Semantic Representations of the UN/CEFACT CCTS-based Electronic Business Document Artifacts

Version 0.1

Draft OASIS Profile, May 4, 2009

Document identifier:
20090504SemanticRepresentationOfDocumentArtifacts.pdf

Specification URIs:
This Version:
Previous Version: [N/A]
Latest Version:

Document identifier:
20090504SemanticRepresentationOfDocumentArtifacts.pdf

Location:
Establish

Editors:

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asuman Dogac</td>
<td>Middle East Technical University, Software R&amp;D Center, Turkey</td>
</tr>
<tr>
<td>Yildiray Kabak</td>
<td>Software Research and Development and Consultancy Ltd., Turkey</td>
</tr>
<tr>
<td>#</td>
<td>#</td>
</tr>
</tbody>
</table>
Abstract:
The purpose of this SET TC deliverable is to provide standard semantic representations of electronic document artifacts based on UN/CEFACT Core Component Technical Specification (CCTS) and hence to facilitate the development of tools to support semantic interoperability. The basic idea is to explicate the semantic information that is already given both in the CCTS and the CCTS based document standards in a standard way to make this information available for automated document interoperability tool support.

UN/CEFACT CCTS specifies the semantics of document artifacts in several dimensions: through the Core Components Data Types; through the structure of the core components; the semantics implied by the naming convention used; the semantics implied by the context, the Business Information Entities and the code lists. However, currently this semantics is available only through text-based search mechanisms.

In order to help with the interoperability of the document artifacts, we explicate the CCTS based business document semantics. By ‘explicating”, we mean to define their semantic properties through a formal, machine processable language as an ontology and the Web Ontology Language (OWL) is used for this purpose1.

1 Note that in defining the semantic properties of document artifacts, we kept the “context” semantics at an absolute minimum since UN/CEFACT UCM is working on this subject.
The semantics is explicated at two levels: At the first level, an upper ontology describing the CCTS document content model is specified. Furthermore, at this level, the upper ontologies for the prominent CCTS based standards, namely, GS1 XML, OAGIS 9.1 and UBL are also developed. The various equivalence relationships between the classes of the CCTS upper ontology and the CCTS based document standard ontologies are defined. These relationships are later used to find the similarities among the document artifacts from different document schemas.

At the next level, the semantics of the document schemas in each standard are described based on its upper ontology. The difference between the document schema specific ontology and the upper ontology is that the upper ontology describes the generic entities in a document content model whereas document schema ontologies describe the actual document artifacts as the subclasses of the classes in the upper ontology.

Furthermore, we explicate some semantics related with the different usages of document data types in different document schemas to obtain some desired interpretations by means of such informal semantics. The intention is to give the reasoner the same information that the humans use in transforming document schemas into one another.

When these ontologies are harmonized using a DL reasoner, the computed inferred ontologies reveal the implicit equivalences and subsumption relationships between the document artifacts. In other words, the shared semantic properties of the CCTS based document artifacts together with the implicit relationships inferred, help to identify their similarities. As expected, the Harmonized Ontology is effective only to discover equivalence of both semantically and structurally similar document artifacts. Yet different document standards use core components in different structures. Semantic properties of document artifacts are not enough to find the similarity of the structurally different but semantically equivalent document artifacts; possible differences in structures must be provided through heuristics to enhance the practical uses of the specified semantics. This heuristics is about possible ways of organizing core components into compound artifacts and is given in terms of predicate logic rules.

Note that a DL reasoner by itself cannot process predicate logic rules and we resort to a well accepted practice of using a rule engine to execute the more generic rules and carry the results back to the DL reasoner through wrappers developed. The results involve declaring further class equivalences in the ontology.

Finally, the similarities discovered among the document artifacts are then used to automate the mapping process by generating the XSLT rules.

The SET Harmonized Ontology contains about 4758 Named OWL Classes and 16122 Restriction Definitions conforming to the specification described in this document consisting of the following:

- All of the CCs/BIEs in UN/CEFACT CCL 07B.
- All of the BIEs in the common library of UBL 2.0.
- All of the common library of GS1 XML.
- OAGIS 9.1 Common Components and Fields
- The Harmonized Ontology expresses the relationships among the document artifacts of UN/CEFACT CCL, UBL 2.0, OAGIS 9.1 and GS1 XML according to SET specifications.

- The SET Harmonized Ontology is publicly available from http://www.srdc.metu.edu.tr/iSURF/OASIS-SET-TC/ontology/HarmonizedOntology.owl

Related with performance, an issue that needs to be addressed is whether the gain in automation justifies the resources needed to develop the ontological representation of the document schemas. In order to reduce this cost, we provide the SET XSD-OWL Convertor tool to create OWL definitions of the document schemas. This component converts a CCTS based document schema into OASIS SET TC OWL Definition and is publicly available from http://www.srdc.metu.edu.tr/iSURF/OASIS-SET-TC/tools/OASISSET.zip.

Note that, by conforming to a standard ontological representation and hence having all the document schema ontologies in a common pool, the users of the Harmonized Ontology only need to create a document schema ontology if it is not already in the Harmonized Ontology and benefit from all the existing connections when they do so.

Another issue related with performance is the computational complexity of the reasoning process involved. On a PC with 2GB RAM, the Racer Pro 1.9.2 Beta reasoner\(^2\) takes about 120 seconds to compute the Harmonized Ontology. Considering that the Harmonized Ontology will be re-computed only when a new document schema or a new CCTS based upper document ontology is introduced to the system, this performance is quite acceptable.

This work will be discussed to be further enhanced in the SET TC and technical support will be provided to the SET TC Members who develop their own use cases using the Harmonized Ontology. The SET XSD-OWL Converter tool can be used to generate the OWL definitions of their own document artifacts. The aim is to demonstrate the feasibility and practicability of the specifications to encourage industry take up.

**Status:**

This document is an OASIS Semantic Support for Electronic Business Document Interoperability (SET) TC Working Draft Profile and the work by the Editors is realized within the scope of the ICT 213031 iSURF Project (http://www.iSURFProject.eu) sponsored by the European Commission, DG Enterprise Networking Unit (http://cordis.europa.eu/ist/ict-ent-net/index.html).

Committee members should send comments on this specification to the set@lists.oasis-open.org list. Others should subscribe to and send comments to the set-comment@lists.oasis-open.org list. To subscribe, send a blank email message to set-comment-subscribe@lists.oasis-open.org. Once you confirm your subscription, you may post messages at any time.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the OASIS SET TC web page (http://www.oasis-open.org/committees/set/).

\(^2\) SET TC Members will receive a password to use Racer Pro for free for three months.
# Table of Contents

1. Introduction ..........................................................................................................................................8
   1.1 Terminology .........................................................................................................................................9

2. Enabling Technologies and Standards (Informative) ..........................................................................10
   2.1 A Brief Introduction to UN/CEFACT CCTS ..........................................................................................10
   2.2 A Brief Introduction to Web Ontology Language (OWL) .................................................................10
       2.2.1 OWL Lite Constructs .....................................................................................................................11
           RDF Schema Features ..........................................................................................................................11
           (In)Equality ........................................................................................................................................11
           Property Characteristics ......................................................................................................................11
           Property Restrictions ..........................................................................................................................11
           Restricted Cardinality ........................................................................................................................12
           Class Intersection ................................................................................................................................12
           Versioning ...........................................................................................................................................12
           Annotation Properties ........................................................................................................................12
           Datatypes ............................................................................................................................................12
       2.2.2 OWL DL Constructs .......................................................................................................................12
           Class Axioms ......................................................................................................................................12
           Boolean Combinations of Class Expressions .....................................................................................12
           Arbitrary Cardinality ..........................................................................................................................13
           Filler Information ..............................................................................................................................13
   2.3 A Brief Introduction to SPARQL ........................................................................................................13

3. The Problem Addressed ......................................................................................................................14

4. The SET Framework (Informative) ....................................................................................................16
5  Semantics implied by the CCTS Framework (Informative) .................................................................18
5.1 Core Component Data Type (CCT) Semantics ........................................................................18
5.2 Core Component Context Semantics ......................................................................................18
5.3 The Semantics exposed by the Use of the Code Lists ..........................................................18
5.4 Core Component Structure and Naming Semantics ..............................................................18
6  Specification of the Semantics Exposed by the CCTS Framework through Web Ontology Language (Normative) .................................................................................................................................20
6.1 Explicating Semantics through Core Component Types (CCT) and Data Types (DT) 20
6.2 Explicating Semantics through Context ..............................................................................21
6.3 Explicating Semantics through Code Lists ...........................................................................22
6.4 Explicating Semantics of Core Components .........................................................................23
6.5 Explicating Semantics of Business Information Entities (BIEs) .............................................25
6.6 The Overall Upper Ontology for the CCTS Framework .......................................................27
6.7 Explicating the Semantics of CCL Artifacts .........................................................................28
7  Explicating Semantics of CCTS based Document Standards – GS1 Upper Ontology (Normative)....31
7.1 Explicating the Semantics of GS1 Document Schemas ..........................................................32
8  Explicating Semantics of CCTS based Document Standards – UBL Upper Ontology (Normative)......35
8.1 Explicating the Semantics of UBL Document Schemas ........................................................36
9  Explicating Semantics of CCTS based Document Standards – OAGIS 9.1 Upper Ontology (Normative) 39
9.1 Explicating the Semantics of OAGIS 9.1 Document Schemas ..................................................40
9.2 An Overview of SET Upper Ontologies .............................................................................45
9.3 An Overview of SET Upper Ontologies and Document Schema Ontologies ....................46
10 Explicating Semantics Related with Different Usages of Document Artifacts in Different Standards (Informative) ..............................................................................................................................47
10.1 Explicating the Semantics on the Different Usages of CCTS Data Types ..............................47
11 Harmonizing the Ontologies of the Document Standards (Informative) ......................................48
12 Document Component Discovery Support (Informative) ................................................................. 53

12.1 SPARQL Queries .......................................................................................................................... 54

12.2 Queries that Require Reasoning Support ................................................................................. 55

13 Providing Heuristics to Discover Structurally Different Document Artifacts (Informative) .......... 58

13.1 A Heuristic to Help Finding the Equivalent BBIEs at Different Structural Levels ............... 58

13.2 Addressing Further Structural Differences in Document Artifacts ....................................... 59

13.2.1 Heuristics to Discover Structurally Different BBIEs .......................................................... 59

13.2.2 Heuristics to Discover Structurally Different ASBIEs ...................................................... 60

13.2.3 Heuristics to Discover Structurally Different ASBIE-BBIE Pairs ....................................... 61

13.2.4 Heuristics to Discover Structurally Different ABIEs ....................................................... 61

13.2.5 Further Heuristics ............................................................................................................... 63

13.2.6 An Example Tracing the Use of the Harmonized Ontology and the Provided Heuristics 63

14 How Does SET TC Specifications Support Automated XSLT Generation? .......................... 67

14.1 An Example: Translating UBL “Address.Details” to GS1 “Name and Address” ................. 67

14.1.1 Obtaining the XPath expressions for UBL "Address" ABIE and for its BBIEs/ASBIEs automatically .............................................................................................................. 67

14.1.2 Obtaining XPath expressions for GS1 "NameAndAddress" ABIE and for its BBIEs .......... 69

14.1.3 Constructing the XSLT Definitions ....................................................................................... 71

15 The Overall SET Framework (Informative) ............................................................................. 76

16 Performance of the System (Informative) ................................................................................. 79


1 Introduction

Today, an enterprise's competitiveness to a large extent is determined by its ability to seamlessly interoperate with others. Recognizing this need, the European Commission’s Enterprise Networking Unit defined the Interoperability Service Utility (ISU) as a utility-like capability\(^3\). The iSURF Project\(^4\) is realizing ISU services that facilitate real-time information sharing and collaboration between enterprises by providing semantic support for electronic business document interoperability.

Business Document interoperability initiatives started in the 1970s before the invention of the Internet. The first standard developed was the Electronic Data Interchange (EDI). Starting with the late 1990s eXtensible Markup Language (XML) became popular for describing data exchanged on the Internet. The relative human readability and the amount of XML tools available made XML a popular basis for a number of new document standards such as Common Business Library (CBL) and Commerce XML.

The earlier standards have focused on static message/document definitions which were inflexible to adapt to different requirements that arise according to a given context which could be a vertical industry, a country or a specific business process. The leading effort for defining business document semantics came from the UN/CEFACT Core Components Technical Specification\(^5\) (CCTS) in the early 2000s. UN/CEFACT CCTS provides a methodology to identify a set of reusable building blocks, called Core Components to create electronic documents.

CCTS is gaining widespread adoption by both the horizontal and the vertical standard groups. Universal Business Language\(^6\) (UBL) was the first implementation of the CCTS methodology in XML. Some earlier horizontal standards such as Global Standard One (GS1) XML\(^7\) and Open Applications Group Integration Specification\(^8\) (OAGIS®) have also taken up CCTS.

However, the CCTS based standards, although they share some common semantics inherited from CCTS, are not interoperable as detailed in\(^9\). There is a need to expose their common semantics in a standard way to facilitate the development of tools to support their interoperability.

This document attempts to deliver a framework and a specification for expressing the semantics of some of the CCTS based standards, namely, UN/CEFACT Core Component Library\(^10\) (CCL), UBL, OAGIS® 9.1\(^11\) and GS1 XML. The upper ontologies for UN/CEFACT/CCTS (Core Components and Business Information Entities), UBL, GS1 and OAGIS® are specified to describe the document content models for each of the standards.

Furthermore for some chosen document schemas from each of the document standard, the semantic descriptions in the form of ontologies are given. Through a reasoner, a Harmonized Ontology is computed. The Harmonized Ontology reveals the implicit relationships between the document artifacts defined by different electronic business document standards. Query templates using the Harmonized Ontology are formulated to facilitate the discovery and reuse of document components. Furthermore, since the Harmonized Ontology shows the correspondences among document artifacts, how this can be

\(^4\) http://www.iSURFProject.eu
\(^7\) Global Standard One XML, http://www.gs1.org/productsolutions/ecom/xml/
\(^8\) OAGIS®, http://www.openapplications.org/global/intro.htm (Copyright (c) Open Applications Group. All Rights Reserved)
\(^10\) UN/CEFACT - Core Component Library (CCL), http://www.unece.org/cefact/codesfortrade/uncl/CCL07B.zip
used to automate the generation of XSLT\textsuperscript{12} rules to map between electronic business document standards is demonstrated.

### 1.1 Terminology

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in IETF RFC 2119 [RFC211].

\textsuperscript{12} http://www.w3.org/TR/xslt.html
2 Enabling Technologies and Standards (Informative)

This section briefly describes some of the enabling standards and technologies. Further information is available from the references.

2.1 A Brief Introduction to UN/CEFACT CCTS

UN/CEFACT CCTS (Core Component Technical Specification) defines a framework on how to assemble documents from core components and provides rules for naming, structuring and reusing core components. A Core Component is designed to be context-independent so that it can later be adapted to different contexts and reused. When a Core Component is restricted to be used in a specific business context, it becomes a Business Information Entity (BIE) and given its own unique name. CCTS uses ISO 11179 Part 5 naming convention for the CCs and BIEs. Eight categories has been defined for the business context and specific code lists and classification schemas are suggested for each category. The business context categories are: Business Process Context; Product Classification Context; Industry Classification Context; Geopolitical Context; Business Process Role Context; Supporting Role Context; System Capabilities Context and Official Constraints Context.

The aim of CCTS is to provide interoperability among electronic business documents by requiring all Business Information Entities (BIEs) to be related back to the common Core Components (CCs) and hence to share a common semantics.

The UN/CEFACT Core Component Library (UN/CCL) is the repository for the Core Components. It provides a repository for the Core Components to increase the reuse of data elements during modelling and improving enterprise interoperability by providing a common basis for business information description. UN/CEFACT envisions this library to grow and also change over time as users can either modify existing components or design and submit new Core Components in case the existing ones are not sufficient to fulfil the actual business requirements. Currently there 212 ASCCs, 96 ACCs, 636 BCCs, 1011 BBIEs, 337 ASBIEs and 184 ABIEs in the Core Component Library.

2.2 A Brief Introduction to Web Ontology Language (OWL)

Web Ontology Language (OWL) is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL builds upon the Resource Description Framework (RDF). The complementary RDF Vocabulary Description Language, RDF Schema standard describes how to use RDF to describe RDF vocabularies.

OWL provides three decreasingly expressive sublanguages:

- **OWL Full** is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. It is unlikely that any reasoning software will be able to support complete reasoning for OWL Full.

---

15 OWL Web Ontology Language, http://www.w3.org/2004/OWL/
16 Resource Description Framework (RDF), http://www.w3.org/RDF/
17 http://www.w3.org/TR/rdf-schema/
18 OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/
• **OWL DL** supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL is so named due to its correspondence with description logics which form the formal foundation of OWL.

• **OWL Lite** supports those users primarily needing a classification hierarchy and simple constraints.

Within the scope of this document, only OWL DL constructs are considered and in the rest of the document, “OWL” is used to mean “OWL DL” unless otherwise stated.

OWL describes the structure of a domain in terms of classes and properties.

The list of OWL language constructs is as follows:

### 2.2.1 OWL Lite Constructs

**RDF Schema Features**

- Class (Thing, Nothing)
- rdfs:subClassOf
- rdf:Property
- rdfs:subPropertyOf
- rdfs:domain
- rdfs:range
- Individual

**(In)Equality**

- equivalentClass
- equivalentProperty
- sameAs
- differentFrom
- AllDifferent
- distinctMembers

**Property Characteristics**

- ObjectProperty
- DatatypeProperty
- inverseOf
- TransitiveProperty
- SymmetricProperty
- FunctionalProperty
- InverseFunctionalProperty

**Property Restrictions**

- Restriction
- onProperty
- allValuesFrom
• someValuesFrom

Restricted Cardinality
• minCardinality (only 0 or 1)
• maxCardinality (only 0 or 1)
• cardinality (only 0 or 1)

Class Intersection
• intersectionOf

Versioning
• versionInfo
• priorVersion
• backwardCompatibleWith
• incompatibleWith
• DeprecatedClass
• DeprecatedProperty

Annotation Properties
• rdfs:label
• rdfs:comment
• rdfs:seeAlso
• rdfs:isDefinedBy
• AnnotationProperty
• OntologyProperty

Datatypes
• xsd datatypes

2.2.2 OWL DL Constructs

Class Axioms
• oneOf, dataRange
• disjointWith
• equivalentClass (applied to class expressions)
• rdfs:subClassOf (applied to class expressions)

Boolean Combinations of Class Expressions
• unionOf
• complementOf
• intersectionOf
2.3 A Brief Introduction to SPARQL

SPARQL\(^{19}\) is a query language for RDF graphs. It is similar to Structured Query Language (SQL) and queries are written against the triples of RDF graph. The SPARQL uses the RDF view of an OWL ontology. Therefore, it does not benefit from the semantic described in an OWL ontology very effectively. A recent work, called SPARQL-DL\(^{20}\), is initiated to enhance the expressive power of SPARQL for OWL-DL ontologies. In SPARQL-DL the queries are formalized against the class hierarchy of an OWL-DL ontology. The initiative is very new and as it becomes mature, the SPARQL queries might be migrated to SPARQL-DL.

---

\(^{19}\) SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/

3 The Problem Addressed

A survey and analysis of CCTS based standards are given based on their document design principles, the use of code lists and XML namespaces, how they use CCTS methodology and how they address standards handle extensibility and customization. This analysis reveals that there are considerable differences in the resulting document schemas hampering their interoperability. There are efforts for harmonization of these standards.

As a simple example to clarify this point, consider the schemas of two components describing “Address” in two different standards as shown in Figure 1: the OAGIS® 9.0 “AddressBaseType” Component and GS1 XML “NameAndAddressType” document element. It is clear that although both of these standards are based on UN/CEFACT CCTS Methodology, there are considerable differences in the resulting document fragments. Currently, the mechanism to transform such documents into one another is to use Extensible Stylesheet Language Transformations. However this process is both manual and very tedious because each mapping has to be defined explicitly by a human being.

Figure 1 An Example Comparing Related Parts of OAGIS® BOD 9.0 and GS1 XML Documents

Therefore although the electronic document standards developed so far proved to be very useful for industry and government applications, further efforts are needed for their harmonization and semantic

24 http://www.w3.org/TR/xslt
interoperability.
The ultimate aim of the SET TC can be visualized through Figure 2: Given two document schemas from two different document standards both based on CCTS, automatically find the equivalent document artifacts in these schemas and generate the XSL Transformations for the discovered equivalences again automatically. The result is to be presented to a human for further possible adjustments. Once these transformations are fixed at the schema level, the instance level transformations are automatic.

Figure 2 Semi-automatic Translation of Electronic Business Documents from different CCTS-based Standards

---

25 This figure is inspired from the following paper: Werher Stuhec, “How to Solve the Business Standards Dilemma The Context Driven Business Exchange”, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/a6c5dce6-0701-0010-45b9-f6ca8c0e6474
4 The SET Framework (Informative)

The semantics is expressed by using a machine processable ontology language and a vocabulary. Different representation techniques and/or vocabularies will make it difficult (if not impossible) to harmonize the semantics of document standards. CCTS based standards already made sharing some common semantics possible; can we express this in a standard way?

There is not a unique way of representing the semantics of document artifacts: for example, what one representation may choose as a subclass, may be represented as an object property in another representation. When this happens, harmonizing different ontological representations will become a challenge. Therefore, within the scope of the SET TC, we propose a standard semantic representation for the document artifacts so that all such semantics collected in a common pool. In this way, when the semantics of a new document schema is introduced to the pool, it benefits from the already existing semantic connections in the Harmonized Ontology.

We first describe the semantics of the CCTS Framework. We term this part as “explicating” semantics because what we specify is already in the CCTS Framework specification, albeit in a spreadsheet format. We explicate this semantics through a formal machine processable language as an ontology that will serve the purposes of discovery, reuse and translation.

The semantics of document standards are defined at two levels: An upper ontology describing the document content model for each standard considered is developed and then the semantics of each of the document schemas in that standard are described based on its upper ontology.

1. We start with the specification of an upper ontology that describes the CCTS document content model (Core Components and Business Information Entities).
2. UN/CEFACT Core Component Library26 (CCL) contains the document artifacts based on CCTS. An upper ontology is created to contain all the artifacts defined in CCL conforming to the CCTS ontology. In CCL, the CCs and the BIEs are provided in MS Excel spreadsheets. To generate an ontology from the CCL artifacts conforming to the specified CCTS upper ontology, the CCL spreadsheets are first converted to a custom XML format by using XML Map mechanism of MS Excel. Then, the necessary OWL classes are created automatically from this XML file using the software developed.
3. A new upper ontology is created for each of the UBL, GS1 and OAGIS® 9.1 standards separately. While creating these ontologies, their relations with the CCTS upper ontology classes are also established.
4. For a selected set of document schemas from each of the document standard, the semantic description in the form of an ontology is created automatically using the software developed for this purpose. The difference between the document schema specific ontology and the upper ontology is that the upper ontology describes the generic entities in the document content models whereas document schema ontologies describe the actual document artifacts as the subclasses of the classes in the upper ontology.
5. **The first 4 items are the specifications within the scope of the SET TC. The remaining work is not a specification but realized to demonstrate how this specification can be used.**
6. We explicate some semantics related with the different usages of document data types in different document schemas to obtain some desired interpretations by means of informal semantics. The intention is to give the reasoner the same information that the humans use in transforming document schemas into one another.
7. Through a Description Logics (DL) reasoner, a Harmonized Ontology is computed. The Harmonized Ontology gives the specified as well as the computed equality and subsumption relations among the classes of both the upper ontologies and the document schema ontologies. The Harmonized Ontology is useful for three purposes:
   a. It helps to discover equivalence of structurally similar document artifacts between two document schemas

26 UN/CEFACT - Core Component Library (CCL), http://www.unece.org/cefact/codesfortrade/unccl/CCL07B.zip
b. For translating such document artifacts through automatically generated XSLT rules.

c. Query templates (SPARQL and Reasoner based queries) are formulated to facilitate the
discovery and reuse of document components using the Harmonized Ontology.

8. The semantic properties of the CCTS based document artifacts help discovering the equivalences of
structurally similar and semantically equivalent elements. However different document standards use
core components in different structures. Semantic properties of document artifacts are not enough to
find the similarity of the structurally different but semantically equivalent document artifacts; possible
differences in structures must be provided as heuristics to enhance the practical uses of the specified
semantics. With the help of additional heuristics, it becomes possible:

a. To discover equivalency of structurally different but semantically equivalent document
   artifacts between two document schemas.

b. Query templates (SPARQL and Reasoner based queries) are formulated to facilitate the
discovery and reuse of structurally different but semantically equivalent document
   components.

c. At a later stage tools can be developed for translating structurally different document
   artifacts.

We then present the overall SET framework\textsuperscript{27} to put all the parts together.

Note that for defining heuristics to handle structurally different document artifacts, the Description Logic is
not sufficient but more general purpose Predicate Logic Rules are needed. If a DL reasoner like RacerPro
starts supporting a rule language like SWRL, this will be an ideal solution. However, for the time being,
we use RacerPro as the DL reasoner for performance reasons and use JESS Rule Engine to execute
more generic rules and carry the results back and forth through wrappers developed. Note also that this is
about the technology used to implement the specifications; it does not effect the specifications.

\textsuperscript{27} We use the following definition from wiki (http://en.wikipedia.org/wiki/Framework): A software framework is a
re-usable design for a software system (or subsystem). A software framework may include support programs, code
libraries, a scripting language, or other software to help develop and glue together the different components of a
software project. Various parts of the framework may be exposed through an API.
5 Semantics implied by the CCTS Framework (Informative)

In the CCTS approach, the semantics of electronic business documents are specified in several dimensions:

- Core Component Data Type semantics;
- The semantics exposed by the contexts in which BIE are used;
- The semantics exposed by the use of the code lists;
- The semantics implied by the structure and the naming of the Core Components;
- The semantics implied by the structure and naming of the Business Information Entities

5.1 Core Component Data Type (CCT) Semantics

UN/CEFACT CCTS defines 14 Core Component Data Types (CCTs). CCTs are used in Data Types. There are two types of Data Types: Qualified Data Types, which restrict CCTs, and Unqualified Data Types, which are equal to Core Component Types.

5.2 Core Component Context Semantics

A Core Component is designed to be context-independent so that it can later be adapted to different contexts and reused. When a Core Component is restricted to be used in a specific business context, it becomes a Business Information Entity (BIE) and given its own unique name. Eight context domains have been defined by UN/CEFACT. UN/CEFACT Techniques and Methodologies Group (TMG) is working on the Unified Context Methodology (UCM)\(^\text{28}\). This part of the work will benefit from UCM when the specification is completed.

5.3 The Semantics exposed by the Use of the Code Lists

The code lists used to convey the meaning of the values in the elements of the document artifacts.

5.4 Core Component Structure and Naming Semantics

The CCTS approach discloses the following semantics about the document parts through the way it structures the document components:

- A document part with a specific business meaning such as "Address" or "Purchase Order", is termed as an Aggregate Core Component (ACC).
- The properties of ACCs are termed as Basic Core Component (BCC) and are described by using a Data Type.
- The associations between ACCs are defined through Association Core Components.

Another source of semantics for CCTS comes from the naming conventions. When naming the CCs and BIEs, ISO 11179 Part 5\(^\text{29}\) is used. This naming convention has three major parts: Object Class, Property Term and Representation Term/Associated Object Class Term. For example, when the Basic Core Component "Invoice.TaxAmount.Amount" is expressed according to the CCTS naming convention,

\(^{28}\) Unified Context Methodology, http://75.43.29.149:8080/display/public/UCM++Unified+Context+Methodology

“Invoice” is the Object Class, “TaxAmount” is the Property Term and “Amount” is the Representation Term. ASCCs and ASBIEs have Associated Object Class Terms instead of Representation Terms.

Hence, the following semantics becomes readily available through the naming conventions:

- Each ACC has an “Object Class Term”.
- Each BCC has an “Object Class Term”, a “Property Term”, and a “Representation Term”.
- Each ASCC has an “Object Class Term”, “Associated Object Class Term”, and a “Property Term”.
6 Specification of the Semantics Exposed by the CCTS Framework through Web Ontology Language (Normative)

This section specifies how the existing semantics in the CCTS Framework document artifacts can be explicated by using OWL constructs so that this semantics can later be used in an automated manner to discover useful implicit relationships among the document artifacts of other CCTS based standards.

6.1 Explicating Semantics through Core Component Types (CCT) and Data Types (DT)

First lets provide some insight on why there is a need to explicate the data type semantics: UN/CEFACT CCTS defines 14 CCTs. When two document artifacts use the same CCT, this can be considered as a hint towards these artifacts meaning the same thing if their other semantic properties also match.

The Core Component Type semantics SHOULD be explicated through the “owl: CoreComponentType” class. UN/CEFACT CCTS defines 14 CCTs and for each of them, a corresponding class SHOULD be created and inserted as the subclass of “CoreComponentType” as shown in Listing 1.

```xml
<owl:Class rdf:ID="CoreComponentType" />
<owl:Class rdf:ID="Amount.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="BinaryObject.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="Code.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="Date.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="DateTime.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="Identifier.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="Indicator.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="Measure.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="Numeric.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="Quantity.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="Text.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
<owl:Class rdf:ID="Rate.Type">
  <rdfs:subClassOf rdf:resource="#CoreComponentType"/>
</owl:Class>
```
6.2 Explicating Semantics through Context

The context in which a document artifact is used gives it a certain semantics. Therefore if two document artifacts have related contexts and if their other semantic properties are related, this gives a hint on their possible equivalence.

There **SHOULD** be an OWL class for context concept, called “Context” as given in Listing 2.

```xml
<owl:Class rdf:ID="Context" />
```

**Listing 2 Representing “Context” as an OWL Class**

The context categories defined by UN/CEFACT **SHOULD** be defined as subclasses of “Context” class as given in Listing 3.

```xml
<owl:Class rdf:ID="BusinessProcessContext" >
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>

<owl:Class rdf:ID="GeopoliticalContext" >
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>

<owl:Class rdf:ID="IndustryContext" >
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>

<owl:Class rdf:ID="ProductContext" >
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>

<owl:Class rdf:ID="RoleContext" >
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>

<owl:Class rdf:ID="OfficialContext" >
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>

<owl:Class rdf:ID="SupportingRoleContext" >
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>

<owl:Class rdf:ID="SystemConstraintsContext" >
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>
```

**Listing 3 UN/CEFACT Context Categories as subclasses of Class “Context”**

For each “Context” subclass, such as “IndustryContext”, an ontology can be defined based on the
taxonomies or classifications already used by the industry\textsuperscript{30} such as Universal Standard Product and Service Specification\textsuperscript{31} (UNSPSC) or Standard International Trade Classification\textsuperscript{32} (SITC). However, UN/CEFACT Techniques and Methodologies Group (TMG) is working on the Unified Context Methodology (UCM)\textsuperscript{33}. This part of the work will benefit from UCM work.

### 6.3 Explicating Semantics through Code Lists

Insight: The code lists are important to identify the meaning of a BCC or BBIE. As an example, assume that two document standards name a BBIE differently. However, if BBIEs use the same code list or use code lists for the same purpose, there is a possibility that they are similar. It should be noted that code lists used for a BCC or BBIE can vary according to context. Therefore, the classification of code lists is also important. For the classification categories, the identified context categories can be used.

There SHOULD be an OWL class for “codelist” concept, called “CodeList” as given in Listing 4.

<table>
<thead>
<tr>
<th>Listing 4 OWL class “CodeList”</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;CodeList&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;BusinessProcessCodeList&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#CodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;GeopoliticalCodeList&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#CodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;IndustryCodeList&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#CodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;ProductCodeList&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#CodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;RoleCodeList&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#CodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;OfficialCodeList&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#CodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;SupportingRoleCodeList&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#CodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;SystemConstraintsCodeList&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#CodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
</tbody>
</table>

Once the “CodeList” subclasses are defined in this, the specific code lists in use SHOULD be defined as a subclass of the related context. Some examples are provided in Listing 5.

<table>
<thead>
<tr>
<th>Listing 5 OWL class examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;iso-ch.3166.1999&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#GeopoliticalCodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class rdf:ID=&quot;ntis-gov.naics.1997&quot; &gt;</code></td>
</tr>
<tr>
<td><code>&lt;rdfs:subClassOf rdf:resource=&quot;#IndustryCodeList&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
</tbody>
</table>


\textsuperscript{31} http://www.unspsc.org/

\textsuperscript{32} http://unstats.un.org/unsd/cr/registry/regcst.asp?Cl=14

\textsuperscript{33} http://75.43.29.149:8080/display/public/UCM+-+Unified+Context+Methodology
6.4 Explicating Semantics of Core Components

Insight: There are a number of terms giving meaning to the CCs. When such semantics is explicated in an ontology, it may help to find similarities in document artifacts from different document schemas. For example, if two document artifacts have the same “Object Class Term”, this may give a hint on their similarity.

A CCTS Core component SHOULD be expressed as an OWL class as given in Listing 6.

Listing 6 OWL CoreComponent Class Definition

The following OWL classes MUST be defined to represent these terms as given in Listing 7.

Listing 7 OWL “ObjectClassTerm” and “RepresentationTerm” Class Definitions

A CCTS Basic Core Component (BCC) SHOULD be defined as an OWL class to have the following object properties “hasDataType”, “hasObjectClassTerm”, “hasRepresentationTerm”, and “possibleCodeLists” as given in Listing 8.
ACCs MAY contain BCCs and ASCCs, and ACCs SHOULD only have Object Class Terms. A CCTS Aggregate Core Component SHOULD be defined as an OWL class to have the following object properties “contains” and “hasObjectClassTerm” as as given in Listing 9.

```
<owl:Class rdf:ID="AggregateCoreComponent" >
  <owl:equivalentClass>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#CoreComponent"/>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#contains"/>
        <owl:allValuesFrom>
          <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
              <owl:Class rdf:about="#BasicCoreComponent"/>
              <owl:Class rdf:about="#AssociationCoreComponent"/>
            </owl:intersectionOf>
          </owl:Class>
        </owl:allValuesFrom>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasObjectClassTerm"/>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#ObjectClassTerm"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:equivalentClass>
</owl:Class>
```

Listing 9 OWL "AggregateCoreComponent" (ACC) Class Definition

A CCTS Association Core Component SHOULD be defined as an OWL class to have the following object properties “refersTo”, “hasObjectClassTerm”, and “hasAssociatedObjectClassTerm” as as given in Listing 10.

```
<owl:Class rdf:ID="AssociationCoreComponent" >
  <owl:equivalentClass>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#CoreComponent"/>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#refersTo"/>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#AggregateCoreComponent"/>
        </owl:allValuesFrom>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasObjectClassTerm"/>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#ObjectClassTerm"/>
        </owl:allValuesFrom>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasAssociatedObjectClassTerm"/>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#ObjectClassTerm"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:equivalentClass>
</owl:Class>
```
Note that, we choose not to use “Property Term” since the usage of property terms show a great degree of variances and hence do not contribute to discovering similarities among document artifacts.

6.5 Explicating Semantics of Business Information Entities (BIEs)

Insight: The semantics of a BIE is given by the core component from which it is derived and the context it is constrained to.

A BIE SHOULD be expressed as an OWL class as given in Listing 11.

Listing 11 OWL "BusinessInformationEntity" (BIE) Class Definition

The “BasicBusinessInformationEntity” class is based on “BasicCoreComponentClass” and this SHOULD be expressed as given in Listing 12.

Listing 12 OWL “basedOn” Object Property Definition

A BIE is a CC used in a context and there SHOULD be an “owl:ObjectProperty” called "usedInContext". This object property MUST have “BusinessInformationEntity” class as its domain and “Context” class as its range as given in Listing 13.

Listing 13 OWL “usedInContext” Object Property Definition

Just like a “BasicCoreComponent” it is derived from, a BIE SHOULD have Object Properties for its data type, naming terms and possible code lists but there is no subclass relationship among them since a BIE, being a restriction of a Core Component to a context, may not inherit all the properties of that Core Component.

A CCTS “BasicBusinessInformationEntity (BIE)” SHOULD be defined as an OWL class to have the following object properties “hasDataType”, “hasObjectClassTerm”, “hasRepresentationTerm”, and “possibleCodeLists” as given in Listing 14.

Listing 14 OWL "BasicBusinessInformationEntity" Definition
Listing 14 OWL “BasicBusinessInformationEntity (BIE)” Class Definition

An “AssociationBusinessInformationEntity” (ASBIE) SHOULD be defined as an OWL class to have the following object properties "refersTo", "hasObjectClassTerm", "hasAssociatedObjectClassTerm", "hasPropertyTerm", and “possibleCodeLists” as given in Listing 15.

```xml
<owl:Class rdf:ID="AssociationBusinessInformationEntity" >
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#BusinessInformationEntity" />
        <owl:Restriction>
          <owl:onProperty rdf:resource="#basedOn"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#AssociationCoreComponent"/>
          </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#refersTo"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#AggregateBusinessInformationEntity"/>
          </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasObjectClassTerm"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#ObjectClassTerm"/>
          </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasAssociatedObjectClassTerm"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#ObjectClassTerm"/>
          </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasPropertyTerm"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#ObjectClassTerm"/>
          </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#possibleCodeLists"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#CodeList"/>
          </owl:allValuesFrom>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```
Listing 15 OWL “AssociationBusinessInformationEntity” (ASBIE) Class Definition

An “AggregateBusinessInformationEntity” (ABIE) SHOULD be defined as an OWL class to have the following object properties “contains”, “basedOn”, and “hasObjectClassTerm” as given in Listing 16.

Listing 16 OWL “AggregateBusinessInformationEntity” (ABIE) Class Definition

6.6 The Overall Upper Ontology for the CCTS Framework

As a result of these definitions, the overall semantics exposed by the CCTS framework as an upper ontology is as pictorially shown in Figure 3.
6.7 Explicating the Semantics of CCL Artifacts

Insight: To be able to determine the semantically similar document artifacts at the schema level, the semantics of each document schema is explicat ed conforming to its own upper ontology. The relationships among the document schema ontology classes are established through reasoning process by using the explicit relationships defined among the upper ontology classes.

The semantics of UN/CEFACT Core Component Library (CCL) artifacts are explicated conforming to the CCTS Upper Ontology defined. The generation of ontologies from the artifacts defined in CCL conforming to the CCTS upper ontology as follows: To create the ontology classes corresponding to the CCL artifacts which are given in MS Excel spreadsheets, the CCL spreadsheets are first converted to a custom XML format by using XML Map mechanism of MS Excel. Then, through a piece of software developed, the necessary OWL classes conforming to the specified CCTS upper ontology are created from this XML file.

Listing 17 provides an example on how "Structured_Address.Details" artifact of CCL is represented in the Harmonized Ontology conforming to the CCTS Upper Ontology.

```xml
<owl:Class rdf:ID="Structured_Address.Details">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#AggregateBusinessInformationEntity"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasObjectClassTerm"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#Address"/>
          </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#usedInContext"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#Context"/>
          </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#contains"/>
          <owl:allValuesFrom>
            <owl:Class>
              <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Structured_Address.Identification.Identifier"/>
              </owl:Class>
            </owl:allValuesFrom>
          </owl:Restriction>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```
Listing 17  CCL "Structured_Address.Details" Class conforming to the CCTS Upper Ontology

Listing 18 provides an example on how "Structured_Address.Identification.Identifier" artifact of CCL is represented in the Harmonized Ontology conforming to the CCTS Upper Ontology.
Listing 18 CCL "Structured_Address.Identification.Identifier" Class conforming to the CCTS Upper Ontology
# 7 Explicating Semantics of CCTS based Document Standards – GS1 Upper Ontology (Normative)

Figure 4 shows the upper ontology for the GS1 XML document standard and as shown in this figure GS1 classes are related with the corresponding CCTS classes by using the “owl:equivalentClass” property.

Figure 4 The Upper Ontology for the Semantics Exposed by the GS1 XML Document Standard

A GS1 BBIE SHOULD be defined as an OWL class named “GS1.XML.BBIE” and it SHOULD be declared equivalent to the BBIE class defined in CCTS upper ontology as given in Listing 19.

Listing 19 OWL GS1 BBIE Class Definition

A GS1 ABIE SHOULD be defined as an OWL class named “GS1.XML.ABIE” and it SHOULD be declared equivalent to the ABIE class defined in CCTS upper ontology. The “contains” Object Property of the “GS1.XML.ABIE” class should be restricted to “GS1.XML.BBIE” and “GS1.XML.ASBIE” as given in Listing 20.

Listing 20 OWL GS1 ABIE Class Definition
A GS1 ASBIE SHOULD be defined as an OWL class named “GS1.XML.ASBIE” and it SHOULD be declared equivalent to the ASBIE class defined in CCTS upper ontology. The “refersTo” Object Property of the “GS1.XML.ASBIE” class should be restricted to “GS1.XML.ABIE” as given in Listing 21.

```xml
<owl:Class rdf:ID="GS1.XML.ASBIE">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#AssociationBusinessInformationEntity"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#refersTo"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#GS1.XML.ABIE"/>
          </owl:allValuesFrom>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Listing 21 OWL GS1 ASBIE Class Definition

### 7.1 Explicating the Semantics of GS1 Document Schemas

GS1 XML defines the Business Information Entities either in “pdf” Business Message Specifications or through the Global Data Dictionary (GDD). The GDD is a Web accessible registry, where a component is queried by its name. When the “%” character is entered to the search box, all of the BIEs are returned to the user. Through the browser, only the entity names and the component that contains the entity is displayed, however in the HTML source, the type of the entity (e.g. ABIE, ASBIE or BBIE), the id of the entity and the id of the ABIE it belongs to are also available. In order to create the OWL ontology corresponding to the GS1 XML artifacts, the HTML code is processed and the created classes are inserted to the ontology as the subclasses.

As an example, the following HTML code shown in Listing 22 fragment is for “companyNumber” concept.

```html
<tr><td width='375' valign='top' rowspan='7'>companyNumber</td>
<td><a href="javascript:doDisplayDoc('2.1.0', 'AIDC: GS1 Company Prefix', '723', '6383', 'BBIE')">AIDC: GS1 Company Prefix</a></td>
<td><a href="javascript:doDisplayDoc('2.1.0', 'AIDC: Global Location Number', '724', '6383', 'BBIE')">AIDC: Global Location Number</a></td>
<td><a href="javascript:doDisplayDoc('2.1.0', 'AIDC: Global Service Relation Number', '725', '6383', 'BBIE')">AIDC: Global Service Relation Number</a></td>
<td><a href="javascript:doDisplayDoc('2.1.0', 'AIDC: GlobalReturnableAssetIdentifier', '726', '6383', 'BBIE')">AIDC: GlobalReturnableAssetIdentifier</a></td>
<td><a href="javascript:doDisplayDoc('2.1.0', 'AIDC: GlobalIndividualAssetIdentifier', '727', '6383', 'BBIE')">AIDC: GlobalIndividualAssetIdentifier</a></td>
<td><a href="javascript:doDisplayDoc('2.1.0', 'AIDC: Global Document Type Identifier', '728', '6383', 'BBIE')">AIDC: Global Document Type Identifier</a></td>
</tr>
```

Listing 22 Example HTML code

As shown in this listing, the “companyNumber” has as its id “6383” and it is a BBIE. Furthermore, it exist in components numbered from 723 to 728.
In order to create GS1 XML OWL ontology, the HTML code is processed and the created classes are inserted to the ontology as the subclasses of GS1 BusinessInformationEntity.

Listing 23 provides an example on how "NameAndAddress.Details" artifact of GS1 is represented conforming to the GS1 Upper Ontology.

```
<owl:Class rdf:ID="NameAndAddress.Details">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Address"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasObjectClassTerm"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#Address"/>
          </owl:allValuesFrom>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Listing 23 GS1 "NameAndAddress.Details" Class Definition

Listing 24 provides an example on how "Address" artifact of GS1 is represented conforming to the GS1 Upper Ontology.

```
<owl:Class rdf:about="#Address">
  <rdfs:subClassOf rdf:resource="#ObjectClassTerm"/>
</owl:Class>
```

33
<owl:Class rdf:ID="city">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<owl:Class rdf:about="#GS1.XML.BBIE"/>
<owl: Restriction>
<owl: onProperty rdf:resource="#hasObjectClassTerm"/>
<owl: allValuesFrom>
<owl:Class rdf:about="#Address"/>
</owl: allValuesFrom>
</owl: Restriction>
<owl: Restriction>
<owl: onProperty rdf:resource="#hasRepresentationTerm"/>
<owl:allValuesFrom>
<owl:Class rdf:about="#Text"/>
</owl:allValuesFrom>
</owl:Restriction>
<owl: Restriction>
<owl: onProperty rdf:resource="#usedInContext"/>
<owl:allValuesFrom>
<owl:Class rdf:about="#Context"/>
</owl:allValuesFrom>
</owl:Restriction>
<owl: Restriction>
<owl: onProperty rdf:resource="#hasDataType"/>
<owl:allValuesFrom>
<owl:Class rdf:about="#Text.Type"/>
</owl:allValuesFrom>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<owl:Class>
</owl:Class>

Listing 24 GS1 "Address" Class Definition
8 Explicating Semantics of CCTS based Document Standards – UBL Upper Ontology (Normative)

Figure 5 shows the upper ontology for the UBL document standard and in this figure UBL classes are related with the corresponding CCTS classes by using the “owl:equivalentClass” property.

A UBL BBIE SHOULD be defined as an OWL class named “UBL.BBIE” and it SHOULD be declared equivalent to the BBIE class defined in CCTS upper ontology as given in Listing 25.

```xml
<owl:Class rdf:ID="UBL.BBIE">
  <owl:equivalentClass rdf:resource="#BasicBusinessInformationEntity"/>
</owl:Class>
```

Listing 25 OWL UBL BBIE Class definition

A UBL ABIE SHOULD be defined as an OWL class named “UBL.ABIE” and it SHOULD be declared equivalent to the ABIE class defined in CCTS upper ontology. The “contains” Object Property of the “UBL.ABIE” class should be restricted to “UBL.BBIE” and “UBL.ASBIE” as given in Listing 26.

```xml
<owl:Class rdf:ID="UBL.ABIE">
  <owl:equivalentClass rdf:resource="#AggregateBusinessInformationEntity"/>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#AggregateBusinessInformationEntity"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#contains"/>
          <owl:allValuesFrom>
            <owl:Class>
              <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#UBL.BBIE"/>
                <owl:Class rdf:about="#UBL.ASBIE"/>
              </owl:intersectionOf>
            </owl:Class>
          </owl:allValuesFrom>
        </owl:Restriction>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
</owl:Class>
```

Listing 26 OWL UBL ABIE Class definition
A UBL ASBIE SHOULD be defined as an OWL class named "UBL.ASBIE" and it SHOULD be declared equivalent to the ASBIE class defined in CCTS upper ontology. The "refersTo" Object Property of the "UBL.ASBIE" class should be restricted to "UBL.ABIE" as given in Listing 27.

```xml
<owl:Class rdf:ID="UBL.ASBIE">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#AssociationBusinessInformationEntity"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#refersTo"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#UBL.ABIE"/>
          </owl:allValuesFrom>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Listing 27 OWL UBL ASBIE Class definition

8.1 Explicating the Semantics of UBL Document Schemas

The semantics of UBL artifacts are explicated conforming to the UBL Upper Ontology defined. In UBL, the BIEs are provided in MS Excel spreadsheets. To create UBL artifacts ontology conforming to the specified UBL upper ontology, the UBL spreadsheets are first converted to a custom XML format by using XML Map mechanism of MS Excel. Then, the necessary OWL classes are created from this XML file and populated in the OWL ontology.

Listing 28 provides an example on how "Address.Details" artifact of UBL is represented conforming to the UBL Upper Ontology.

```xml
<owl:Class rdf:ID="Address.Details">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#UBL.ABIE"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasObjectClassTerm"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#Address"/>
          </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#usedInContext"/>
          <owl:allValuesFrom>
            <owl:Class rdf:about="#Context"/>
          </owl:allValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#contains"/>
          <owl:allValuesFrom>
            <owl:Class>
              <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Address.Identifier"/>
                <owl:Class rdf:about="#Address.AddressTypeCode.Code"/>
              </owl:intersectionOf>
            </owl:Class>
          </owl:allValuesFrom>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```
Listing 28 UBL "Address.Details" Class

Listing 29 provides an example on how "Address.Identifier" artifact of UBL is represented conforming to the UBL Upper Ontology.
<owl:Class rdf:about="#Identifier.Type"/>
</owl:Class>
</owl:intersectionOf>

Listing 29 UBL "Address.Identifier" Class
9 Explicating Semantics of CCTS based Document Standards – OAGIS 9.1 Upper Ontology (Normative)

Although GS1 XML and UBL use the same terminology for document artifacts such as ABIE as the CCTS framework, OAGIS names its document components differently.

Figure 6 shows the upper ontology for the OAGIS 9.1 document standard and how OAGIS 9.1 classes are related with the corresponding CCTS classes by using the "owl:equivalentClass" property. An OAGIS 9.1 "Component" corresponds to "ABIE" class in CCTS upper ontology; "ComponentRef" corresponds to "ASBIE" and "Field" corresponds to "BBIE".

An OAGIS 9.1 “Field” SHOULD be defined as an OWL class named “Field” and it SHOULD be declared equivalent to the BBIE class defined in CCTS upper ontology as given in Listing 30.

```xml
<owl:Class rdf:ID="Field">
  <owl:equivalentClass rdf:resource="#BasicBusinessInformationEntity"/>
</owl:Class>
```

Listing 30 OWL OAGIS “Field” Class Definition

An OAGIS 9.1 “Component” SHOULD be defined as an OWL class named “Component” and it SHOULD be declared equivalent to the ABIE class defined in CCTS upper ontology. The “contains” Object Property of the “Component” class should be restricted to “Field” and “ComponentRef” as given in Listing 31.

```xml
<owl:Class rdf:ID="Component">
  <owl:equivalentClass>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#AggregateBusinessInformationEntity"/>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#contains"/>
        <owl:allValuesFrom>
          <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
              <owl:Class rdf:about="#Field"/>
              <owl:Class rdf:about="#ComponentRef"/>
            </owl:intersectionOf>
          </owl:Class>
        </owl:allValuesFrom>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:equivalentClass>
</owl:Class>
```

Listing 31 OWL OAGIS “Component” Class Definition
An OAGIS 9.1 “ComponentRef” SHOULD be defined as an OWL class named “ComponentRef” and it SHOULD be declared equivalent to the ASBIE class defined in CCTS upper ontology. The “refersTo” Object Property of the “ComponentRef” class should be restricted to “Component” as given in Listing 32.

Listing 32 OWL OAGIS “ComponentRef” Class Definition

9.1 Explicating the Semantics of OAGIS 9.1 Document Schemas

The semantics of OAGIS 9.1 artifacts are explicited conforming to the OAGIS Upper Ontology defined. OAGIS provides the XSD schemas of its Components and Fields (e.g. Components.xsd and Fields.xsd) and does not name its components according to ISO 11179 Part 5. Therefore a special adapter is developed to generate the OAGIS document schema ontology as follows: In OAGIS XSD Schemas, each Component is represented with an element declaration and a corresponding type declaration. For example, the “Address” Component shown in Listing 33 contains the following element and type declarations:

Listing 33 OAGIS “Address” Component XML Schema

Furthermore, as shown in Listing 34 several components may have the same complex type.

Listing 34 OAGIS Example Components sharing the same type definition

When constructing the OAGIS document schema ontology, for each element declaration, one ontology class is created. An example is shown in Listing 35, where the OAGIS "Address.Details" artifact is represented conforming to the OAGIS 9.1 Upper Ontology.
As shown in Listing 35, the Component “Address.Details” is defined as common component implying that it is used in the general “Context”.

Listing 35 OAGIS “Address.Details” Class
OAGIS does not provide the “Object Class Terms” for its Components. However a closer investigation reveals that the names of the complex types in OAGIS give the information captured by the Object Class Terms of CCTS. Hence, the complex type names are used as the Object Class Terms of OAGIS artifact and they are obtained by simply dropping the suffix “Type” from the element’s complex type name. For example, as shown in Listing 35, the Object Class Term for Address Component is “Address”.

However, some of the OAGIS Components are defined based on CCL Core Components. In such cases, the Object Class Terms of the corresponding Core Component is used. For example, there is a Component called “ProjectReference” which is of type “ProjectReferenceType” as depicted in Listing 36.

```
<xsd:element name="ProjectReference" type="ProjectReferenceType"/>
```

Listing 36 OAGIS “ProjectReference” Component

“ProjectReferenceType” is derived from “ProjectBaseType” and “ProjectBaseType” is based on “ProjectABIEType”. In this case, the Object Class Term for “ProjectReference” is “Project” as shown in Listing 37.

```
<xsd:complexType name="ProjectReferenceType">
    <xsd:complexContent>
        <xsd:extension base="ProjectBaseType">
            <xsd:sequence>
                <xsd:element ref="ActivityID" minOccurs="0"/>
                <xsd:element ref="UserArea" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Listning 37 OAGIS “ProjectReferenceType”

In OAGIS, a Component is composed of Fields and/or ComponentReferences. Fields are at the leaf level and they are based on CCTS Core Component Types. All the OAGIS Fields are defined in “Fields.xsd” document. They are inserted to the OAGIS Document Schema Ontology as follows: In XSDs, each Field is defined with an element and a corresponding type declaration. The type declaration usually points to the Core Component Type. For example, as shown in Listing 38, the “PostalCode” field is of type “CodeType”.

```
<xsd:element name="PostalCode" type="CodeType"/>
```

Listing 38 OAGIS “PostalCode” field

This field is inserted to the ontology as a new class, which is a restriction on “Field” class. An example on how “PostalCode.CodeType” field of OAGIS 9.1 is represented conforming to the OAGIS 9.1 Upper Ontology is shown in Listing 39. The representation term of this class is “Code” and the data type of this class is “Code.Type”.

```
<owl:Class rdf:ID="PostalCode.CodeType">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Field"/>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
```

Listing 39 Representation of “PostalCode.CodeType”
Considering the Component References, they are inserted to the ontology as new classes, which are restrictions of “ComponentRef” class. For example, the “Preference.Preference” Component Reference is inserted as shown in Listing 40. It should be noted that it is used in general “Context” and refers to “Preference.Details” Component.

Listing 39 OAGIS "PostalCode.CodeType"
9.2 An Overview of SET Upper Ontologies

Figure 7 gives the overview of the SET Upper Ontologies.
9.3 An Overview of SET Upper Ontologies and Document Schema Ontologies

Figure 8 gives an overview of the SET Upper Ontologies and Document Schema Ontologies.

Figure 8 An Overview of SET Upper Ontologies and Document Schema Ontologies
10 Explicating Semantics Related with Different Usages of Document Artifacts in Different Standards (Informative)

10.1 Explicating the Semantics on the Different Usages of CCTS Data Types

Different document standards use CCTS Data Types differently. For example, “Code.Type” in one standard is represented by “Text.Type” in another standard and yet with “Identifier.Type” in another standard. This knowledge in real world should be expressed in such a way that not only the humans but also the reasoner knows about it.

Therefore we provide the following generic rules describing this knowledge.

<table>
<thead>
<tr>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code.Type ≡ Text.Type</td>
</tr>
<tr>
<td>Identifier.Type ≡ Text.Type</td>
</tr>
<tr>
<td>Name.Type ≡ Text.Type</td>
</tr>
</tbody>
</table>

Listing 41 Declaring the equivalent use of different data types

Obviously such rules will not be needed if the document standards truly conform to the CCTS Framework. But in reality, the existing CCTS based document standards have such different usages and to discover related document artifacts in different standards, the reasoner needs this information. It should be noted that such an assertion may produce some false positives, that is, finding two unrelated document artifacts to be equivalent. However, such false positives are in limited numbers since many other semantic properties of the document artifacts are compared to find equivalences. The false positives, when they happen, need to be sorted out manually.
11 Harmonizing the Ontologies of the Document Standards (Informative)

When a DL reasoner runs through the upper ontologies and the document schema ontologies defined, the resulting inferred (harmonized) ontology gives the correspondences between two CCTS based standards like UBL and GS1. The established relationships can be direct indicating that the two document artifacts are equivalent or in subsumption relationship or it can be indirect through CCL, that is, the document artifacts of two different standards may both be a super class (or subclass) of a CCL artifacts and the relationship between them can only be established through CCL.

In this section we present an example on how the correlation of two document components is established in the Harmonized Ontology through the DL reasoning process. In order to facilitate the description of the reasoning process used in this example, we first express the ontology descriptions as specified in this document through their corresponding logical expressions. Table 1 gives the logical expressions corresponding to “CCL Structured Address ABIE” and Table 2 gives logical expressions corresponding to “UBL Address ABIE”.

Table 1  CCL - Structured Address ABIE Assserted Definition

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Structured_Address.BlockName.Text = BasicBusinessInformationEntity</td>
<td>A V hasDataType {Text.Type} A V hasObjectClassTerm {Address} A V hasRepresentationTerm {Text} A V usedInContext {Context}</td>
</tr>
<tr>
<td>3. Structured_Address.BuildingName.Text = BasicBusinessInformationEntity</td>
<td>A V hasDataType {Text.Type} A V hasObjectClassTerm {Address} A V hasRepresentationTerm {Text} A V usedInContext {Context}</td>
</tr>
<tr>
<td>4. Structured_Address.CityName.Text = BasicBusinessInformationEntity</td>
<td>A V hasDataType {Text.Type} A V hasObjectClassTerm {Address} A V hasRepresentationTerm {Text} A V usedInContext {Context}</td>
</tr>
<tr>
<td>5. Structured_Address.CitySub-DivisionName.Text = BasicBusinessInformationEntity</td>
<td>A V hasDataType {Text.Type} A V hasObjectClassTerm {Address} A V hasRepresentationTerm {Text} A V usedInContext {Context}</td>
</tr>
<tr>
<td>6. Structured_Address.Country.Name = BasicBusinessInformationEntity</td>
<td>A V hasDataType {Identifier.Type} A V hasObjectClassTerm {Address} A V hasRepresentationTerm {Identifier} A V usedInContext {Context}</td>
</tr>
<tr>
<td>7. Structured_Address.CountrySub-DivisionName.Text = BasicBusinessInformationEntity</td>
<td>A V hasDataType {Text.Type} A V hasObjectClassTerm {Address} A V hasRepresentationTerm {Text} A V usedInContext {Context}</td>
</tr>
<tr>
<td>8. Structured_Address.Identification.Identifier = BasicBusinessInformationEntity</td>
<td>A V hasDataType {Identifier.Type} A V hasObjectClassTerm {Address} A V hasRepresentationTerm {Identifier} A V usedInContext {Context}</td>
</tr>
<tr>
<td>9. Structured_Address.CountrySub-DivisionName.Text = BasicBusinessInformationEntity</td>
<td>A V hasDataType {Code.Type} A V hasObjectClassTerm {Address} A V hasRepresentationTerm {Code} A V usedInContext {Context}</td>
</tr>
<tr>
<td>10. Structured_Address.StreetName.Text = BasicBusinessInformationEntity</td>
<td>A V hasDataType {Text.Type} A V hasObjectClassTerm {Address} A V hasRepresentationTerm {Text} A V usedInContext {Context}</td>
</tr>
</tbody>
</table>

Table 2 UBL Address ABIE Assserted Definition

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
</table>
Table 3 The Assertion Related with the different Usage of Datatypes

54.Name.Type ≡ Text.Type

Table 4 Inferred Equalities/Subsumptions between CCL Structured Address and UBL Address in the Harmonized Ontology

<table>
<thead>
<tr>
<th>Assertion</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured_Address.CityName.Text ≡ Address.CityName.Name</td>
<td>Def.4, Def.20, Def.54</td>
</tr>
<tr>
<td>Structured_Address.CitySub-DivisionName.Text ≡ Address.CitySubdivisionName.Text</td>
<td>Def.5, Def.21, Def.54</td>
</tr>
<tr>
<td>Structured_Address.StreetName.Text ≡ Address.StreetName.Name</td>
<td>Def.12, Def.36, Def.54</td>
</tr>
<tr>
<td>Structured_Address.Identification.Identifier ≡ Address.Identifier</td>
<td>Def.9, Def.27</td>
</tr>
<tr>
<td>Structured_Address.Postcode.Code ≡ Address.Postal_Zone.Text</td>
<td>Def.11, Def.32</td>
</tr>
<tr>
<td>Structured_Address.Country.Identifier ≡ Address.CountryIdentifier</td>
<td>Def.6, Def.27</td>
</tr>
<tr>
<td>Structured_Address.CountrySub-DivisionName.Text ≡ Address.CountrySubentity.Text</td>
<td>Def.8, Def.22</td>
</tr>
<tr>
<td>Structured_Address.CountryName.Text ≡ Address.Region.Text</td>
<td>Def.7, Def.34</td>
</tr>
<tr>
<td>Structured_Address.BlockName.Text ≡ Address.BlockName.Name</td>
<td>Def.2, Def.17, Def.54</td>
</tr>
<tr>
<td>Structured_Address.BuildingName.Text ≡ Address.BuildingName.Name</td>
<td>Def.3, Def.18, Def.54</td>
</tr>
<tr>
<td>Structured_Address.PlotIdentification.Text ≡ Address.PlotIdentification.Text</td>
<td>Def.10, Def.31</td>
</tr>
<tr>
<td>Address.Details ⊇ Structured_Address.Details</td>
<td>Def.13, Def.1 (since Address.Details contain more BBIEs than Structured_Address)</td>
</tr>
</tbody>
</table>
The Harmonized Ontology contains the fact that “CCL Structured Address ABIE” is a subclass of “UBL Address ABIE” giving a much needed correspondence.

Similarly, Table 5 gives the logical expressions corresponding to “GS1 NameAndAddress ABIE” and Table 6 gives one of the additional assertions as mentioned in Chapter 10.

<table>
<thead>
<tr>
<th>Table 5 GS1 NameAndAddress ABIE Asserted Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>55. NameAndAddress.Details ≡ GS1.XML.ABIE ⊃</td>
</tr>
<tr>
<td>56. city  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>57. cityCode  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>58. countryCode  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>59. countyCode  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>60. crossStreet  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>61. currency  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>62. languageOfTheParty  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>63. name  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>64. pOBoxNumber  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>65. postalCode  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>66. provinceCode  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>67. state  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>68. streetAddressOne  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>69. streetAddressTwo  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>70. geographicalCoordinates  ≡ GS1.XML.ASIBE ⊃</td>
</tr>
<tr>
<td>71. GeographicalCoordinates.Details  ≡ GS1.XML.ABIE ⊃</td>
</tr>
<tr>
<td>72. latitude  ≡ GS1.XML.BBIE ⊃</td>
</tr>
<tr>
<td>73. longitude  ≡ GS1.XML.BBIE ⊃</td>
</tr>
</tbody>
</table>

Table 6 The Assertion Related with the different Usage of Datatypes

| 74. Code.Type  ≡ Text.Type ⊃ Identifier.Type |

Table 7 gives the inferred equalities and subsumptions in the Harmonized Ontology.
### Table 7 Inferred Equalities/Subsumptions between CCL Structured Address and GS1 Name and Address in the Harmonized Ontology

<table>
<thead>
<tr>
<th>CCL Structured Address Terms</th>
<th>GS1 NameAndAddress Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured_Address.CityName.Text ≡ city</td>
<td>name.Text</td>
</tr>
<tr>
<td>Structured_Address.City-SubdivisionName.Text ≡ countyCode</td>
<td>postalCode.Text</td>
</tr>
<tr>
<td>Structured_Address.StreetName.Text ≡ crossStreet</td>
<td>streetAddressOne.Text</td>
</tr>
<tr>
<td>Structured_Address.StreetName.Text ≡ streetAddressTwo</td>
<td>streetAddressTwo.Text</td>
</tr>
<tr>
<td>Structured_Address.Identification.Identifier ≡ name</td>
<td>countryCode.Text</td>
</tr>
<tr>
<td>Structured_Address.CountrySub-DivisionName.Text ≡ provinceCode</td>
<td>streetAddressOne.Text</td>
</tr>
<tr>
<td>Structured_Address.Country.Identifier ≡ countryCode</td>
<td>streetAddressTwo.Text</td>
</tr>
<tr>
<td>Structured_Address.CountrySub-DivisionName.Text ≡ state</td>
<td>postalCode.Text</td>
</tr>
</tbody>
</table>

As it is shown in Table 7, the reasoner discovered that “GS1-NameAndAddress.Details ⊆ CCL-Structured_Address.Details” and as shown in Table 4, the reasoner discovered that “UBL-Address.Details ⊆ CCL-Structured_Address.Details” since the “UBL-Address.Details” and the “GS1-NameAndAddress.Details” both contain more elements than the “CCL-Structured_Address.Details” document artifact. Additionally as shown in Table 4 and Table 7, the Harmonized Ontology contains all the equivalences between the corresponding terms of different document schemas as pictorially depicted in Figure 9.

![Figure 9 The Semantic Equivalences among the BBIEs of UBL-Address, CCL-Structured Address and GS1-NameAndAddress Discovered through the Harmonized Ontology](image-url)
Document component discovery is very important for the following reasons:

- When creating a new document type say a "Planning Document in UBL", it is necessary to find the already existing document components in UBL to be reused. For the document artifacts that do not exist in UBL, CCL must be searched to find the corresponding components.
- Additionally, when we want to transform a document artifact in one standard into another, if we cannot obtain a mapping from the Harmonized Ontology, we may wish to query the Harmonized Ontology to discover the corresponding artifacts manually.

Currently document artifacts are mostly stored in spreadsheets. Also, there is an initiative, called UN/CEFACT Registry Implementation Specification35, for storing/querying CCTS artifacts. However, all of these mechanisms only support keyword-based queries. For example, UN/CEFACT Registry Implementation Specification allows the users to query Aggregate Business Information Entities (ABIE) according to ABIE's name, definition, business term, property term, object class term and the context values, where the ABIE used. Keyword-based queries fail short in the following respects:

- The users usually may not guess the exact keyword for querying.
- It is not possible to query BIEs/CCs based on the components they contain although this type of information is very useful. For instance, a user cannot issue a query to return the ABIEs which contain a "BBIE A", a "BBIE B" and an "ASBIE C".
- Keyword-based queries cannot make use of the class hierarchy. For example, assume that we are looking for a Business Information Entity (BIE) which is related with USA geopolitical context. If the BIE is related with "North America" context node, this BIE is not returned to the user.

Furthermore, the implicit semantic relationships provided by the Harmonized Ontology are very useful for certain type of queries. For this reason we have identified the possible type of queries and formulated them either with SPARQL (if the query does not necessitate reasoning) or in OWL.

The Harmonized Ontology can be queried for discovering the document components and the following types of queries are identified:

- **Keyword Queries**: This type of queries returns the user the BIEs/CCs whose name or description includes a given keyword.
- **Type Queries**: The type queries allow the users to query BIEs/CCs based on their Data Types, to query BIEs according to their CCs or to query ASBIEs/ASCCs according to their source/target ABIEs/ACCs.
- **Structural Queries**: This type of queries allows the users to search for CCs/BIEs according to their structure. For example, the user can query BCCs/BBIEs in a given ACC/ABIE, can query ACCs/ABIEs that contain a BCC/ASCC or BIE/ASBIE expression. The expression can be composed of using logical operators AND, OR or NOT.
- **Context Queries**: The context queries are used for discovering BIEs, which are used in a specified context set or set expression (e.g. Return BIEs which are used in context set S and not used in context set S')
- **Equivalence/Similarity Queries**: Most of the time, the user would like to obtain CCs/BIEs similar to a user-defined CC/BIE. In other words, the user specifies the desired content and issues this content as a query against the Harmonized Ontology.

---

12.1 SPARQL Queries

For keyword, type and structural queries no further reasoning is needed than what is present in the Harmonized Ontology. Therefore for these three types of queries it is enough to formulate them using SPARQL\(^\text{36}\) for efficiency.

An example **Keyword Query** is as given in Listing 42. This query retrieves the CCs or BIEs that contain “address” (Case-insensitive) in their “label” or “comment” elements.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?subject
WHERE
{ ?subject rdfs:label ?label ;
  rdfs:comment ?comment .
  FILTER ( regex(str(?label), "^address", "i") || regex(str(?comment), "^address", "i") )
}
```

Listing 42 An Example **Keyword Query Template** in SPARQL

An example **Type Query** is as given in Listing 43. This query retrieves BIEs (from the Harmonized Ontology) that have “basedOn” property whose range is restricted to a class whose CCs can only come from “Price.Details”. In other words, this query retrieves BIEs derived from “Price.Details” CC.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX uo: <http://144.122.230.79/ontology/UpperOnt.owl#>

SELECT ?subject
WHERE
 ?temp rdf:type owl:Restriction ;
 owl:onProperty uo:basedOn ;
 owl:allValuesFrom uo:Price.Details .
}
```

Listing 43 An Example **Type Query Template** in SPARQL

An example Structural Query is as given in Listing 44. This query retrieves those ASCCs and/or ASBIEs (from the Harmonized Ontology) that have “refersTo” property whose range is restricted to a class whose ACCs and/or ABIEs can only come from “Period.Details”. In other words, this query retrieves ASCCs and/or ASBIEs that refer to “Period.Details” ACCs and ABIEs.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX uo: <http://144.122.230.79/ontology/UpperOnt.owl#>

SELECT ?subject
WHERE
{ ?subject owl:label ?label ;

```

\(^\text{36}\) SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/
WHERE
{ ?subject owl:equivalentClass ?temp .
  ?temp owl:intersectionOf ?temp0 .
  ?temp4 owl:allValuesFrom uo:Period.Details ;
  owl:onProperty uo:refersTo ;
  rdf:type owl:Restriction .
}

Listing 44 An Example Structural Query Template in SPARQL

An example Context query is as given in Listing 45. This query retrieves BIEs from the Harmonized Ontology that have “usedInContext” property whose range is restricted to a context that whose values can only come from “Trade”. In other words, this query retrieves BIEs that are used in “Trade” context.

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX uo: <http://144.122.230.79/ontology/UpperOnt.owl#>

SELECT ?subject
WHERE
{ ?subject owl:equivalentClass ?temp .
  ?temp6 owl:allValuesFrom uo:Trade ;
  owl:onProperty uo:usedInContext ;
  rdf:type owl:Restriction .
}

Listing 45 An Example Context Query Template in SPARQL

12.2 Queries that Require Reasoning Support

Some of the Context and all of the Equivalence type of queries require reasoning support and they are formulated as new class expressions in the OWL Ontology and the result is obtained by computing the new inferred ontology through the reasoner. In other words, the reasoner classifies the newly introduced class and computes its relationships to all the related classes.

For example, assume the user would like to obtain the BIEs used in North America but not in Mexico. The query is as given in Error! Reference source not found..
Listing 46 An Example Context Query

A DL reasoner computes the new hierarchy and relates the similar classes (BIEs) to the Query class through subClassOf or equivalentClass constructs. The class in Listing 47 is the result of the query, as the reasoner puts the Query as the subclass of the class in the inferred hierarchy.

Listing 47 The Result of the Example Context Query in Listing 46

Reasoner support is needed to formulate the similarity/equivalence queries. Assume that a user would like to obtain an Account ACC which has a Type, Amount Type and Identifier. Such a query is formulated as given in Listing 48.
Listing 48 An Example Equivalence Query

The reasoner puts a subclass of relation from the AccountingAccount.Details ACC, shown in Listing 49, to the Query as the result has more BCCs than the Query class.

Listing 49 The Result of the Example Equivalence Query in Listing 48
13 Providing Heuristics to Discover Structurally Different Document Artifacts (Informative)

Chapter 11 describes discovering the equivalences of structurally similar and semantically equivalent document artifacts using the Harmonized Ontology. The semantic properties of the CCTS based document artifacts help discovering the equivalences of structurally similar and semantically equivalent elements. However different document standards use core components in different structures. Semantic properties of document artifacts are not enough to find the similarity of the structurally different but semantically equivalent document artifacts; possible differences in structures must be provided as heuristics to enhance the practical uses of the specified semantics. However different document standards use core components in different structures.

13.1 A Heuristic to Help Finding the Equivalent BBIEs at Different Structural Levels

A problem in finding the equivalent artifacts in two different document schemas is that the same (meaning semantically equivalent) artifacts may appear at structurally different positions. For example, a BBIE, that directly appears under an ABIE in one schema, may be referred to through an ASBIE (at any depth) in another document schema.

So the problem is how to inform the reasoner in an automated way that when an ABIE A1 refers to a BBIE B in an ABIE A2 through an ASBIE AS in one document schema (or through a sequence of ASBIEs-ABIEs nested in each other at any depth), there is a possibility that this BBIE B may appear directly under ABIE A1 in another document schema? The heuristic we developed is as follows: To give a hint to the reasoner of such possibilities, we developed a piece of software that automatically asserts a subsumption hierarchy among the Object Class Terms of such document artifacts. More specifically, when an ABIE A1 refers to a BBIE B in an ABIE A2 through an ASBIE AS in one document schema, the Object Class Term of the BBIE B is made a subclass of ABIE A1. The only implication we have in mind (and hence use this information for that purpose by defining explicit rules) for of such an assertion is that, when an ABIE A1 refers to a BBIE B in an ABIE A2 through an ASBIE AS (or through a sequence of ASBIEs-ABIEs nested in each other at any depth) in one document schema BBIE B can directly appear under ABIE A1 in another document schema. Note that once such an assertion is made, then the reasoner can recursively trace the ASBIEs at any depth.

Note that this semantics is automatically extracted when the document schemas are processed to create the corresponding OWL ontologies. In fact, the Harmonized Ontology provided to the SET TC already contains all such relationships. Obviously, such a semantic declaration may result in false positives. But many of those false positives will be eliminated when the other semantic properties of the document artifacts do not match. Since our purpose is to develop a support tool for humans to use rather than completely automating this process, the humans need to eliminate the remaining false positives if there are any.

As an example, in Figure 10, CCL’s “Buyer_Party.Primary_Identification.Identifier” BBIE corresponds (implies the same meaning) to UBL’s “PartyIdentification.Identifier” BBIE. However, they are positioned differently in their respective schemas: In CCL, “Buyer_Party.Primary_Identification.Identifier” BBIE is directly under “Buyer_Party.Details” ABIE. In UBL, “Party.Details” ABIE has an ASBIE, called “Party.PartyIdentification”, and this ASBIE refers to “PartyIdentification.Details” ABIE and “PartyIdentification.Identifier” BBIE appears under this ABIE.

Hence, for the above example, the class equivalence in Listing 50 is added to the Harmonized Ontology.

```
<owl:Class rdf:about="#PartyIdentification">
    <rdfs:subClassOf rdf:resource="#Party"/>
</owl:Class>
```
Listing 50 A Class Equivalence Example Obtained through the Heuristics Defined

How this semantics is used is explained in Section 13.2.6 through an example.

13.2 Addressing Further Structural Differences in Document Artifacts

A very common structural difference in semantically similar document artifacts is that although some of the semantic properties of a document artifact “A” is the subclass of the corresponding properties of the document artifact “B”, some other properties of “A” are the super classes of the corresponding attributes of “B”. As an example, a document artifact A’s context may be a subclass of document artifact B’s context but the subclass relationship among their other properties may be in the reverse direction.

However, for the purposes of discovering similarity of document artifacts, we note that it may not be important if the direction of the subsumption relations among the corresponding semantic properties of the document artifacts is the same. When our purpose is to find out whether these artifacts are similar, we may not care whether the direction of the subsumption relationship is different among their corresponding attributes because we are aiming at semantics support for human beings; not full automation of the process; the humans can sort out the remaining part.

However, DL reasoners alone do not allow for defining such extra rules. Therefore we define and execute these rules in JESS Rule Engine and carry the results back and forth.37

The structural differences mentioned above may occur among different document artifacts such as BBIEs, ASBIEs and ABIEs and we investigate the possible cases.

13.2.1 Heuristics to Discover Structurally Different BBIEs

Consider the semantic properties of two BBIEs: if each of them is pair wise equivalent or subclasses of each other, for our purposes we consider these BBIEs to be equivalent. The rule Listing 51 states this.

| Rule BBIE: Assume there are two BBIEs, BBIE1 and BBIE2. IF (their Object Class Terms are the same or subclass of each other) AND (their data types are the same or subclass of each other) AND (their context are the same or subclass of each other) AND (their representation terms are the same or subclass of each other), THEN BBIE1 is equals to BBIE2. |

Listing 51 Rule BBIE

It is not possible to express such a rule in DL and hence process it with a DL reasoner alone (unless the reasoner also supports predicate logic reasoning). Therefore, we have expressed this rule in JESS Rule Language as given in Listing 52.

| (defrule structuralBBIE
  (bbie (name ?n1) (objectClassTerm ?o1) (representationTerm ?r1) (context ?c1) (dataType ?d1))
  (bbie (name ?n2) (objectClassTerm ?o2) (representationTerm ?r2) (context ?c2) (dataType ?d2))
  (or (equalsTo (source ?o1) (target ?o2))
    (or (equalsTo (source ?o2) (target ?o1))
        (or (equalsTo (objectClassTerm ?o1) (objectClassTerm ?o2))
            (or (equalsTo (objectClassTerm ?o1) (objectClassTerm ?o2))
                (or (equalsTo (representationTerm ?r1) (representationTerm ?r2))
                    (or (equalsTo (representationTerm ?r1) (representationTerm ?r2))
                        (or (equalsTo (context ?c1) (context ?c2))
                            (or (equalsTo (context ?c1) (context ?c2))
                                (or (equalsTo (dataType ?d1) (dataType ?d2))
                                    (or (equalsTo (dataType ?d1) (dataType ?d2))
                                        (or (equalsTo (name ?n1) (name ?n2))
                                            (or (equalsTo (name ?n1) (name ?n2))
                                                (or (equalsTo (name ?n2) (name ?n1))))))))))))) |

37 When we find an efficient DL reasoner with rule support (such as RacerPro supporting SWRL), this problem will be over.
When this rule fires, it produces the class equivalence in Listing 53 which is added to the Harmonized Ontology and processed once again with the DL reasoner.

```
<owl:Class rdf:about="#BBIE1">
  <owl:equivalentClass rdf:resource="#BBIE2"/>
</owl:Class>
```

Listing 53 The Class Equivalence obtained through the heuristic Rule BBIE

13.2.2 Heuristics to Discover Structurally Different ASBIEs

Consider all the semantic properties of two ASBIEs: if each of them is pair wise equivalent or subclasses of each other, for our purposes we consider these ASBIEs to be equivalent. The rule in Listing 54 states this.

```
Rule ASBIE: Assume there are two ASBIEs, ASBIE1 and ASBIE2. IF (their Object Class Terms are the same or subclass of each other) AND (the ABIEs that they refer to are the same or subclass of each other) AND (their context are the same or subclass of each other) AND (their representation terms are the same or subclass of each other), THEN ASBIE1 is equals to ASBIE2.
```

Listing 54 Rule ASBIE

In fact this rule is very similar to BBIE rule; the differences come from the differences of the semantic properties of BBIEs and ASBIEs; ASBIEs have an extra semantic property called “refersTo” which identifies the ABIE that the ASBIE refers to.

The corresponding JESS Rule is as given in Listing 55.

```
(defrule structuralASBIE
  (asbie (name ?n1) (objectClassTerm ?o1) (representationTerm ?r1) (context ?c1) (refersTo ?d1))
  (asbie (name ?n2) (objectClassTerm ?o2) (representationTerm ?r2) (context ?c2) (refersTo ?d2))
  (or (equalsTo (source ?o1) (target ?o2)) (subClassOf (child ?o1) (parent ?o2)) (subClassOf (child ?o2) (parent ?o1)))
  (or (equalsTo (source ?r1) (target ?r2)) (subClassOf (child ?r1) (parent ?r2)) (subClassOf (child ?r2) (parent ?r1)))
  (or (equalsTo (source ?c1) (target ?c2)) (subClassOf (child ?c1) (parent ?c2)) (subClassOf (child ?c2) (parent ?c1)))
  (or (equalsTo (source ?d1) (target ?d2))))
```

Listing 55 JESS Rule corresponding to the Rule BBIE
When this rule fires, it produces the class equivalence in Listing 56 which is added to the Harmonized Ontology and processed once again with the DL reasoner.

```xml
<owl:Class rdf:about="#ASBIE1">
  <owl:equivalentClass rdf:resource="#ASBIE2"/>
</owl:Class>
```

Listing 56 The Class Equivalence obtained through the heuristic Rule ASBIE

### 13.2.3 Heuristics to Discover Structurally Different ASBIE-BBIE Pairs

Consider two semantically equivalent BBIEs, BBIE1 and BBIE2. If BBIE1 is in ABIE1 and ASBIE1 is referring to ABIE1, there is a possibility that ASBIE1 is semantically equivalent to the BBIE2 (Listing 57).

Rule ASBIE-BBIE: Assume there are two BBIEs, BBIE1 and BBIE2. IF (BBIE1 is equal to BBIE2) AND (BBIE1 is in ABIE1) AND (ABIE1 is referred by ASBIE1), THEN ASBIE1 is equals to BBIE2.

The corresponding JESS Rule is as given in Listing 58.

```jess
(defrule structuralBBIEASBIE
  (abie (name ?a1) (objectClassTerm ?o) (containsSet ?cs1) (context ?c1) (biecount ?bc1))
  (equalsTo (source ?s) (target ?t))
  (contains_bie (containsset ?cs1) (bie ?s))
  (refersTo (asbie ?as) (abie ?a1))
=>
  (assert (equalsTo (source ?as) (target ?t)))
  (assert (equalsTo (source ?t) (target ?as))))
```

Listing 58 JESS Rule corresponding to the Rule ASBIE-BBIE

When this rule fires, it produces the following class equivalence in Listing 59 which is added to the Harmonized Ontology and processed once again with the DL reasoner.

```xml
<owl:Class rdf:about="#ASBIE1">
  <owl:equivalentClass rdf:resource="#BBIE2"/>
</owl:Class>
```

Listing 59 The Class Equivalence obtained through the heuristic Rule ASBIE-BBIE

### 13.2.4 Heuristics to Discover Structurally Different ABIEs

When it comes to ABIEs, the structural differences that can occur are more complex because each ABIE may contain different number of BBIEs some of which may be semantically equivalent, some may not. Therefore while testing whether two ABIEs are semantically similar, their semantic property “contains” gains a special importance.

The “contains” property of an ABIE gives the set of BIEs (the set of BBIEs and ASBIEs) it contains. We define the “ContainsSet” of an ABIE to be the set of all of its BIEs just to simplify the explanation. The “ContainsSet” is in fact the set of BIEs in the range of the “contains” property of an ABIE. The “ContainsSet”s of two ABIEs may be equal; may have a nonnull intersection; may be in subset relationships or may be disjoint of each other. We give a rule to consider different cases. Note that the
semantic properties of two ABIEs other than “contains” must be in relationship for these rules to fire.

Case 1: “ContainsSet”s of two ABIEs are equivalent or in subset relationship

Consider all the semantic properties of two ABIEs: if each of them is pair wise equivalent or subclasses of one another, and their “ContainsSet”s are the same, for our purposes we consider these ABIEs to be equivalent. The rule in Listing 60 states this.

Listing 60 Rule-1 ABIE

The corresponding JESS Rule is as given in Listing 61.

(Listing 61 JESS Rule corresponding to the Rule-1 ABIE)

When this rule fires, it produces the class equivalence in Listing 62 which is added to the Harmonized Ontology and processed once again with the DL reasoner.

(Listing 62 The Class Equivalence obtained through the heuristic Rule-1 ABIE)

Case 2: “ContainsSet”s of two ABIEs have a nonnull intersection

The semantic properties of two ABIEs may be equivalent and their “ContainsSet” may have a nonnull intersection. How to classify these ABIEs is for its user to decide. What we provide is a “similarityConstant” that the user may set. As an example, if the user considers that when 60% of the BIEs of two ABIEs are the same, they may be considered similar, then he can set the “similarityConstant” to “0.6”. For example, the Jaccard index\(^38\) can be used as the similarity coefficient which is a statistic used for comparing the similarity and diversity of sample sets.

Consider all the semantic properties of two ABIEs: if each of them is pair wise equivalent or subclasses of one another, and the BIEs in their “ContainsSet” sets are “similarityConstant” percent equivalent, for our purposes we consider these ABIEs to be equivalent. The rule in Listing 63 states this.

\(^38\) The Jaccard coefficient is defined as the size of the intersection divided by the size of the union of the sample sets.
Rule-2 ABIE: Assume there are two ABIEs, ABIE1 and ABIE2. IF (their Object Class Terms are the same or subclass of each other) AND (their context are the same or subclass of each other) AND (the BIEs in their "ContainsSet" sets are "similarityConstant" percent equivalent), THEN ABIE1 is equals to ABIE2.

The corresponding JESS Rule is as given in Listing 64.

```
(bind ?*count* 0)
(bind ?*similarityConstant* 0.6)

(defrule structuralABIEContainsSetRelation
  (abie (name ?n1) (objectClassTerm ?o1) (containsSet ?cs1) (context ?c1) (biecount ?bc1))
  (abie (name ?n2) (objectClassTerm ?o2) (containsSet ?cs2) (context ?c2) (biecount ?bc2))
  (test (neq ?n1 ?n2))
  (contains_bie (containsset ?cs1) (bie ?bie1))
  (contains_bie (containsset ?cs2) (bie ?bie2))
  (or (equalsTo (source ?bie1) (target ?bie2)) (subClassOf (child ?bie1) (parent ?bie2))
      (subClassOf (child ?bie2) (parent ?bie1)))
=>
  (bind ?*count* (+ ?*count* 1))
  (if (> (/ ?*count* (- (+ ?bc1 ?bc2) ?*count*)) ?*similarityConstant*) then
    (assert (equalsTo (source ?cs1) (target ?cs2)))
    (assert (equalsTo (source ?cs2) (target ?cs1)))
  )
)
```

Listing 64 JESS Rule corresponding to the Rule-2 ABIE

When this rule fires, it produces the class equivalence in Listing 65 is added to the Harmonized Ontology and the Harmonized Ontology is processed once again with the DL reasoner.

```
<owl:Class rdf:about="#ABIE1">
  <owl:equivalentClass rdf:resource="#ABIE2"/>
</owl:Class>
```

Listing 65 The Class Equivalence obtained through the heuristic Rule-2 ABIE

13.2.5 Further Heuristics

Further heuristics can be developed to discover semantically similar document artifacts. However, developing heuristics is not a part of this specification; the ones provided are to demonstrate the use of the specification and hence will not be elaborated any further.

13.2.6 An Example Tracing the Use of the Harmonized Ontology and the Provided Heuristics

To facilitate the explanation of the use of the proposed heuristics together with the Harmonized Ontology, consider the two example document artifacts given in Figure 10. A human looking at this figure can immediately tell the similarity between the UBL "Party.Details" and CCL "Buyer_Party.Details". Our aim is to discover such similarities in an automated way and then generate the XML Transformations, again in an automated way and present the results to the humans for approval or for further modifications.
Figure 10 UBL’s Party ABIE and CCL’s Buyer_Party ABIE

Given these two document artifacts, the DL reasoner first discovers the following equivalence and subsumption relations among the document artifacts, the UBL “PartyIdentification.Identifier” BBIE and the CCL “Buyer_Party.Primary_Identification.Identifier” BBIE as shown in Table 8.

Table 8 The equivalence and subsumption relations among the document artifacts, the UBL “PartyIdentification.Identifier” BBIE and the CCL “Buyer_Party.Primary_Identification.Identifier” BBIE as discovered by a DL reasoner

<table>
<thead>
<tr>
<th>“PartyIdentification.Identifier” BBIE of UBL</th>
<th>“Buyer_Party.Primary_Identification.Identifier” BBIE of CCL</th>
<th>“PrimaryIdentification” Object Class Term is a subclass of “Party” Object Class Term Inserted to the Harmonized Ontology automatically through “Equivalent BBIEs at Different Structural Levels” heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrimaryIdentification</td>
<td>Party</td>
<td>“PrimaryIdentification” Object Class Term is a subclass of “Party” Object Class Term Inserted to the Harmonized Ontology automatically through “Equivalent BBIEs at Different Structural Levels” heuristics</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier</td>
<td>Both BBIEs have the same Representation Term</td>
</tr>
<tr>
<td>Context</td>
<td>Trade</td>
<td>“Trade” context class is a subclass of the “Context” top class</td>
</tr>
<tr>
<td>Identifier.Type</td>
<td>Identifier.Type</td>
<td>Both BBIEs have the same Data Type</td>
</tr>
</tbody>
</table>

Note that the direction of the subsumption relationship between “PrimaryIdentification” and “Party” classes
is in the opposite direction of the subsumption relationship between “Context” and “Trade”. However since the class equivalences coming from the “Rule BBIE” are already in the Harmonized Ontology, the DL reasoner establishes the relationship between the “PartyIdentification.Identifier” BBIE of UBL and the “Buyer_Party.Primary_Identification.Identifier” BBIE of CCL as being equivalent. Furthermore, the reasoner using the facts coming from the “Rule ASBIE-BBIE”, establishes the fact that “Buyer_Party.Primary_Identification.Identifier” BBIE is equivalent to “Party.PartyIdentification” ASBIE.

Similarly, for the UBL “Party.Postal_Address.Address” ASBIE and the CCL “Buyer_Party.Postal.Structured_Address” ASBIE, a reasoner can discover the following equivalences and subsumptions among the document artifacts through the Harmonized Ontology as shown in Table 9.

<table>
<thead>
<tr>
<th>“Party.Postal_Address.Address” ASBIE of UBL</th>
<th>“Buyer_Party.Postal.Structured_Address” ASBIE of CCL</th>
<th>Both ASBIEs have the same Object Class Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party</td>
<td>Party</td>
<td>Both ASBIEs have the same Representation Term</td>
</tr>
<tr>
<td>Address</td>
<td>Address</td>
<td>“Trade” context class is a subclass of the “Context” top class</td>
</tr>
<tr>
<td>Context</td>
<td>Trade</td>
<td></td>
</tr>
<tr>
<td>Address.Details</td>
<td>Structured_Address_Details</td>
<td>“Address.Details”⇒Structured_Address_Details” As obtained in Table 3</td>
</tr>
</tbody>
</table>

Here the direction of the subsumption relationship between “Address.Details” and “Structured_Address.Details” classes is in the opposite direction of the subsumption relationship between “Context” and “Trade”. However since the class equivalences coming from the “Rule ASBIE” are already in the Harmonized Ontology, the DL reasoner establishes the relationship between the “Party.Postal_Address.Address” ASBIE of UBL and the “Buyer_Party.Postal.Structured_Address” ASBIE of CCL as being equivalent.

In Figure 10, for the sake of simplicity some of the BBIEs and ASBIEs of the document artifacts UBL “Party.Details” and CCL “Buyer_Party.Details” are not shown. When the DL reasoner considers these extra BBIE and ASBIEs, the equivalence among the semantic properties becomes as shown in Figure 11. In other words, the BIEs (the set of BBIEs and ASBIEs) of UBL “Party.Details” are a superset of BIEs of CCL “Buyer_Party.Details”.

Table 9 The equivalences and subsumtions among the UBL “Party.Postal_Address.Address” ASBIE and the CCL “Buyer_Party.Postal.Structured_Address” ASBIE as discovered by a DL reasoner
Figure 11 The Relationship between the Semantic Properties of “UBL Party.Details” and “CCL Buyer_Party.Details” Document Artifacts

The relationships shown in Figure 11 together with the Rule-1 ABIE trigger the semantic equivalence of artifacts UBL “Party.Details” and CCL “Buyer_Party.Details”.

14 How Does SET TC Specifications Support Automated XSLT Generation?

The Harmonized Ontology gives the specified as well as the computed equality and subsumption relations among the classes of both the upper ontologies and the document schema ontologies. In order to translate document instances between different document schemas, this knowledge should be used to determine the relationship between the elements in the document instances. In other words, the equivalences discovered should be carried to the data level, where the document instances are described in XML. The instance level translation is achieved using XSLT and the XSLT definitions are generated automatically for the identified mappings. Note that the XSLT definitions are generated only once in the design phase.

The process is as follows:

- While constructing the upper ontologies, the relations between the ontology classes and XSD schema elements are identified and the corresponding XPath expressions are generated.
- To obtain the XSLT definition from a document schema to another document schema:
  - The Harmonized Ontology classes that correspond to BIEs in the document schemas are matched pairwise.
  - Then, the XPath expressions for the identified classes are retrieved.
  - Finally, XSLT expressions are generated automatically from these XPath expressions using the equivalences among document artifacts discovered through the Harmonized Ontology.

The extra elements, for which the methodology cannot establish the corresponding elements in the target document schema (either there is no corresponding element or there is such an element but the methodology is not able to find the related element) are inserted into the extension parts of the target document. The domain experts can further handcraft these XSLT definitions.

14.1 An Example: Translating UBL “Address.Details” to GS1 “Name and Address”

In this section, the translation process is described through an example.

14.1.1 Obtaining the XPath expressions for UBL "Address" ABIE and for its BBIEs/ASBIEs automatically

The ABIEs in UBL are represented with "xsd:complexType" definitions in XSDs. The "xsd:complexType" of "Address.Details" ABIE is "AddressType". In the annotation part of the UBL XSD schemas, the dictionary entry names of the BIEs are also provided. As mentioned previously, the class names for BIEs in the document schema ontologies are generated from the dictionary entry names. Therefore, this information is directly used in constructing the XPath expressions for UBL ontology classes (i.e. this information implicitly describes the ontology class). In Listing 66, a part of UBL "AddressType" XSD declaration is given.

```xml
<xsd:complexType name="AddressType">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:Component>
        <ccts:ComponentType>ABIE</ccts:ComponentType>
        <ccts:DictionaryEntryName>Address.
        <ccts:Definition>Information about a structured address.</ccts:Definition>
      </ccts:Component>
    </xsd:documentation>
  </xsd:annotation>
  ...
</xsd:complexType>
```

The XPath language itself refers to XML elements, not to XSD complex types. Therefore, when generating the XPath expression for the "Address.Details" ABIE, the XML elements that refer to the "AddressType" are collected. In UBL XSDs, the XML elements that refer to "AddressType" are as given in Listing 67.

Listing 66 A part of UBL "AddressType" XSD declaration

```xml
  <ccts:ObjectClass>Address</ccts:ObjectClass>
</ccts:Component>
</xsd:documentation>
</xsd:annotation>
<xsd:element ref="cbc:ID" minOccurs="0" maxOccurs="1">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:Component>
        <ccts:ComponentType>BBIE</ccts:ComponentType>
        <ccts:DictionaryEntryName>Address. Identifier</ccts:DictionaryEntryName>
        <ccts:Definition>An identifier for a specific address within a scheme of registered addresses.</ccts:Definition>
        <ccts:Cardinality>0..1</ccts:Cardinality>
        <ccts:ObjectClass>Address</ccts:ObjectClass>
        <ccts:PropertyTerm>Identifier</ccts:PropertyTerm>
        <ccts:RepresentationTerm>Identifier</ccts:RepresentationTerm>
        <ccts:DataType>Identifier. Type</ccts:DataType>
        <ccts:AlternativeBusinessTerms>DetailsKey</ccts:AlternativeBusinessTerms>
      </ccts:Component>
    </xsd:documentation>
    </xsd:annotation>
  </xsd:element>

  <xsd:element ref="LocationCoordinate" minOccurs="0" maxOccurs="1">
    <xsd:annotation>
      <xsd:documentation>
        <ccts:Component>
          <ccts:ComponentType>ASBIE</ccts:ComponentType>
          <ccts:DictionaryEntryName>Address. Location Coordinate</ccts:DictionaryEntryName>
          <ccts:Definition>An association to Location Coordinate.</ccts:Definition>
          <ccts:Cardinality>0..1</ccts:Cardinality>
          <ccts:ObjectClass>Address</ccts:ObjectClass>
          <ccts:PropertyTerm>Location Coordinate</ccts:PropertyTerm>
          <ccts:AssociatedObjectClass>Location Coordinate</ccts:AssociatedObjectClass>
        </ccts:Component>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>

</xsd:sequence>
</xsd:complexType>
```
Listing 67 The UBL elements that refer to "AddressType"

Hence, the XPath expression for "Address" ABIE is as given in Listing 68.

Listing 68 The XPath expression for "Address" ABIE

This XPath expression states that "Address.Details" ontology class corresponds to Address, ApplicableAddress, ApplicableTerritoryAddress, DeliveryAddress, DespatchAddress, JurisdictionRegionAddress, OriginAddress, PostalAddress and RegistrationAddress XML elements in a UBL XML document. The "//" in the beginning of the XPath expression states that the element can be at any depth in the XML document. The XPath expressions (Listing 69) for the BBIEs and ASBIEs in the "Address.Details" ABIE are generated by concatenating the corresponding XML elements names to the above XPath expression.

Listing 69 The XPath expressions for the BBIEs and ASBIEs in the "Address.Details" ABIE

In the above example, XPath expressions for "AddressLine", "Country" and "LocationCoordinate" ABIEs are also provided for the sake of completeness.

14.1.2 Obtaining XPath expressions for GS1 "NameAndAddress" ABIE and for its BBIEs

The ABIEs are represented in GS1 XML's XSD schemas through "xsd:complexType" declarations.
Furthermore, in GS1 XML, the "xsd:complexType" name of an ABIE can be identified by concatenating "Type" keyword to the ABIE name. Therefore, the complex type name of "NameAndAddress" is "NameAndAddressType". The next step is to find the declaration of this type. In GS1 XML, for all ABIEs there is either a separate XSD file (which can be identified by concatenating ".xsd" extension to the ABIE's name, e.g., "NameAndAddress" has "NameAndAddress.xsd") or the ABIE's "xsd:complexType" declaration is in the parent ABIEs XSD file (e.g. "GeographicalCoordinates" ABIE's declaration is in "NameAndAddress.xsd"). Therefore, the complex type "NameAndAddressType" is declared in "NameAndAddress.xsd" file, as given in Listing 70.

```
<xsd:complexType name="NameAndAddressType">
  <xsd:sequence>
    <xsd:element name="city">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="35"/>
          <xsd:minLength value="1"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    ...
    <xsd:element name="geographicalCoordinates" type="eanucc:GeographicalCoordinatesType" minOccurs="0"></xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Listing 70 The complex type "NameAndAddressType"

As mentioned previously, the XPath language itself refers to XML elements, not to XSD complex types. Therefore, when generating the XPath expression for the "NameAndAddress" ABIE, the XML elements that refer to the "NameAndAddressType" should be identified. In GS1 XSDs, one of the XML elements that refer to "NameAndAddressType" is "shipToNonCommercial" as shown in Listing 71.

```
<xsd:element name="shipToNonCommercial" type="eanucc:NameAndAddressType"></xsd:element>
```

Listing 71 "shipToNonCommercial" element in GS1 XSD

The XPath expression for "NameAndAddress" ABIE is as given in Listing 72.

```
#NameAndAddress.Details  --> //shipToNonCommercial
```

Listing 72 XPath expression for "NameAndAddress" ABIE

Furthermore, the XPath expressions for the BBIEs in "NameAndAddress" can be obtained by concatenating the corresponding element name to the above XPath expression. The XPath expressions for the BBIEs of "NameAndAddress" are as given in Listing 73.

```
#city    --> //shipToNonCommercial/city
#cityCode   --> //shipToNonCommercial/cityCode
#countryCode   --> //shipToNonCommercial/countryCode/countryISOCode
#countyCode   --> //shipToNonCommercial/countyCode
#crossStreet   --> //shipToNonCommercial/crossStreet
#currency   --> //shipToNonCommercial/currency
#languageOfTheParty   --> //shipToNonCommercial/languageOfTheParty
#name   --> //shipToNonCommercial/name
#pOBoxNumber   --> //shipToNonCommercial/pOBoxNumber
#postalCode   --> //shipToNonCommercial/postalCode
#provinceCode   --> //shipToNonCommercial/provinceCode
```

Listing 73 XPath expressions for "NameAndAddress" ABIE's BBIEs
For the sake of completeness, the XPath expressions of "GeographicalCoordinates" are also provided.

### 14.1.3 Constructing the XSLT Definitions

In constructing the XSLT definitions, the generated XPath expressions for the ontology classes and the semantic equivalences discovered are used.

As an example, Figure 9 in Chapter 11 gives the semantic equivalences among the BBIEs shown in Listing 74.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:eanucc="urn:ean.ucc:2">
    <xsl:output method="xml" encoding="utf-8" indent="yes" omit-xml-declaration="yes"/>
    <xsl:template match="/"
        call-template name="nameAndAddress"/>
</xsl:stylesheet>
```

The XSLT Definition in Listing 75 is constructed using the equivalences given in Listing 74 and the generated XPath expressions to convert an UBL "Address" ABIE instance to GS1 "NameAndAddress" instance.
Listing 75 An Example XSLT Definition constructed using the equivalences given in Listing 74 and the generated XPath expressions to convert an UBL "Address" ABIE instance to GS1 "NameAndAddress" instance

Assume that the user has the UBL "Address" instance given in Error! Reference source not found..
Listing 76 An Example UBL “Address” Instance

After applying the XSLT definitions to the example UBL “Address” Instance given in Listing 76, the GS1 "NameAndAddress" instance in Listing 77 is obtained.
<countryCode>
    <countryISOCode>1</countryISOCode>
</countryCode>

<countryCode>UBLCitySubdivisionName</countryCode>
<crossStreet>UBLStreetName</crossStreet>
<name>1</name>
<postalCode>UBLPostalZone</postalCode>
<provinceCode>UBLCountrySubentityCode</provinceCode>
<state>UBLCountrySubentityCode</state>
<streetAddressOne>UBLBlockName UBLBuildingName
UBLPlotIdentification UBLRegion</streetAddressOne>
<streetAddressTwo>UBLBlockName UBLBuildingName
UBLPlotIdentification UBLRegion</streetAddressTwo>
</eanucc:shipToNonCommercial>

<extension>
    <cbc:AddressTypeCode>UBLAddressTypeCode</cbc:AddressTypeCode>
    <cbc:Postbox>UBLPostbox</cbc:Postbox>
    <cbc:Floor>UBL2</cbc:Floor>
    <cbc:Room>UBL3</cbc:Room>
    <cbc:AdditionalStreetName>UBLAdditionalStreetName</cbc:AdditionalStreetName>
    <cbc:BuildingNumber>UBL4</cbc:BuildingNumber>
    <cbc:InhouseMail>UBLInhouseMail</cbc:InhouseMail>
    <cbc:Department>UBLDepartment</cbc:Department>
    <cbc:MarkAttention>UBLMarkAttention</cbc:MarkAttention>
    <cbc:MarkCare>UBLMarkCare</cbc:MarkCare>
    <cbc:CountrySubentity>UBLCountrySubentity</cbc:CountrySubentity>
    <cbc:District>UBLDistrict</cbc:District>
    <cbc:TimezoneOffset>UBLTimezoneOffset</cbc:TimezoneOffset>
    <cac:AddressLine>
        <cbc:Line>UBLLine</cbc:Line>
    </cac:AddressLine>
    <cac:Country>
        <cbc:Name>UBLName</cbc:Name>
    </cac:Country>
    <cac:LocationCoordinate>
        <cbc:CoordinateSystemCode>UBLCoordinateSystemCode</cbc:CoordinateSystemCode>
        <cbc:LatitudeDegreesMeasure unitCode="04">UBL0.0</cbc:LatitudeDegreesMeasure>
        <cbc:LatitudeMinutesMeasure unitCode="04">UBL0.0</cbc:LatitudeMinutesMeasure>
        <cbc:LongitudeDegreesMeasure unitCode="04">UBL0.0</cbc:LongitudeDegreesMeasure>
        <cbc:LongitudeMinutesMeasure unitCode="04">UBL0.0</cbc:LongitudeMinutesMeasure>
    </cac:LocationCoordinate>
</extension>
Listing 77 A GS1 "NameAndAddress" instance

It should be noted that the unmapped elements are inserted to the extension part of the document.
15 The Overall SET Framework (Informative)

In this section, we present the overall framework in which the SET TC Specifications work as shown in Figure 12, to put together all the parts described and demonstrate how the specified semantics can be used.

Figure 12 The SET Overall Framework

As shown in Figure 12, the SET Framework is composed of the following components:

A. Harmonized Ontology: This ontology contains two types of OWL-DL ontologies: (1) the Upper Ontology and (2) the Document Schema Ontologies. The Upper Ontology describes the CCTS artifacts, such as Business Information Entities and Core Components, as generic classes. The
Document Schema Ontologies, on the other hand, describe the BIEs generated by the CCTS based electronic business document standards. There is a Document Schema Ontology for each electronic business document standard. The Document Schema Ontologies are defined conforming to the generic classes defined in the Upper Ontology. The Harmonized Ontology is obtained by running the DL-Reasoner against the Upper Ontology and Document Schema Ontologies.

B. DL-Reasoner: A Description Logic (DL) Reasoner is used to identify the equivalence and subsumption relations in the Harmonized Ontology. As the DL reasoner, Racer Pro 1.9.2 Beta is used. The discovered similarities among the document artifacts are then used to generate XSLT definitions for transforming between different electronic business document standards' XML instances.

C. Predicate Logic Rule Engine: In some cases, the Description Logic is not sufficient to find relations between document artifacts. Therefore, in these cases, generic heuristics in the form of Predicate Logic Rules are used. The JESS Rule Engine is used to execute the heuristics to find additional relations among the Document Schema Ontology classes.

D. Ontology-PL Facts Wrapper: The document artifacts are represented through OWL classes and properties in the Harmonized Ontology and they are represented as facts in the Predicate Logic Rule Engine. This wrapper converts the OWL definitions to facts definitions which are then asserted to the rule engine. After the rule engine is executed, new relations among the classes are inferred. These new relations are also represented as facts. The wrapper converts the newly obtained fact definitions back to OWL class equivalences to be inserted to the Harmonized Ontology.

E. Additional Heuristics: These heuristics are given through the Predicate Logic Rules to identify the relations among the document artifacts which cannot be identified through Description Logic.

Additionally a number of tools are developed to support this framework:

1. The Set XDS-OWL Converter: This component converts a CCTS based document schema into OASIS SET TC OWL Definition.
2. The XPath Generator Tool: The XPath Generator extracts the correspondences between the XSD Schema elements in the document schemas and the OWL classes in the Document Schema Ontologies through XPath expressions. These expressions are then used to generate the XSLT definitions.
3. The XSLT Generator Tool: This component generates the XSLT definitions by using the XPath definitions and the newly computed equivalence/subsumption relations. These XSLT definitions are used in the transformation between two XML Instances conforming to different electronic business document standards.
4. Ontology-PL Facts Wrapper as explained above.

This framework is used to transform a source XML instance to a target XML instance. If the document schemas of these instances are already in the Harmonized Ontology, the corresponding XSLT transformations can directly be used.

If the source, or the target or both of these document schemas are not yet in the Harmonized Ontology, first they must be inserted through the following procedure as shown in Figure 12:

1a. First the OWL Document Schema Ontology conforming to the OASIS SET specifications are created from the XSD Document Schemas (2a) by using the Set XDS-OWL Converter.
1b. In the mean time, the XPath Generator Tool is used to keep track of the correspondences among the XSD elements and OWL classes to generate XPath Mappings (2b).
3. The OWL ontologies are inserted to the Harmonized Ontology.
4. The ontologies are classified with the DL Reasoner and the Harmonized Ontology is computed. In this step new equality/subsumption relations are produced (5).
5. At the same time, the OWL definitions are converted to Predicate Logic facts by using the Ontology-PL Facts Wrapper Tool and asserted to the Predicate Logic Rule Engine.
6. The rule engine is executed and new relations are identified as facts (7).
8. These facts are converted to OWL Definitions through Ontology-PL Facts Wrapper.
9. The newly generated OWL Definitions are appended to the Harmonized Ontology and the DL-Reasoner is executed again to compute new equality/subsumption relations. The steps from 5 to 9 are executed repeatedly until the Harmonized Ontology reaches a certain maturity level.
10. The equality/subsumption definitions and the XPath Mappings are input to XSLT Generator to produce XSLT Definitions automatically (11).
12. The XSLT Definitions are displayed to the user for further editing (13).
14. The XSLTs are used to transform to XML Instances conforming to different standards.
16 Performance of the System (Informative)

Related with performance, an issue that needs to be addressed is whether the gain in automation justifies the resources needed to develop the ontological representation of the document schemas. In order to reduce this cost, as already described, automated tool support is provided to create OWL definitions of the document schemas. Additionally, by conforming to a standard ontological representation and hence having all the document schema ontologies in a common pool, the users of the Harmonized Ontology only need to create a document schema ontology if it is not already in the Harmonized Ontology and benefit from all the existing connections when they do so.

Another issue related with performance is the computational complexity of the reasoning process involved. The current version of the Harmonized Ontology contains the ontological representations of:

- All of the CCs and BIEs in CCL 07B.
- All of the BIEs in the common library of UBL 2.0.
- All of the OAGIS 9.1 Common Components and Fields.
- All of the elements in the common library of GS1 XML.

There are about 4758 Named OWL Classes and 16122 Restriction Definitions in the current version of the Harmonized Ontology. On a PC with 2GB RAM, the Racer Pro 1.9.2 Beta reasoner takes about 120 seconds to compute the Harmonized Ontology. Considering that the Harmonized Ontology will be recomputed only when a new document schema or a new CCTS based upper document ontology is introduced to the system, this performance is quite acceptable.

Yet, as the Harmonized Ontology evolves (i.e. as more document schema ontologies are introduced) the number of classes will increase. However, Figure 13 indicates that there is no sharp increase in the execution time as the number of classes increase.

![Figure 13 Number of Classes vs. Execution Time](image)