eBTWG – Scope & Requirements for UML2XML Design Rules

UN/CEFACT/CSG/eBTWG
UML2XML Design Rules

Revision #1

17 Oct 2001

Table of Contents

Status of this Document

This document specifies an eBTWG WORK IN PROGRESS for the UN/CEFACT eBusiness community.

Distribution of this document is unlimited.

The document formatting is based on the eBTWG Standard format.

This version: http://www.ebtwg.org/...

Previous version: http://wwwebtwg.org/...

Introduction

Summary of Contents of Document

This specification specifies the scope and requirements for the “UML2XML” project.

Audience

The initial target audience is the UML2XML project team, the eBTWG Steering Committee and other eBTWG project teams that have an interface with the UML2XML project scope.

Related Documents

As mentioned above, other documents provide detailed definitions of some of the components of XXXX and of their inter-relationship. They include [ebXML | eBTWG] Specifications on the following topics:

...

Objectives

Goals/Problem Description

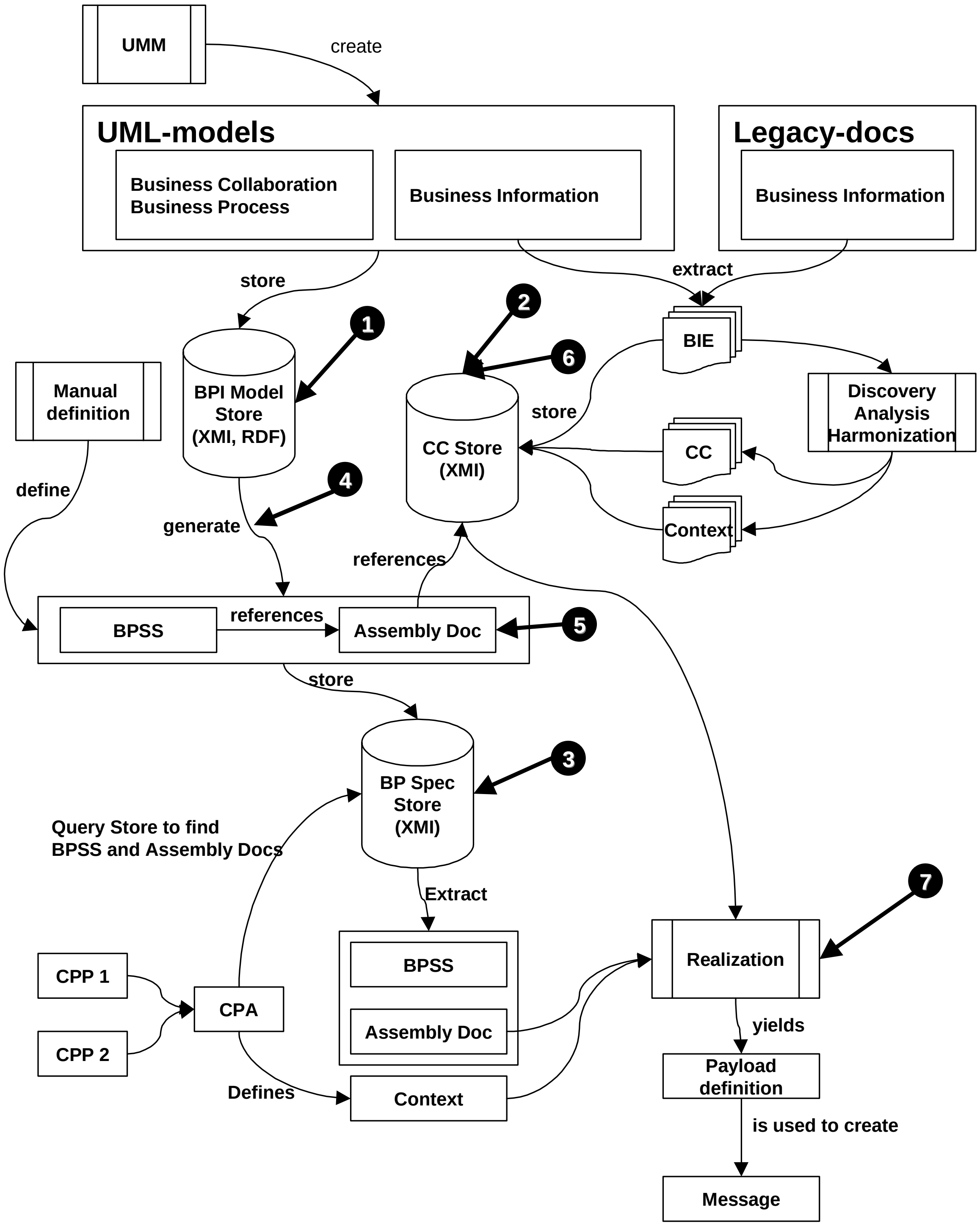

Figure 1 - UML2XML in the e-Business Architecture

Figure 1 visualizes the relationship between the major concepts of the UN/CEFACT EBTWG e-Business architecture that are relevant to the UML2XML project scope. The text below explains this picture, identifies the elements that are part of the UML2XML project scope and identifies their relation with other eBTWG projects.

1) **Storage of standards development models.**

   Under normal conditions all standards development will be done according to the UMM-methodology, leading to a set of UML-models that describe the business collaborations, the business processes and the business information. These models will be stored in the "BPI Model Store" in a format that allows the exchange of these models. This requirement is currently being defined in the BPIMES-project (Business Process and Information Model Exchange Scheme). The UML2XML project will support the BPIMES-project by investigating the suitability of XMI and RDF as the format to store and retrieve these models (see ❶ on Figure 1).

2) **Storage of Core Components (and related elements)**

   Core Components (CCs) are discovered based on so-called "Business Information Entities" (BIEs). These BIEs are either extracted from business information models that are based on the UMM-approach or from legacy documents that describe existing standards. In both cases a discovery, analysis and harmonization process will lead to the definition of Core Components and Contexts. BIEs, CCs and context are all stored in the "CC Store" in line with the technical specifications defined by the Core Components project. The UML2XML project will support the Core Components project by investigating the suitability of XMI as the format to store and retreive these elements (see ❷ on Figure 1).

3) **Storage of formal specifications for e-Business solutions**

   e-Business solutions are formally described in a Business Process Specification Scheme (BPSS). This BPSS contains the process steps that will be executed in an e-Business solution and references the documents that need to be exchanged during these process steps. The documents to be exchanged are formally described in "Assembly Documents" that refer to the used CCs and BIEs. The definition of a BPSS and its Assembly Documents will normally be the result of the UMM-approach, continuing from the Business Process and Information Models. Nevertheless, the possibility to define both BPSS and Assembly Documents in a manual way must be considered as well. Both BPSS and Assembly Documents are stored in a "BP Specification Store". The UML2XML project will support the BPSS project at two levels:

   a) By investigating the suitability of XMI as the format to store and retrieve the BPSS and the Assembly Documents (see ❸ on Figure 1).

   b) By defining a set of design rules to generate the BPSS and Assembly Documents from the Business Process and Information Model (see ❹ on Figure 1).

4) **Message realization**

At some point two companies represented by their CPP (i.e. their list of supported business processes and contexts) will engage in a CPA (i.e. a contract to execute a business process in a given context). The CPA will be used to extract the required BPSS and corresponding Assembly Documents from the “BP Specification Store”. The CPA will also define the full context that will govern the business process. The combination of a particular Assembly Document and the CPA-context will make it possible to define the required Payload Definition, by extracting the relevant BIEs from the CC-store. The resulting Payload Definition will then be realized in the required syntax (i.e. syntax-binding) in order to be used in Messages between the CPA-partners. The UML2XML project will support this message realization at following levels:

a) By defining rules and guidelines on the syntax-neutral definition of Assembly Documents and their relation to CCs and BIEs (e.g. by the use of a modeling notation such as UML) (see ❺ on Figure 1).

b) By defining the set of meta-information that is required to realize a Payload Definition in XML. This meta-information will be stored in the CC Store (see ❻ on Figure 1).

c) By defining a set of design rules to convert syntax-neutral BIEs into an XML-realization (see ❼ on Figure 1).

Requirements

Based on the above goals and problem description following initial requirements can be defined:

R01 Formal definition of Stores

It will be possible to describe the organization and structure of each of the identified stores (“BPI Model Store”, “BP Specification Store” and “CC Store”) according to a formal storage scheme description.

[Note] The current thinking is to investigate the suitability of XMI2 for this formal storage scheme description.

R02 Validation capabilities

It will be possible to validate the conformance of all information that is stored in each of the identified stores (“BPI Model Store”, “BP Specification Store” and “CC Store”) with the formal storage scheme description.

R03 Retrieval capabilities

It will be possible to search and retrieve all information in each of the identified stores (“BPI Model Store”, “BP Specification Store” and “CC Store”) based on multiple search criteria..

[Note] The list of required search criteria will have to be detailed per Store.

R04 Implementation independence

The formal scheme description will not a priori preclude or enforce any specific implementation of the identified storage scheme.

[Note] This requirement aims at openness versus multiple XML-implementations like DTD, W3C-Schema or Relax NG.

R05 Support of UML patterns

The storage scheme will fully support a pre-defined subset of UML-artifacts. This pre-defined set of UML-artifacts will cover all UML-requirements related to the UMM-compliant e-Business standards development.

[Note] This subset will have to be detailed in function of the supported projects (e.g. UMM, BPIMES, BPSS, CC). In case these projects are not explicit about the way to use UML, this information will either have to be added to these specifications or will have to be included in the UML2XML technical specification.

[Note] This may also include the need to support UML-stereotypes.

R06 UMM support

Any reusable artifacts like CCs, BIEs and Assembly Documents will be stored in such a way that they can be easily accessed and reused during UMM-compliant standards development.

[Note] This means that it must be possible to easily include these reusable artifacts in UML-diagrams.

R07 Core Component compliance

The realization of BIEs in XML will comply with the technical specification for Core Components.

[Note] It will be necessary to identify the CC-requirements that are relevant for the above (e.g. possibility to support context, UID, all data types that are defined for CCs, etc.)

[Note] Need to verify whether the current CC-specification supports the use of URIs as data types (e.g. to refer to another document rather than to integrate its content)

R08 Processing instructions

It will be possible to indicate that some processing is required at a specific place in a realized XML-document (e.g. encryption, data authentication and enrichment).

[Note] This "generation" requirement will require a matching "storage" requirement, indicating the need to store processing instructions in the relevant stores.

R09 Naming conventions

Naming conventions defining rules and guidelines for the naming of XML-elements and XML-attributes will be defined. This will include rules regarding the character set to use for XML-names.

R10 Predictability

The set of XML design rules will ensure a maximum predictability of the resulting payload definition of the XML-instance.

[Note] This implies the need to restrict flexibility, meaning that it would for instance be necessary to define the set of parameters to use for XMI2 (in case XMI2 would be retained as a final solution).

R11 Automation

It will be possible to automate the set of XML design rules.

R12 XML-specific information

It will be possible to store any XML-specific information that is required to realize a Payload Definition in XML.

[Note] The exact list of required XML-specific information will have to be specified. This list will be as limited as possible in order to comply with requirement R08 on predictability, but there might for instance be a requirement to store an XML-name for a BIE.

R13 Business Rules

It will be possible to define how and when business rules will be captured in UML, in XML instances and possibly in specific XML-implementations (e.g. W3C Schema or Relax NG).

[Note] This might take into account the fact that rules may reside in another document than the actual data being transferred and therefore the need for referencing and/or linking.

R14 Trace-ability

It will be possible to refer any element and attribute in an XML-instance back to its syntax-neutral definition in the "CC Store".

It will be possible to refer any XML-document back to its syntax-neutral definition in the "BP Specification Store" and/or in the "BPI Model Store".

[Note] This doesn't necessarily mean that each instance must contain full references: information might also be documented in a "documentation scheme".

R15 Performance

The XML design rules will take into account any requirements related to a performant processing of a resulting XML-instance.

[Note] More specific / explicit requirements have to be defined. Some initial thoughts have indicated the document size, the possibility to send partial documents (e.g. containing only what has changed, etc.).

R16 Character set

The resulting XML-instance will be capable to transport information using UTF-8.

[Note] The reasons for this requirement are that (1) UTF-8 can represent almost any known character, (2) UTF-8 is interoperable with many other encoding schemes through (automated) conversion algorithms and (3) UTF-8 is the shortest method to represent the characters that are commonly used in many environments (ASCII and EBCDIC).

[Ed. Note] The above list of requirements has partially been based on the initial requirements document that was made during the meeting in San Francisco. More work is still required to complete the inclusion of these requirements (see below the list that hasn't been processed yet) and to add requirements that are relevant to the full scope of the project.

1. Generic versus specific tags
How do you model documents; how do you decide to reuse a generic component (e.g. a person) or rather a specific component (e.g. a driver). Has also to do with the use of optionality and the use of inheritance (choice). Need to verify whether the complete requirement can be captured in UML. Possible option is the use of inheritance and packages.
Needs to be supported by majority of products (parsers, …)

2. Style sheet support
Do we need design rules for style sheets as well (to get a standardized look). Style sheets can also be useful for business rules. We might need information in UML to influence the resulting style sheet (e.g. important for small & medium enterprises (SMEs) where they want to display the document).

3. Look & Feel (e.g. Depth first?, Ordering of declarations & definitions)
What features are required in UML to be able to control the look & feel? Importance of human-readability and human-editable. This is related to how automatable it will be. Have a look at options Near & Far.

4. Maintenance
Rules for maintenance between major releases (e.g. only allowed to add optional elements).

5. Backward/Upward compatibility
Need to offer at least a minimum (= needs to be defined) of compatibility. May need different approaches for major releases. This may impact the need to edit the results.

6. Versioning
What will the versioning scheme be (minor release vs major release vs maintenance). Need to capture version of the spec according to which the XML is generated + version of the tool + version of the (modeling) elements that have been used.

7. Header & envelopes
There needs to be a way to generate the full document (header + payload). Need to look at how to model the header.

8. No transformation
We don't want to have transformation of documents between sender & receiver.

9. One message per schema?
Will we only support one document per schema or also schema containing multiple documents? Also look at the need to split large documents over multiple instances.

Caveats and Assumptions

This specification is developed to...

Overview

What the XXXX Specification Does

XXXX provides yyyy to ...

How the XXXX Specification Works

XXXX uses yyyy to ...

Where the XXXX Specification May Be Implemented

There are several ways in which xxxx may be implemented ...

XXXX

This section defines ...[used to outline more or all details of this specific specification].

References

This section defines ...[used to outline more or all details of this specific specification].

Disclaimer

Contact Information

ebTWG Chair - Klaus-Dieter Naujok, knaujok@home.com

Project Team Membership

Project Team Leader – Frank Vandamme, frank.vandamme@swift.com

Project Editor – Barbara Price, bprice@us.ibm.com

Editing Team - name, email address

Project Team - name, email address

Copyright Statement