required	Specifies a URI where a UML collaboration diagram can be retrieved/viewed. It is recommended that relative URIs be used to allow contracts to be exchanged easily.

2.2.9.3.2 Content

This element allows no content.

2.2.10 The interfaceInteractions Element

2.2.10.1 Overview

The purpose of this element is to provide a description of the interface provided by the contract in question. The typical usage of this tag is shown below. This element provides no information on it's own but, rather, is used to group all relevant interface information under one tag.

```
<interfaceInteractions>
        …
</interfaceInteractions>
```

2.2.10.2 Context

This element should be a child of the BBContract element.

2.2.10.3 Information Representation

2.2.10.3.1 Attributes

This element contains no attributes.

2.2.10.3.2 Content

This element allows one type of element content, interfaceInformation, which must appear at least once, but may appear as many times as needed.

2.2.11 The interfaceInformation Element

2.2.11.1 Overview

The purpose of this element is to provide a technology specific description of some or all of a particular interface.

```
< interfaceInformation>
        description = "…"
        fileURI = "…" />
```

2.2.11.2 Context

This element should be a child of the interfaceInteractions element.

2.2.11.3 Information Representation

2.2.11.3.1 Attributes

Name	Type	Usage	Comments
description	String	required	Short textual description of what is contained within the referenced file.
fileURI	anyURI	required	A reference to a file or set of files that describe the interface.

2.2.11.3.2 Content

This element allows no content.

2.2.12 The boundaryInformationModel Element

2.2.12.1 Overview

The purpose of this element is to provide references to UML class diagrams that represent the information supported by the contract being described. The typical usage of this tag is shown below. As with the supportedReferencePoints element this tag provides no information on it's own but, rather, is used to aggregate information that describes the boundaryInformationModel.

```
<boundaryInformationModel>
        …
</boundaryInformationModel>
```

2.2.12.2 Context

This element should be a child of the BBContract element.

2.2.12.3 Information Represented

2.2.12.3.1 Attributes

This element contains no attributes.

2.2.12.3.2 Content

This element allows only element content and in particular only one type of element, called informationModel. This child element must appear at least once (i.e. boundaryInformationModel cannot be empty) and can appear as many times as desired.

2.2.13 The informationModel Element

2.2.13.1 Overview

The purpose of this element is to describe either part or the entire boundary information model supported by the contract. The element also provides the facility to provide additional documentation of the class diagrams if necessary. The typical usage of this tag is shown below.

```
<informationModel modelURI="…" />
```

2.2.13.2 Context

This element should be a child of the boundaryInformationModel element.

2.2.13.3 Information Represented

2.2.13.3.1 Attributes

Name	Type	Usage	Comments
modelURI	anyURI	required	Specifies a URI to a file describing the information model. It is recommended that relative URIs be used to allow contracts to be exchanged easily. Additional documentation may also be linked.

If a UML diagram is used to specify the information model, the following rules should be followed in generating the class diagrams that represent the Boundary Information Model

- The objects in the model represent only information that is passed over the Contract. They should not represent information internal to any software that provides an implementation of the Contract but which is not visible via the Contract.
- The model should contain class elements and associations between classes
- Classes may only possess a class name and attributes, classes may not possess methods.
- Class attributes may only be simple type or arrays of simple types. Currently the set of simple types is char, string, real, integer, Boolean, though this may be expanded in future (perhaps aligned with the DMTF CIM type set). Complex types must be broken down into classes. Enumerations can use a class of stereotype <<enumeration>> and contain simply a set of attribute names each of which represents one value of the enumeration.
- Class attributes can be given initial values, e.g. “foo : integer = 1”

- Classes may be inherited from other classes in the model.
- Associations should be named. The classes at each end of the association should be given a name indicating their role in the association. If no name is given, the name of the class is used to indicate their role in the association.
- The cardinality of each end of an association should be given. Where no cardinality is given it is assumed to be "one and only one". Other common cardinalities are: "0..1" meaning zero or one, "0..*" meaning zero or more and "1..*" meaning one or more. Of course others such as "2..*" or "3..6" are also permitted.
- Aggregation associations can be represented by the association symbol (a diamond). A black centred diamond represents strong aggregation meaning that the child objects cannot exist without the parent object. A white centred diamond represents weak aggregation, where the child objects can exist without the parent object.
- If required, associations can be linked to an association class (via a dotted line to the association arc). Typically this is done when the association is itself associated to another class or when additional attributes are required by the association. The name of the association then defaults to the name of the association class

If required logical constraints to part of the model may be attached to a class as a documentation object. Constraints are then expressed within the text of the documentation, either in English or using some constraint language such as the Object Constraint Language (which is part of UML) or perhaps policy languages.

2.2.13.3.2 Content

This element allows no content.

2.2.14 The technologyDescription Element

2.2.14.1 Overview

The purpose of this element is to provide a description of the technologies and standards that were used in the specification of the contract being described. The usage of this element is illustrated below.

```
<technologyDescription>
        …
</technologyDescription>
```

2.2.14.2 Context

This element should be a child of the BBContract element.

2.2.14.3 Information Represented

2.2.14.3.1 Attributes

This element contains no attributes.

2.2.14.3.2 Content

This element consists solely of free text, the format of which is at the discretion of the contract designer. However it is expected that this text will name the standards that were used in the construction of the contract and that any technology dependencies or assumptions will be clearly identified.

2.2.15 The dependencies Element

2.2.15.1 Overview

The purpose of this element is to provide examples of the type of associations or dependencies that the contract being described will have with other contracts. These dependencies are not meant to be prescriptive but rather illustrate the behaviour that this contract should exhibit in certain situations. The typical usage of this tag is shown below. This element provides no information on it's own but, rather, is used to group all relevant information under one tag.

```
<dependencies>
        …
</dependencies>
```

2.2.15.2 Context

This element should be a child of the BBContract element.

2.2.15.3 Information Representation

2.2.15.3.1 Attributes

This element contains no attributes.

2.2.15.3.2 Content

This element allows only element content and in particular only one type of element, called dependencyGroup. This child element must appear at least once (i.e. dependencies cannot be empty) and may be repeated as often as desired.

2.2.16 The dependencyGroup Element

2.2.16.1 Overview

The purpose of this element is to group associations and dependencies between particular contracts or types of contracts. Grouping can be achieved in a number of different ways e.g. a grouping may contain the associations needed to a sub-service provided by the contract. The typical usage of this tag is shown below. This element provides no information on its own but, rather, is used to group all related dependency information under one tag.

```
<dependencyGroup>
        …
</dependencyGroup>
```

2.2.16.2 Context

This element should be a child of the dependencies element.

2.2.16.3 Information Representation

2.2.16.3.1 Attributes

This element contains no attributes.

2.2.16.3.2 Content

This element allows only element content and, in particular, only two types of elements are allowed, called dependency and collaborations. The dependency child element must appear at least once (i.e. this element cannot be empty) and may be repeated as often as desired. The collaborations element, on the other hand, must appear once and only once.

2.2.17 The dependency Element

2.2.17.1 Overview

The purpose of this element is to name a single Building Block Contract with which the contract currently being described can have a dependency on or association with. The typical usage of this tag is shown below.

```
<dependency BBContractName=”…”/>
```

2.2.17.2 Context

This element should be a child of the dependencyGroup element.

2.2.17.3 Information Representation

2.2.17.3.1 Attributes

Name	Type	Usage	Comments
BBContractName	anyURI	required	Specifies the name of the contract with which there is an association or dependency.

2.2.17.3.2 Content

This element allows no content.

2.2.18 The collaborations Element

2.2.18.1 Overview

The purpose of this element is to provide a list of the collaboration diagrams that illustrate a certain set of dependencies or associations between contracts. The typical usage of this tag is shown below. On its own this tag provides no information but rather provides a point under which related collaboration diagrams can be grouped.

```
<collaborations>
        …
        …
</collaborations>
```

2.2.18.2 Context

This element should be a child of the dependencyGroup element.

2.2.18.3 Information Representation

2.2.18.3.1 Attributes

This element contains no attributes.

2.2.18.3.2 Content

This element allows only element content and in particular only one type of element, called collaboration. This child element must appear at least once (i.e. collaborations cannot be empty) and may be repeated as often as desired.

2.2.19 The collaboration Element

2.2.19.1 Overview

The purpose of this element is to provide a reference to a single collaboration diagram. The typical usage of this tag is shown below. It is only when aggregated in the collaborations element that they become useful.

```
<collaboration diagramURI="…"/>
```

2.2.19.2 Context

This element should be a child of the collaboration element.

2.2.19.3 Information Representation

2.2.19.3.1 Attributes

Name	Type	Usage	Comments
diagramURI	anyURI	required	Specifies a URI where a UML collaboration diagram can be retrieved/viewed. It is recommended that relative URIs be used to allow contracts to be exchanged easily.

2.2.19.3.2 Content

This element allows no content.

3 Packaging of Contracts

3.1 Overview

The first section of this document dealt with explaining the XML schema that is used to describe a contract. However the question of how to bundle the XML file(s) and the related files in a portable format still needs to be addressed. The purpose of this section is to the provide guidelines necessary to ensure that all contracts are packaged in a consistent format.

3.2 Contract Names

Contracts must have globally unique names and it is therefore necessary for clear guidelines to be provided for their names. For our purposes it has been proposed that a URI style approach be used. Therefore contract names will have the following format "<*domain name*>/[<*optional subclassifications*>/]<*ContractName*>, for example cs.tcd.ie/FORM/ServerMonitor.

3.3 File Formats

To be useful as a repository it would be best if all contributors used the same file format. It is recommended, for ease of viewing, that all references to files be to HTML files. In cases where diagrams are necessary these should be supplied in a standard graphic format, preferably JPEG and embedded in a HTML page.

3.4 URIs

The issue addressed here is the format of the URIs provided within the XML. As was stated in the previous section of the document all URIs should be relative i.e. locate the resource relative to where the XML file is located. This will allow the contracts and their associated files to be moved and distributed without having to maintain the URIs. As will be seen later in this section this will mean that for the most part URIs will simply name files as the files themselves will normally be located in the same directory as contract description.

There is one exception to the above guidelines and that is for the referenceArchitecture attribute of the supportedReferencePoints element. The URI specified by this attribute should identify the location of a document describing the reference architecture but as such a document is likely to be highly referenced it would be inefficient to insist that all contracts include their own copy. It is permissible, therefore, for this attribute to contain an absolute reference to this document as long as the named location is globally accessible, e.g. a complete web address. Please note that contract specifiers can still include this document with their specification in which case the original guidelines must be followed.

3.5 File Names

As is outlined in the next section each contract in a repository is kept in a separate directory and as such there cannot be a naming conflict between the files necessary for two distinct contract descriptions. Also note that it was decided that files cannot be shared between contracts. These two decisions mean that filenames can be left entirely up to the contract specifier. Nonetheless it was decided to provide some guidelines for file naming to achieve some level of consistency.

Each part of the schema that allows a reference to a file is identified below. For each of these parts a filename stub is identified that should be used to start the file name of the referenced HTML file and all its associated files.

3.5.1 BBContract\contractScope\useCaseDiagram – diagramURI

File Name Format: UseCase<*number*>

Field	Value
<*number*>	Each file should be numbered in order i.e. 1,2,3 etc.

3.5.2 BBContract\contractScope\useCaseDiagram\useCaseCollaboration – diagramURI

File Name Format: UseCaseCollaboration<*usecasenumber*>

Field	Value
<*usecasenumber*>	Number of the file containing the associated use case.

3.5.3 BBContract\interfaceInteraction\interfaceInformation – fileURI

File Name Format: InterfaceInformation<*number*>

Field	Value
<*number*>	Each file should be numbered in order i.e. 1,2,3 etc.

3.5.4 BBContract\boundaryInformationModel\InformationModel – modelURI

File Name Format: BoundaryInfoModel<*number*>

Field	Value
<*number*>	Each file should be numbered in order i.e. 1,2,3 etc.

3.5.5 BBContract\dependencies\dependencyGroup\collaborations\collaboration–diagramURI

File Name Format: DependencyCollaboration<*number*>

Field	Value
<*number*>	Each file should be numbered in order i.e. 1,2,3 etc.

3.6 The Distribution File

The first issue to be decided is the format of the distribution file. Here it was decided that a standard zip file would be the best choice for a number of reasons not least being its almost ubiquitous use and the wide availability of tools.

The directory structure within the zip file will closely match the URI style name provided for each contract. Therefore if we take the example of a contract called cs.tcd.ie/FORM/ServerMonitor then all the files relating to that contract can be located in the directory \cs.tcd.ie\FORM\ServerMonitor. Within each of these directories will be located a file called “Contract.xml” which will contain a single contract description in XML, that is an XML document with BBContract as the root element. All files referred to by the contract, i.e. the various UML diagrams, will also be stored in that directory.

Additionally each distribution file must contain a single XML file in the root directory called “ContractIndex.xml”. This XML file must conform to the provided schema and have a BBContractIndex as the root element of the document. This document is simply used to provide a list of all the contracts and their associated descriptions. Such an index file may also be placed in each directory that is an immediate parent of a directory holding a contract if desired. A XSLT [W3C XSL Transformation] document will be produced to extract this information from a Contract.xml file and their production can be made automatic.

Additional files may be in the various directories but should be kept to a minimum. In particular it is anticipated that companies producing such distributions may include copyright notices or similar files.

3.7 Amalgamation into Repositories

The overall aim when writing these guidelines was to ensure that distributions could be amalgamated to produce large contract repositories. Given the guidelines developed above the amalgamation of two separate distributions can be automated quite easily. Firstly it will be necessary to amalgamate the two directory structures excluding all contract indexes. Assuming that unique naming is adhered to then there should be no conflicts in this process. As stated previously the process of generating the contract indexes can and should be automated. The program performing this step will be required to use the provided XSLT document to extract the required information and then amalgamate the information extracted from numerous contracts into a single index file.

4 References

[W3C XML Schema] "XML Schema Part 1: Structures", W3C Recommendation 2 May 2001, http://www.w3.org/TR/xmlschema-1/

[W3C XML Datatypes] "XML Schema Part 2: Datatypes", W3C Recommendation 02 May 2001, http://www.w3.org/TR/xmlschema-2/

[W3C XSL Transformation] "XSL Transformations (XSLT) Version 1.0", W3C Recommendation 16 November 1999, http://www.w3.org/TR/xslt