ebXML Core Component Project Team
3/Jul/2000

Methodology for describing Core Components Analysis and Proposal

(Draft-Rev.04)

By Work Group 7 of Core Component Project Team

Hisanao Sugamata
Martin Bryan

Contents

1. Introduction

The first step for achieving to get specifications of the ebXML core components will be analysis of the existing data models currently used in e-business. The concept model for describing the business entities, which were nominated as the candidates of the core components at the 2nd project team meeting of ebXML in Orlando, will be proposed in the paper. This paper will also offer the templates for describing the results of analysis on the business entities.

The purposes of the paper includes,

1. To show the way to OO-edi methodology through the reverse engineering for the current EDI messages for business experts, who have been engaged in designing legacy EDI messages; and
2. To show how to analyze business entities currently being used in EDI in order to design the object classes for object modeling experts, who have been engaged in designing the software in the manner of Object Oriented Approach (OOA).

It is expected that business entities, which are selected and analyzed by the business experts of the Core Component Working Group, can be used in the business process models. The entities can also be stored in the repository(s) aligning with the ebXML standards.

The analysis and proposal prepared in this paper, respecting the ebXML requirements specification, is intended to be,

1. Syntactically neutral,
2. Conforming to ISO11179 and
3. Aligning with the Unified Modeling Methodology.
4. A mechanization for developing Core Components

Revision note :

There have been several discussions around the first draft of this paper.
One of the disputing subjects was how to describe the analysis pattern for the business entities.
You can find the analysis and the proposal for describing the pattern in the section 4.5.
In the 3rd revision of the draft paper, Representation Class candidates are introduced in the section 3.1.
In addition to the previous version of the paper, Mr. Martin Bryan proposed the XML based templates instead of the paper-based form for mechanizing the development of Core Components.
In the appendix file attached you can find the usable templates in XML based form.
In the 4th revision of the draft paper, the XML based templates are amended through some testing.
Also the meta model for Core Component is added in the section 3.2.

2. Business entity analysis

At the 2nd meeting of the core component team in Orlando, typical business entities were on the table for discussion. One of of the business entities discussed was 'Party'.

2.1 Example <Party>

The business entity 'Party' is observed in several patterns or within contexts. The 'Party' is defined within segment groups in the UN/EDIFACT messages.
In the UN/EDIFACT messages of ORDERS and INVOIC, the pattern of the business entity 'Party' has rich attributes which is specially used in purchasing applications.

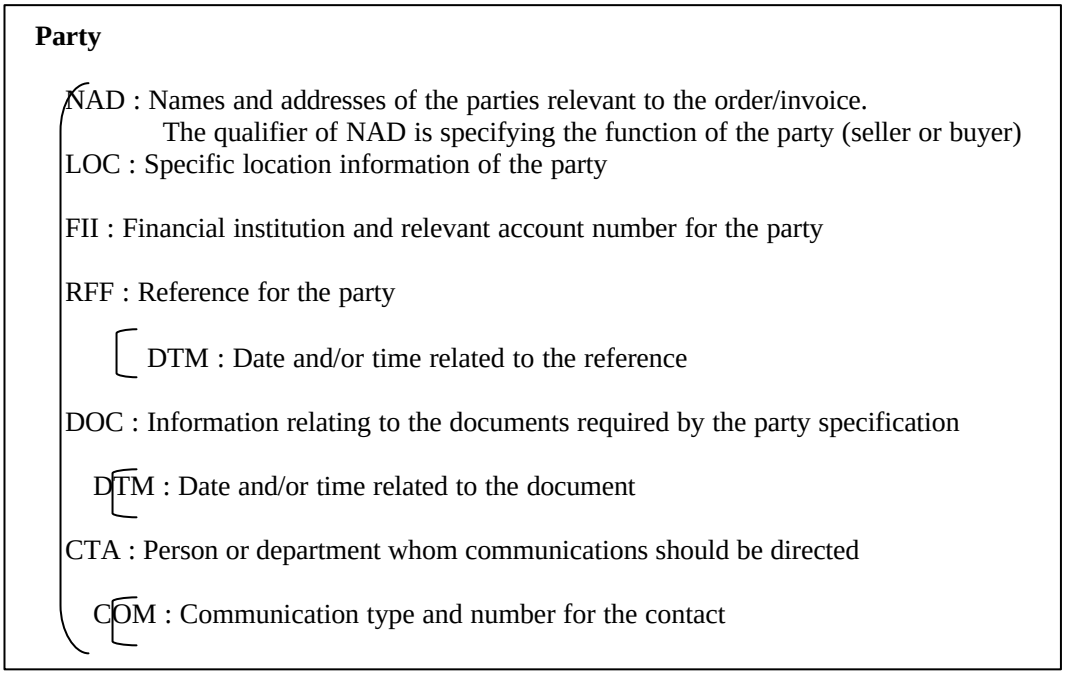


Fig.1 Party example – 1

On the other hand, the 'Party' entity in other message types have a simpler pattern. The simple patterns can be seen in the messages, such as CUSDEC (Customs declaration message), DELFOR (Delivery schedule message), BANSTA (Banking status message), BUSCRD (Business credit report message) and IPPOAD (Insurance policy administration message). The typical pattern of the simple form of 'Party' is as follows.

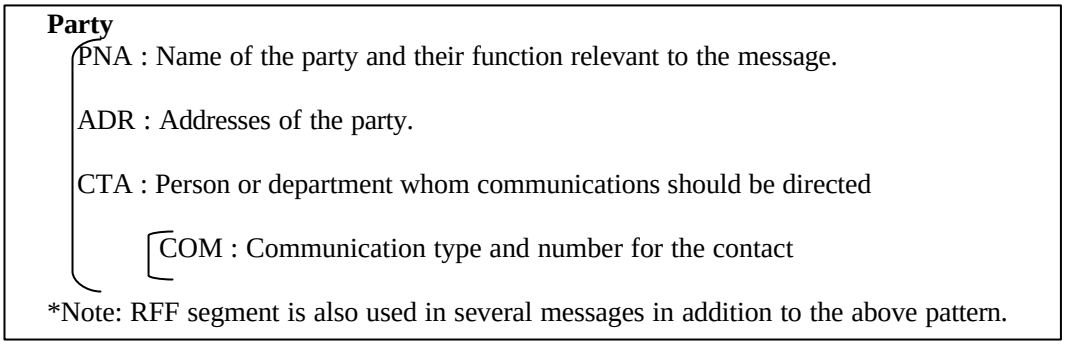


Fig.2 Party example – 2

The above examples of the pattern for 'Party' are representing the organizations involved in the

relevant business. Sometimes the one side of the business party may be an individual person. The single person case can be seen in the application of the medical industry, the life insurance industry or the labor market industry. The following sample shows the pattern in the message MEDREQ (Medical service request message)
.

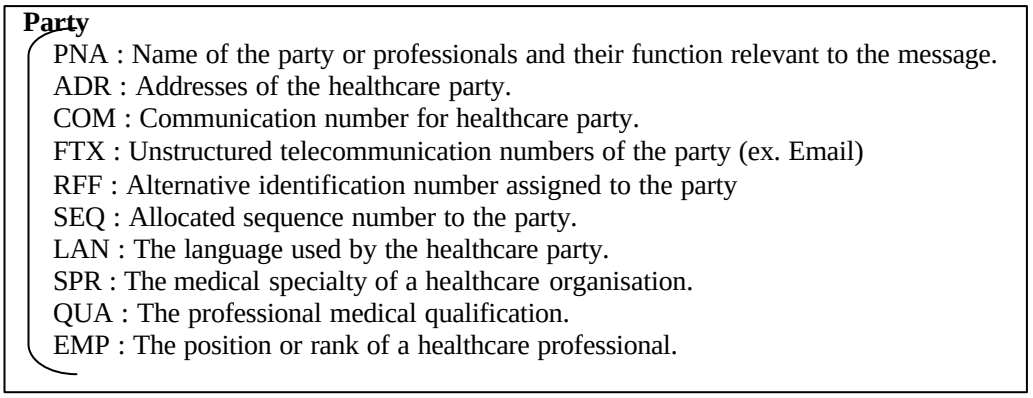


Fig.3 Party example – 3

In the above example, the party can be the organization or the individual professional person.
When the entity 'Party' is used for the individual, another entity 'Person' is recommended. Using the entity 'Person' avoids the complex pattern of the entity.

We should consider which pattern is efficient for being used in the e-business. There may be several measurements to decide which is better for ebXML standard.

Two measurements shall be considered as the first priority. The measurements are (1) 'Reusability in general applications' and (2) 'Usability in the specific application'. We should carefully model the business entity patterns considering the tradeoff between two of them. (see Section.4)

In above samples of the business entity 'Party', we can choose the second one (Fig.2) as one of the common business entities. We can choose it because it is used in various kinds of applications.

2.2 Requirement for defining the 'Party' entity within UN/EDIFACT

(1) Abstraction

The first segment of the 'Party' entity has a qualifier, which is specifying the function of the party.

The first data element 3055 specifies the function of the party. (see Fig.4)

The party without the qualification has no meaning in the real business. The party can be 'the seller', 'the buyer' or the other party who has the specific function in the relevant business. In another words, the entity 'Party' is an abstraction for many roles in the business. We can call the 'Party' as a super class and the party who has the specific function in the business as a sub-class in the object-oriented world. In the XML world the party would have meaning based upon the context of where the party element was included in the hierarchical model.

<Note 2> The discussion on how to describe the relations between classes can be found in the section 4.5.

PNA : Party identification
3035 Party function code qualifier
C082 Party identification details
3039 Party identifier
1131 Code list identification code
3055 Code list responsible agency code
*Note: Above structure is selected for the organizational party used in trading.

Fig.4 PNA segment structure

(2) Relation

The entity 'Party' is composed with the segments PNA(Party identification), ADR(Address), CTA(Contact information), COM(Communication contact) in UN/EDIFACT messages. PNA (Party identification) is used only in the entity 'Party', but others can be used in other relations of other entities. For example, ADR (Address) may be used for specifying the delivery address. Therefore, it is clever to distinguish the entity and the relation of the entities (the pattern). The entities and the patterns are the candidates for the core components of ebXML standards.

(3)Attributes

In the Fig.5, you can see several attributes for the entity 'Party'.

Party

Attribute-1	Party identification ----------→ a code value
Attribute-2	Address----------→ an entity ADR
Attribute-3	Contact----------→ an entity CTA

Fig.5 Attributes of Party

The first attribute 'Party identification' has a value that identify the party. The second and third attributes shall be specified through the other entities. The address of the party is specified by the attribute of the entity ADR. The telephone number or E-mail address for contacting the Party is specified through the entity CTA and COM.

(4) Representation

When the attributes get their values, the characteristics for them shall be defined. The identification of Party may be coded form, the address may be specified in Postal form, the telephone number may be numeric and the E-mail address may be character string. Any values of attribute shall have their certain representations.

(5) Value

The first data element (3039) of the composite data element C082 in Fig.4 shall have the value 'Party identifier'. The value of 'Party identifier' shall be specified in a code list that is defined by the following two data elements. There can be many code lists. Even the same value of the data element 3039 has deferent meaning in the deferent code list. These code lists are called value domains.

When the data element has the number representation, the value can be any arithmetical value. All the arithmetical values are one of the value domains.

When the data element has the calendar date, the value can be specified in the Gregorian calendar dates. All the calendar dates are another value domain.

3. Common Business Entity concept

Through the analysis of the business entities and the patterns in the previous section, the concept model for Common Business Entity can be described as follows.

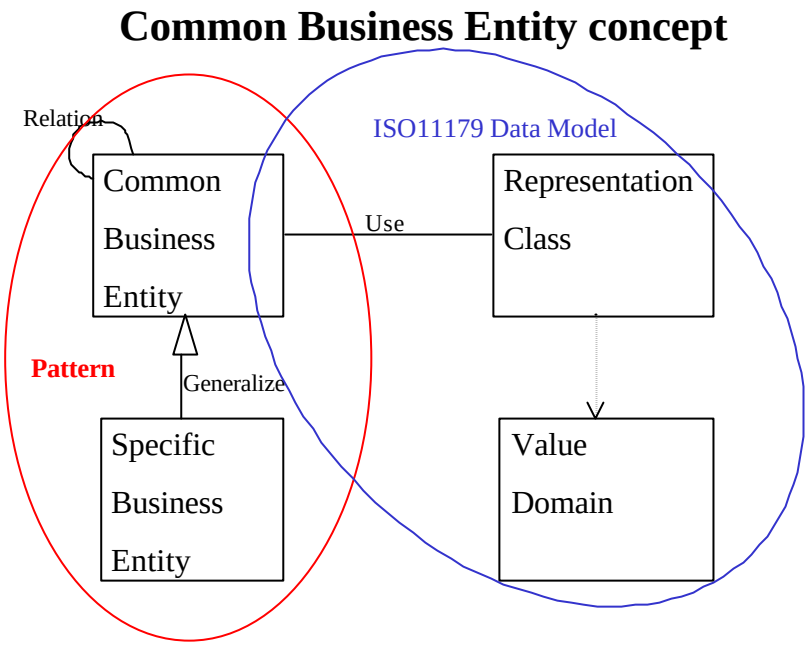


Fig.6 Common Business Entity concept

(1) Common Business Entity is a concrete class or an abstraction class generalized from one or more Specific Business Entities.
(2) A concrete class of Common Business Entity or a Specific Business Entity has one or more instances.
(3) Common Business Entity may have any kind of relations to other Common Business Entities. The series of relations related to a certain business behaviour is called Pattern.
(4) Common Business Entity shall have attributes those are related to other Common Business Entities or shall use Representation classes.
(5) Representation classes have properties. The properties for representation classes have a value domain, data type and, if necessary, a unit of measure or a character set.
(6) The concept (lexical meaning) and the format (syntactical expression) are the two schemas of a data type.
(7) The value domain is defined in the scope of certain concept (lexical meaning) with the certain format (syntactical expression).

The following items are kinds of Core Components.

(1) Analysis Pattern
(2) Common Business Entity
(3) Specific Business Entity
(4) Representation class
(5) Value domain

Analysis patterns may be used in the business process.
Analysis patterns, Common Business Entities and Representation classes shall be registered in a ebXML compliant repository.
The highly reusable Specific Business Entities shall be registered in a ebXML compliant repository. Other ones may be defined by each specific application for each specific industry.
The values in the highly reusable value domains shall be registered in a ebXML compliant repository. The application unique value domains, including code sets, may be defined by each specific application for each specific industry.

3.1 Representation Class candidates

The following lists are the candidates for Representation Class.

amount	A number of monetary units. It is normally associated with a type of currency.
code	A character string that represents a member of a set of values.
description	A series of sentences describing a person, object, place, event or concept.
identifier	A character string used to identify and distinguish uniquely, one instance of a value within an identification scheme.
name	A word or phrase that constitutes the distinctive designation of a person, object, place, event or concept. What the person, object, place, event or concept is known by or called.
number	An arithmetical expression representing a particular value. Note: This may often be used to imply sequence or a member of a series.
percent	A rate expressed in hundredths between values that have the same unit of measure.
quantity	A number of non-monetary units. It is normally associated with a unit of measurement.
rate	A quantity or amount measured with respect to another measured quantity or amount.
date or time	A date and/or time as measured in the time dimension.
age	A length of time that a person or thing has existed.
Indicator	An attribute indicating a condition such as on/off, true/fouls, yes/no, 0/1.
measure	A standard unit used to express size, amount or degree.(*)

(*) Another idea is to use ‘height’, ‘width’, ‘length’ ,’degree’ or ‘size’ instead of ‘measure’.

3.2 Core Component Meta model

The figure 7 shows the meta model for Core Component.

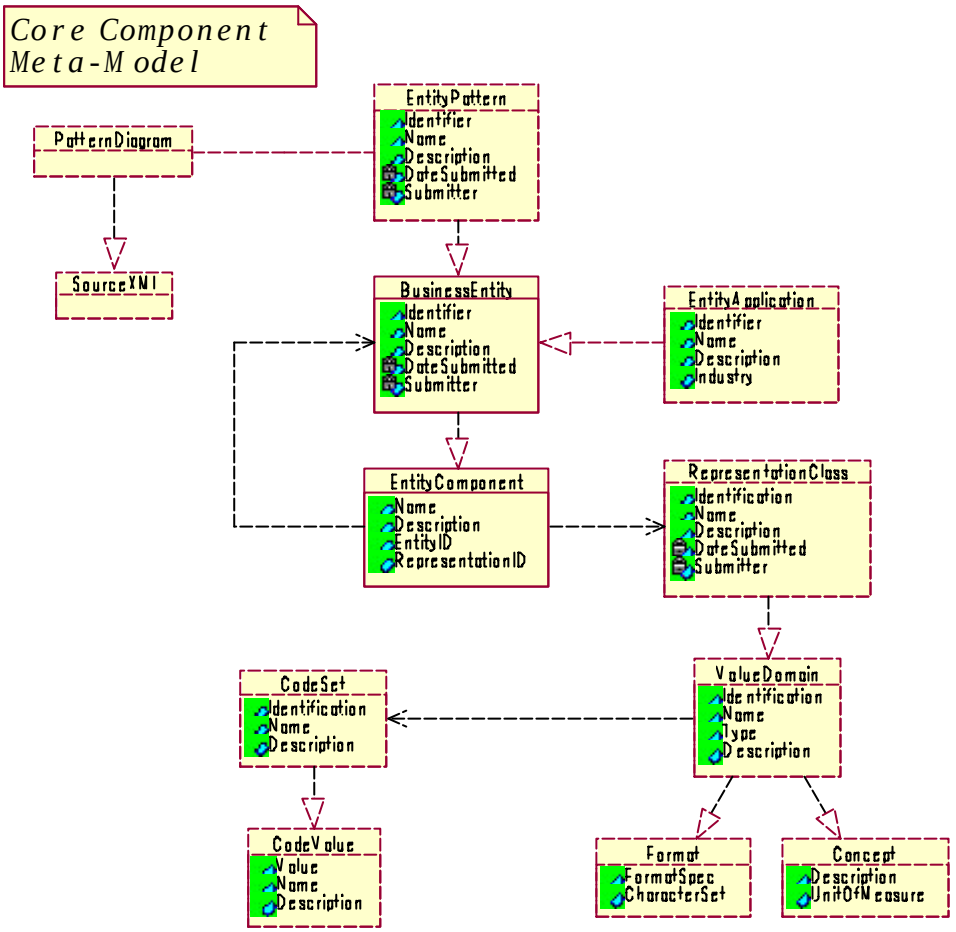


Fig 7 Core Component Meta model

4. Considerations

4.1 Reusability

The reusability of the core component is the key concern for efficiency of the e-business application software development. Reusing the standardized core components can push forward the interoperability between the e-business applications. On the other hand, having too much attachment to the widelevel usability in any application, you may recognize that only atomic level objects can be standardized, such as Numeric, Monetary amount, Gregorian calendar or Percentage. These atomic level objects are highly reusable. However, any other structured objects or object patterns have to be developed when you implement any e-business application. It makes it very difficult to implement the interoperability of applications.

When highly constructed objects are agreed upon for interchange, you can minimize your development efforts for implementation of the e-business application. The turnkey application packages or the fixed format EDI messages are easy to implement if all the parties involved are using the same platform (hardware) and the same application package (software). But it is impossible for a single uniformed application to be installed in all the enterprises in various industries. Therefore, the high level constructed components can be used only in certain applications in certain industries, or can be used only between certain trading partners.

Because of the above considerations, we shall select and standardize the proper level of

components.

4.2 Syntactically neutral

It is true that no model can be described without any syntax. This may be one of FDTs (Formal Descriptive Technique), natural languages or graphical charts. The meaning of a syntactically neutral model is a model free from an implementation level syntax, such as EDIFACT, XML or JAVA. In other words, the model described in a syntactically neutral manner can be implemented in the computer system using any proper implementation level syntax. Also the syntactically neutral model described by the certain FDT for modeling can be mapped to another model using another FDT.
UML (Unified Modeling Language) may be one of the FDTs used for the syntactical neutral modeling.

4.3 Basic Semantic Register (BSR)

According to the definition of TC154-BSR project, BSC (BSR Semantic Component) is a generic term comprising the components of BSR semantic units. In the context of the BSR there are two types of BSR semantic component, representation class and concept**.** The concept of BSC is almost same as Common Business Entity of Core Components, and the representation class of BSR is same as Representation class of Core Components. When Common Business Entities and Representation classes are selected and specified, the work done by TC154-BSR project may be referred.
BSU (BSR Semantic Unit) is concept unambiguously defined, independently of any particular physical representation, and which is semantically complete. It is independent of the process or application in which it is used. It is constructed using BSR semantic components**.** But BSU is a completely deferent approach from Core Component analyzed in this paper. BSUs represent attributes of Specific Business Entity. In the concept of Common Business Entity, attributes are defined in Common Business Entity and Specific Business Entities inherit the attributes of Common Business Entity.

4.4 Naming the component

There are three purposes of naming the component.

1. To identify the component uniquely in the certain domain if there are no identifiers other than the name.
2. To be recognized easily by human.
3. To specify the domain structure.

The data element conforming ISO11179 shall have the unique identifier other than the name, therefore, the name of the data element is the primary means of identification of objects and concepts for humans. Otherwise, the name is the only identification for the component in UML. Also the element name is the only identification for the component in XML. However, an element name in XML may have different definitions depending upon the context of the element within the XML structure.
We need some identification methodology for naming core components even when we start to analyze Business Entities. At the analysis phase, the identifiers should be recognized easily by both humans and computers.

4.5 Segmentation of Core Components

This consideration looks at how it might be possible to use some of the less commonly used features of UML to create sets of core components that can be reused in multiple contexts.
In the paper on *Transformation from EDIFACT to XML* Pharos group members from the EDIFACT Transport group suggest that multiple associations should be used to identify the different “roles” a particular set of data elements play. They state that:
“A role name defines a task or duty of a class in an association with another class.” The example they use is the qualifier of the Party segment that indicates whether the party concerned is the Buyer, Seller, Consignor, Dispatch Party, etc.
The Pharos document also contains a number of “rules” for the creation of UML models to represent

business messages. Rule 5 states:

> **"Pharos Rule 5:** Create multiple associations
>
> Multiple associations are created for message classes that include subsections of the role list category.

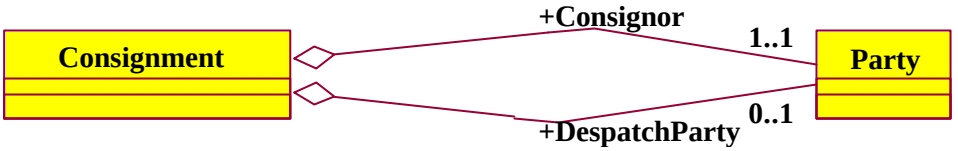


Fig. 8 Example of multiple associations.

> Each of the multiple associations is given a multiplicity and a role name. The role names are found as data term names in subsections of the role list category.
> The role name is used in the generated XML/DTD or XML Schema as a 'group' name for the attributes in the child message class structure. The UML notation uses a '+' before the role name to indicate that the role name is 'public', but this symbol will be stripped in the generated XML/DTD or XML Schema.
> The example multiple associations in the figure can be read as:
> A Consignment has minimum 1 and maximum 1 Consignor
> A Consignment has minimum 0 and maximum 1 DispatchParty"

Whilst the use of associations to distinguish between the roles played by multiple occurrences of a class that can occur more than once within a document is highly commendable, it cannot be agreed upon with the group name assigned within the context. It is difficult to determine programmatically the context for a class within the XML DTD or Schema. A class with multiple roles should be represented by an abstract class in the UML model, with specific instances of the class being identified in the XML DTD or Schema by means of the name that associates the abstract class with the message.
The section 2.2 (1) takes a more traditional EDI approach to the purpose of these "qualifiers" of data element groups. It states:
The idea that Party is a super class suggested here might be better mapped to the concept of an abstract class, because there is no real sense of inheritance needed in this example
The comment on this concept was that "Assigning a 'role' qualifier to a party specification means that the information therein cannot be easily reused as part of an alternative role. If the role of the party is defined by its context, i.e. by its parentage/container within an XML tree, then reusability of Party-related information will be easier to manage. Unfortunately XML trees do not work in the same manner as traditional OO classes. We need to be able to identify things like Order\Buyer\PartyName and Order\Seller\PartyName rather than Order\Parties\Buyer and Order\Parties\Seller, which is what is implied if the party role defines a sub-class of Party. (Note the fact that the XML container needs to be Parties, not Party, if you use a class-based model, as the container is intended to contain information about all parties in current context.)"
In trying to reconcile the views, the sequence of containment can be expressed as:

Order
    Parties
        Seller
            Party Information
                Party Identification
                ….

In discussions of the UK Data Harmonization Group at e-centre[uk] it was pointed out that the Parties and Party Information component of this sequence were really just containers that allowed the correct management of data that forms part of the same abstract class. The other three components of the "tree" represent the Business Process being undertaken (Order), the association of the abstract class containing party information with the business process (Seller), and the core components that identify the information to be interchanged for the completion of that part of the business process (Party Identification …).

The 'real abstract class' in this example is the Party Information abstract class. This may need to take a number of different forms within different messages. For example, in the EB-Simpl model most of the occurrences of data elements making up the Party Information are pre-exchanged, in order that messages used by business processes only need to pass a key of the pre-exchanged information (the Party Identification code). In other messages a wider range of the fields from the Party Information set needs to be interchanged.

5. Describe the Core Component

Each business entity pattern can be described as the class diagram with the definitions of the involved entities. But highly reusable representation classes shall be defined independent from any business entities using them.

(1) Describing the business entity pattern

The set of the documents describing the business entity pattern is as follows.

- Entity pattern definition (One cover sheet)
  - Entity patter ID
  - Entity pattern name
  - Entity pattern description
  - Class diagram representing the entity pattern
- Entity definitions (One sheet for each related entity)
  - Entity ID
  - Entity name
  - Entity description
  - Attribute list
    - Attribute ID
    - Attribute name
    - Attribute type (Designating entity or Using representation)
    - Reference identification for designating entity or using representation
    - Parameters for using representation class
    - Attribute description
  - Sub-class list
    - Sub-class ID
    - Sub-class name
    - Sub-class function

(2) Describing representation class

Each representation class can be described in one sheet.

- Representation class definition
  - Representation class ID
  - Representation class name
  - Representation class description
  - Data type definitions
    - Concept (lexical meanings)
      - Concept ID
      - Concept specification
      - Unit of Measure if needed

Format (syntactical expressions)
Format ID
Format specification
Character set if needed
Value domain list
Value domain ID
Value domain name
Value specification in the domain

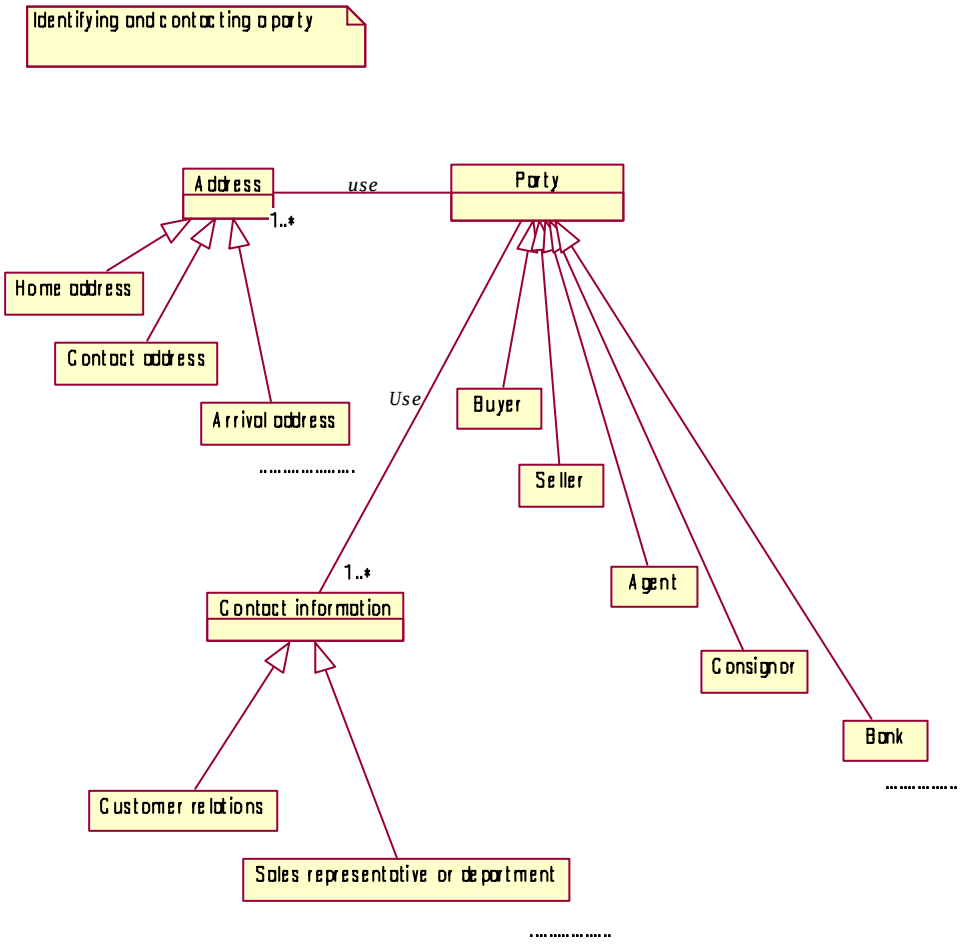
6. Sample description

Pattern Definition

Date: 24/Mar/2000
Name: H. Sugamata

Pattern ID : SAMPLE-P-01
Pattern Name : Identification and contacting a party
Description : To identify the party, who is involved in the business process, with their contacts.

Class diagram



*Note: If there can be too many Sub-classes, Package may be used.

Business Entity Definition

Date: 24/Mar/2000
Name: H. Sugamata

Entity ID: SAMPLE-E-01
Entity Name: Party
Description: Unique framework of authority designating to act toward some purpose in the business.

Attribute List

Attribute name	Type	Ref.ID	Description
Party identifier (UN: 7402)	R	SAMPLE-R-01 (C1, F1, Vn)	Identification of the party.
Address	E	SAMPLE-E-02	Address of the party.
Contact information	E	SAMPLE-E-03	To identify a person or a department of the party, to whom communication should be directed.

*Note1: Type is R (using Representation class) or E (designating Business Entity).
*Note2: 3 Parameters shall be specified with Representation class in Ref.ID field.
Cn : Parameter of the Ref.ID for Concept
Fn : Parameter of the Ref.ID for Format
Vn : Parameter of the Ref.ID for Value domain

Sub-Classes

Sub-Class Name	Function
Buyer	Party to whom merchandise and/or service is sold.
Seller	Party selling merchandise to a buyer.
……...etc	
	The sub-classes of Party are defined in the code list of the data element 3055 (Party function code qualifier) in UN/EDIFACT directory. There are 498 functions for Party in D.00A.

*Notes: If there are too many sub-classes, you can designate the relevant code set.

Business Entity Definition

Date: 24/Mar/2000
Name: H. Sugamata

Entity ID: SAMPLE-E-02
Entity Name: Address
Description: To specify an address.

Attribute List

Attribute name	Type	Ref.ID	Description
Address type code (UN:3131)	R	SAMPLE-R-02 (C1, F1, V1)	Code specifying the type of an address.
Address status code (UN:3475)	R	SAMPLE-R-02 (C2, F1, V2)	Code specifying the status of an address.
Address component description (UN:3286)	R	SAMPLE-R-03 (C1, Fn, Vn)	Free form description of the component of an address.
City name (UN:3164)	R	SAMPLE-R-04 (C1, F1, V1)	Name of a city.
Postal identification Code (UN:3251)	R	SAMPLE-R-02 (C3, F2, V3)	Code specifying the postal zone or address.
Country name code (UN:3207)	R	SAMPLE-R-02 (C4, F3, V4)	Identification of the name of the country
Country sub-entity name (UN:3228)	R	SAMPLE-R-04 (C2, F1, V2)	Name of a country sub-entity.
Location name code (UN:3225)	R	SAMPLE-R-02 (C5, F4, V5)	Code specifying the name of the location which is defined in UNLOCODE.

*Note1: Type is R (using Representation class) or E (designating Business Entity).
*Note2: 3 Parameters shall be specified with Representation class in Ref.ID field.
Cn : Parameter of the Ref.ID for Concept
Fn : Parameter of the Ref.ID for Format
Vn : Parameter of the Ref.ID for Value domain

Sub-Classes

Sub-Class Name	Function
Home address	The address is the home address.
Contact address	Address where contact may be made.
Arrival address	Address of arrival.
……...etc	The sub-classes of Address are defined in the code list of the data element 3299 (Address purpose code) in UN/EDIFACT directory. There are 7 functions for Address in D.00A.

*Notes: If there are too many sub-classes, you can designate the relevant code set.

Business Entity Definition

Date: 24/Mar/2000
Name: H. Sugamata

Entity ID: SAMPLE-E-03
Entity Name: Contact information
Description: To identify how to contact a person or a department to whom communication should be directed.

Attribute List

Attribute name	Type	Ref.ID	Description
Communication address (UN:3148)	R	SAMPLE-R-05 (C1, F1, Vn)	A communication address of a department or a person to whom communication should be directed.
Department or employee name code (UN:3413)	R	SAMPLE-R-02 (C6, F2, V6)	Code specifying the name of a department or employee.
Department or employee name (UN:3412)	R	SAMPLE-R-04 (C3, F1, V3)	Name of a department or employee .

*Note1: Type is R (using Representation class) or E (designating Business Entity).
*Note2: 3 Parameters shall be specified with Representation class in Ref.ID field.
Cn : Parameter of the Ref.ID for Concept
Fn : Parameter of the Ref.ID for Format
Vn : Parameter of the Ref.ID for Value domain

Sub-Classes

Sub-Class Name	Function
Customer relations	Individual responsible for customer relations.
Sales representative or department	The sales representative or department contact within an organization.
…….. etc	
	The sub-classes of Contact are defined in the code list of the data element 3139 (Contact function code) in UN/EDIFACT directory. There are 95 functions for Contact in D.00A.

*Notes: If there are too many sub-classes, you can designate the relevant code set.

Representation Class Definition

Date: 29/Mar/2000
Name: Hisanao Sugamata

Representation ID :SAMPLE-R-01
Representation Name : Identifier
Description: A character string used to identify and distinguish uniquely, one instance of a value within an identification scheme.

Data type

Concept (lexical meanings)

Specification	Unit of Measure
1 Party identifier	
2	
3	

Format (syntactical expressions)

Specification	Character set
1 An..35	
2	
3 * The formats of identifiers are defined in each schema.	

Value domain list

Domain name	Specification
1 Duns	Dun & Bradstreet Corporation assigned identifier
2 S.W.I.F.T.	S.W.I.F.T. assigned identifier
3 ……etc	
*The identifier schemas are defined in the code list of data element 1131/3055 in UN/EDIFACT directory.	

Representation Class Definition

Date: 29/Mar/2000
Name: Hisanao Sugamata

Representation ID :SAMPLE-R-02
Representation Name : Code
Description A character string that represents a member of a set of values.

Data type

Concept (lexical meanings)

Specification	Unit of Measure
1 Address type code	
2 Address status code	
3 Postal identification code	
4 Country name code	
5 Location name code	
6 Department or employee name code	
7 ……. Etc	

Format (syntactical expressions)

Specification	Character set
1 an..3	
2 an..17	
3 a2	
4 an..25	

Value domain list

Domain name	Specification
1 The code list for types of Address.	Refer DE3131 of UN/EDIFACT
2 The code list for status of address	Refer DE3475 of UN/EDIFACT
3 The code list for postal identification	The code lists are assigned by the country authority
4 The code list for countries name	Codes specified in ISO3166
5 The code list for locations	UNLOCODE specified UN/ECE recommendation 16
6 The code list for departments or employees	Code specified by organisation concerned

Representation Class Definition

Date: 29/Mar/2000
Name: Hisanao Sugamata

Representation ID :SAMPLE-R-03
Representation Name : Address component description
Description A description of the component of an address.

Data type

Concept (lexical meanings)

Specification	Unit of Measure
1 Address component description	
2	

Format (syntactical expressions)

Specification	Character set
1 Address format : Street name followed by number 2 Address format: Number, road type, road name in this sequence 3 Address format: Road type, road name, number in this sequence 4 Address format: Post office box 5 Address format: Unstructured address 6 Address format: Street name followed by number, building, suite 7 Address format: Rural route number 8 Address format: Post office drawer number 9 Address format: Building name followed by suite Note:1 Address formats are specified in the code list of the data element 3477 in UN/EDIFACT	

Value domain list

Domain name	Specification
1 Address component Description -1	Street name followed by number
2 Address component Description -2	Number, road type, road name in this sequence
3 Address component Description -3	Road type, road name, number in this sequence
4 Address component Description -4	Post office box
5 Address component Description -5	Post office box
6 Address component Description -6	Street name followed by number, building, suite
7 Address component Description -7	Rural route number
8 Address component Description -8	Post office drawer number
9 Address component Description -9	Building name followed by suite

Representation Class Definition

Date: 29/Mar/2000
Name: Hisanao Sugamata

Representation ID :SAMPLE-R-04
Representation Name : Name
Description: A word or phrase that constitutes the distinctive designation of a person, object, place, event or concept. What the person, object, place, event or concept is known by or called.

Data type

Concept (lexical meanings)

Specification	Unit of Measure
1 City name	
2 Country sub-entity name	
3 Department or employee name	

Format (syntactical expressions)

Specification	Character set
1 an..35	
2	
3	

Value domain list

Domain name	Specification
1 City name	Name of a city in alphabetic characters
2 Country sub-entity name	Country sub-entity name in alphabetic characters
3 Department or employee name	Name of a department or employee in alphabetic characters

*Notes: If there are too many domains, you can designate the relevant code set.

Representation Class Definition

Date: 29/Mar/2000
Name: Hisanao Sugamata

Representation ID :SAMPLE-R-05
Representation Name : Communication contact address
Description: A communication contact address of a department or a person to whom communication should be directed.

Data type

Concept (lexical meanings)

Specification	Unit of Measure
1 Communication address	
2	
3	

Format (syntactical expressions)

Specification	Character set
1 an..512	
2	
3	

Value domain list

Domain name	Specification
1 International telephone	Telephone number, including country and/or city code as required, for voice or data transmission by telephone beyond the border of a country.
2 World Wide Web	Data exchange via the World Wide Web.
3 Electronic mail	Exchange of mail by electronic means.
……..	
Note:1 The value domain is defined in the data element 3155 in UN/EDIFACT. There are 34 value domains specified in D.00A.	

7. Instructions for capturing ebXML Core Component definitions

The following instructions describe how Microsoft's Internet Explorer 5.0 (IE5) can be used to capture core component definitions as XML files.

WARNING: This process will not work on Netscape Explorer or on older versions of Internet Explorer as it relies on Microsoft specific extensions to HTML.

The UML metamodel for ebXML core components recognizes the following types of record:

- Patterns: UML models that define a set of related data entities, and the associations between them
- Entities: UML model components that identify sets of related components, including data elements with a specific data representation
- Date Representations: the set of data patterns or code lists that can be used to capture a particular component of an entity
- Code Sets: A list of permitted values that can be used to complete a component, together with descriptions of their meaning
- Data Formats: Details of constraints to be placed on the contents particular components of a pattern.

The HTML forms provided in this suite allow each of these data types to be recorded. To allow the relationships between forms to be clearly identified the basic UML model has been extended by adding elements that reference the ID of the next lowest level of component in the model. (This means that once a component, data representation, code list or data format has been defined once it need only be referenced in subsequent models.)

For each of the above record types there is an form with the appropriate name (pattern.htm, entity.htm, representation.htm, CodeSet.htm and DataFormat.htm). Each form contains buttons that allow the current contents to be submitted for storage, reset so that a new entry of the same type can be made or request a form for a record at a lower level in the metamodel.

The package has been designed to be run within a directory called ebxml within the My Documents section of your C drive. If wish to use another drive or root directory you should search each of the above files for any occurrences of the string "C:/My Documents/ebxml/" and replace this with the appropriate identifier for the directory you wish to use.

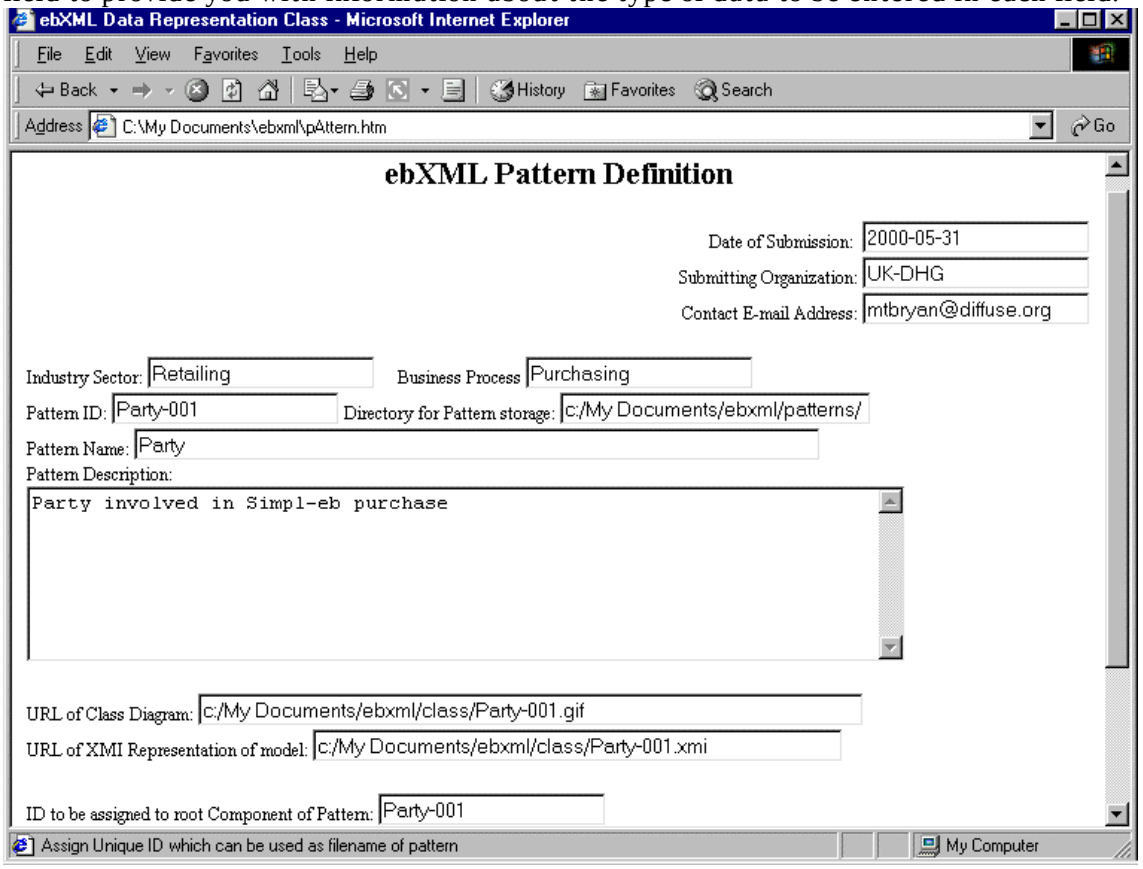
Within the directory that is used to contain the forms you will need to create five subdirectories prior to using the forms. These subdirectories should be labelled patterns, entities, representations, codesets and dataformats respectively. A further directory, called class, can also be defined to store class diagrams, etc. (Directories with these names may or may not have been created for you when you unzipped the source files, depending on your settings of your file unzipper.)

Because the forms submit their contents to the local file store rather than to a remote directory you will need to ensure that IE5 has been set up to permit this. In the Tools Menu select the Internet Options entry and then the Security tab. If your security level is High you will not be able to use these forms. If it is set to Medium you will be asked

to confirm that each record may be written to disc. If it is Low you will be able to write the files without having to confirm each one, but must take care if using the Internet. (I find that the Medium level, which requires me to confirm before writing but still provides for a safe level of Internet access is an adequate compromise.)

7.1 Defining a Pattern

The following figure shows the fields used to define a new pattern for use as a core component. The status line at the foot of the form will change as you move from field to field to provide you with information about the type of data to be entered in each field.

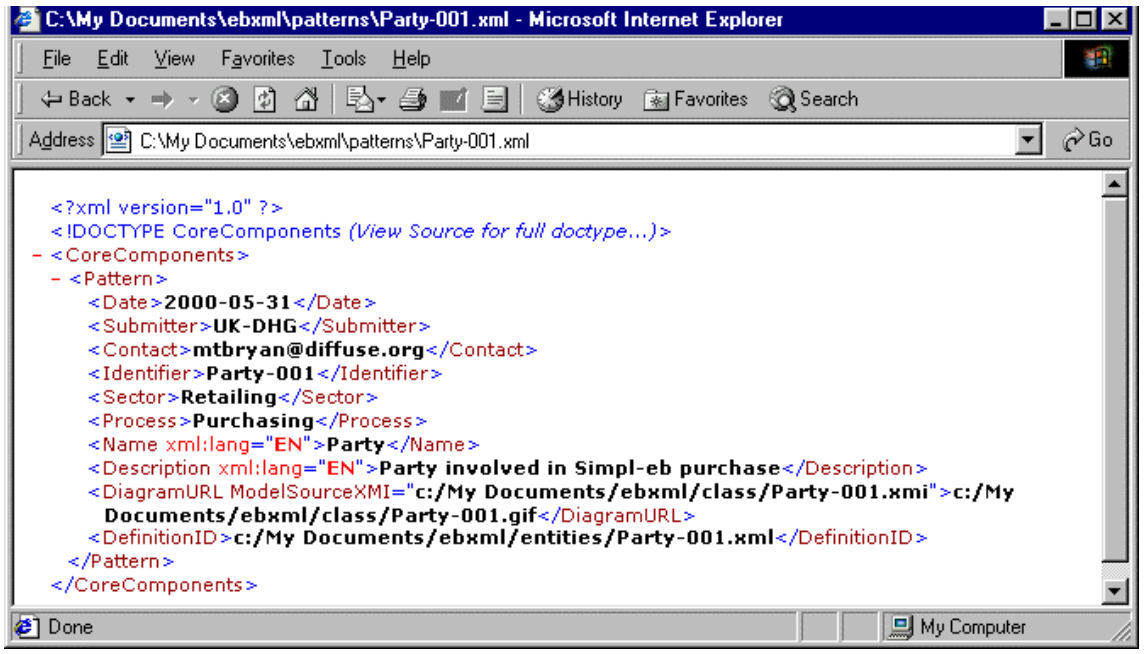


The fields provided are:

- Date of Submission: enter an ISO 8601 conformant date using the format CCYY-MM-DD here
- Submitting Organization: enter the name of the submitting organization here Contact E-mail Address: enter e-mail address to which questions can be submitted
- Industry Sector: identify which communities the pattern is expected to be used by (using SIC codes where available)
- Business Process: record business process(es) to which pattern applies (using SIC codes where available)
- Pattern ID: Enter a unique identifier for the pattern. (This will form the file name of the pattern definition.)
- Directory for pattern storage: Should indicate the path required to reach the required directory. (When the default directory set up is being used this entry should not need to be updated from the pre-assigned values.)

- Pattern Name: Name to be used to identify the pattern. (May or may not be the same as the Pattern ID.)
- Pattern Description: Enter description of role of pattern that allows it to be distinguished from other patterns.
- URL of Class Diagram: Enter URL of file containing printable version of class diagram. (For ease of use on the web we recommend this be a GIF file. If this is stored in the class subdirectory within you ebxml directory then all you need to do is to replace the ??? in the default name displayed with the filename.)
- URL of XMI Representation of model: If an XMI representation of the model is available for interchange enter the appropriate file reference here: otherwise delete the default value.
- ID to be assigned to root Component of Pattern: Indicate which ID you expect to assign to the Entity which will form the root of the pattern
- Directory that will be used for Entity storage: Should indicate the path required to reach the required directory. (When the default directory set up is being used this entry should not need to be updated from the pre-assigned values.)

When all relevant fields have been completed click on the Submit this Pattern button. This will cause the inbuilt program to store an XML record pattern and then display the contents of the file as shown below.

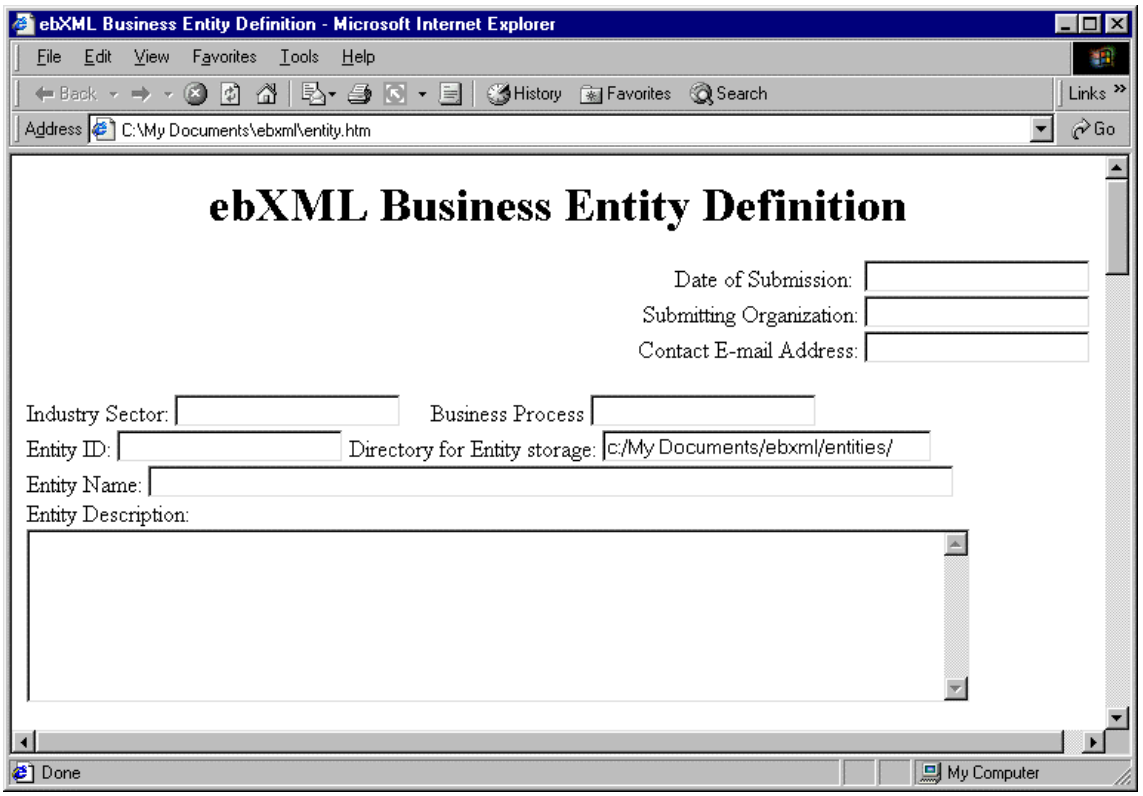


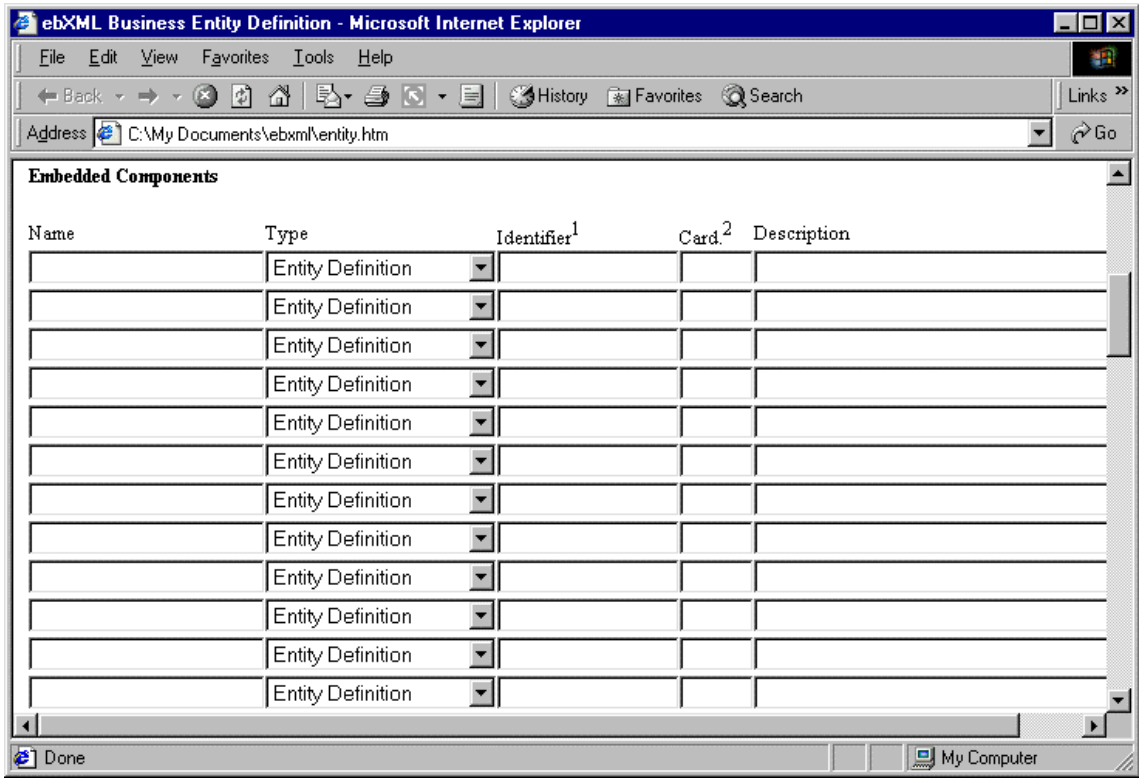
Once you have ascertained that this file correctly records the details of your pattern the window can be dismissed from the screen.

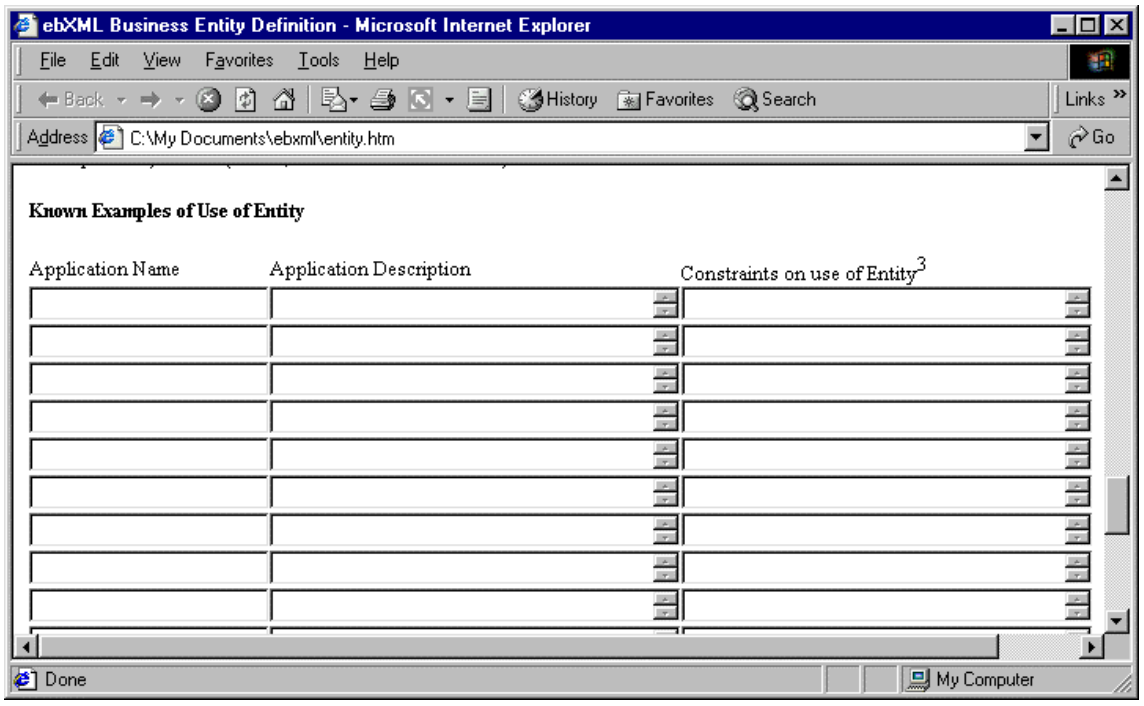
The Pattern form ends with a button that allows you to "Create Entity Definition for Root Component". Clicking on this button will call up the form needed to record each of the entities defined within the pattern.

7.2 Defining an Entity

The following figures show the fields used to define a new entity for use within a core component. The status line at the foot of the form will change as you move from field to field to provide you with information about the type of data to be entered in each field.







The fields provided in the first part of the form are:

- Date of Submission: enter an ISO 8601 conformant date using the format CCYY-MM-DD here
- Submitting Organization: enter the name of the submitting organization here

- Contact E-mail Address: enter e-mail address to which questions can be submitted
- Industry Sector: identify which communities the pattern is expected to be used by (using SIC codes where available)
- Business Process: record business process(es) to which pattern applies (using SIC codes where available)
- Entity ID: Enter a unique identifier for the entity. (This will form the file name of the entity definition.)
- Directory for entity storage: Should indicate the path required to reach the required directory. (When the default directory set up is being used this entry should not need to be updated from the pre-assigned values.)
- Entity Name: Name to be used to identify the entity. (May or may not be the same as the Entity ID.)
- Entity Description: Enter description of role of entity that allows it to be distinguished from other core components.

Each line in the section headed Embedded Components has the following fields:

- Name: Name of component in form suitable for use in XML DTD (i.e. starting with a letter and containing no spaces)
- Type: Type of component (either Entity Definition if there are embedded components or Data Representation if this component is a root one designed to transfer data between systems)
- Identifier: Indicate which ID you expect to assign to the Entity/Representation when you define it (this will create a cross reference to the definition you will create at a subsequent stage using either another copy of this form or the form for recording data representations.)
- Card. If the embedded component is optional and/or repeatable the cardinality of this component with respect to its parent (i.e. how many times it can occur within the parent) as defined in the UML model should be recorded here, expressed as 0..1 (optional), 0..* (optional and repeatable), 1..* (required and repeatable) or m..n (m=min, n=max no of occurrences).
- Description: Brief description of role of component within Entity.

If the entity is a pattern that is intended to be sub-classed or used as an abstract class that is associated with concrete classes, examples of the intended use of the entity can be recorded in the section headed Known Examples of Use of Entity. The following fields can be used for each example:

- Application Name: Name by which sub-class or association is known
- Application Description: Description of purpose of application
- Constraints on use of Entity: Details of any constraints that apply to the use of the entity within this application (e.g. components that must or may not be used for this application of the pattern/entity.)

Note: At present there is no formal language for defining such constraints but in future it is anticipated that text based descriptions entered initially will be replaced by machine processable XML descriptions of the required constraints at a later date.

The buttons at the foot of the Entity Definition form are:

- Submit this Entity:
- New Entity Definition: Resets the form so that details of another component can be defined. (Each embedded component defined using

Entity Description in the Type field requires completion of a separate Entity Description form.)

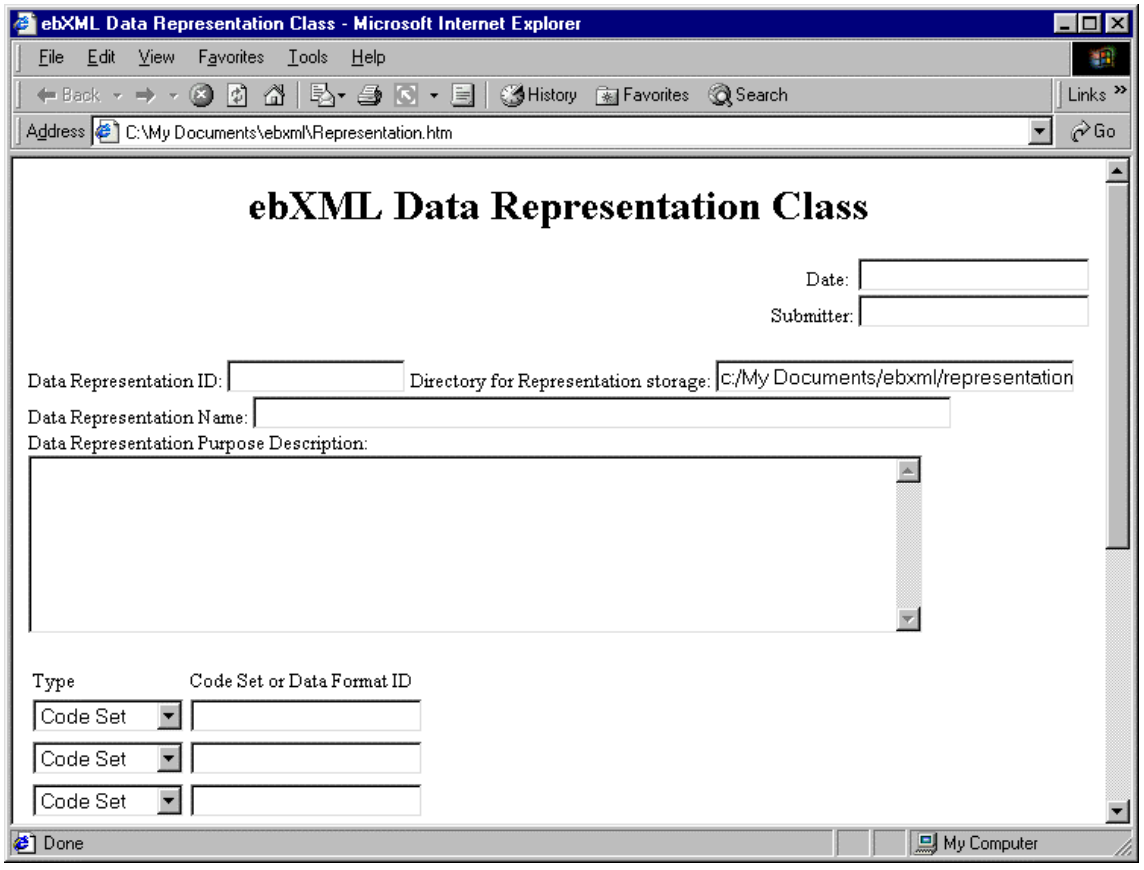
- Create Data Representation Definition: Calls up the form needed to record a Data Representation. (Each embedded component defined using Data Representation in the Type field requires completion of a Data Representation form if the identified format has not previously been defined.)

7.3 Defining a Data Representation

A data representation identifies one or more data formats and/or code sets that can be used to record a particular type of data stored within a core component.

Note: While normally a data representation will only define data of the same type (e.g. one or more data formats or one or more code lists) there are cases where both a code list and a data format will be required (e.g. to define a list of known codes plus a pattern that can be used to extend the list where appropriate.)

The following figure shows the fields used to define a new data representation for use within a core component. The status line at the foot of the form will change as you move from field to field to provide you with information about the type of data to be entered in each field.



The fields provided on this form are:

- Date of Submission: enter an ISO 8601 conformant date using the format

CCYY-MM-DD here

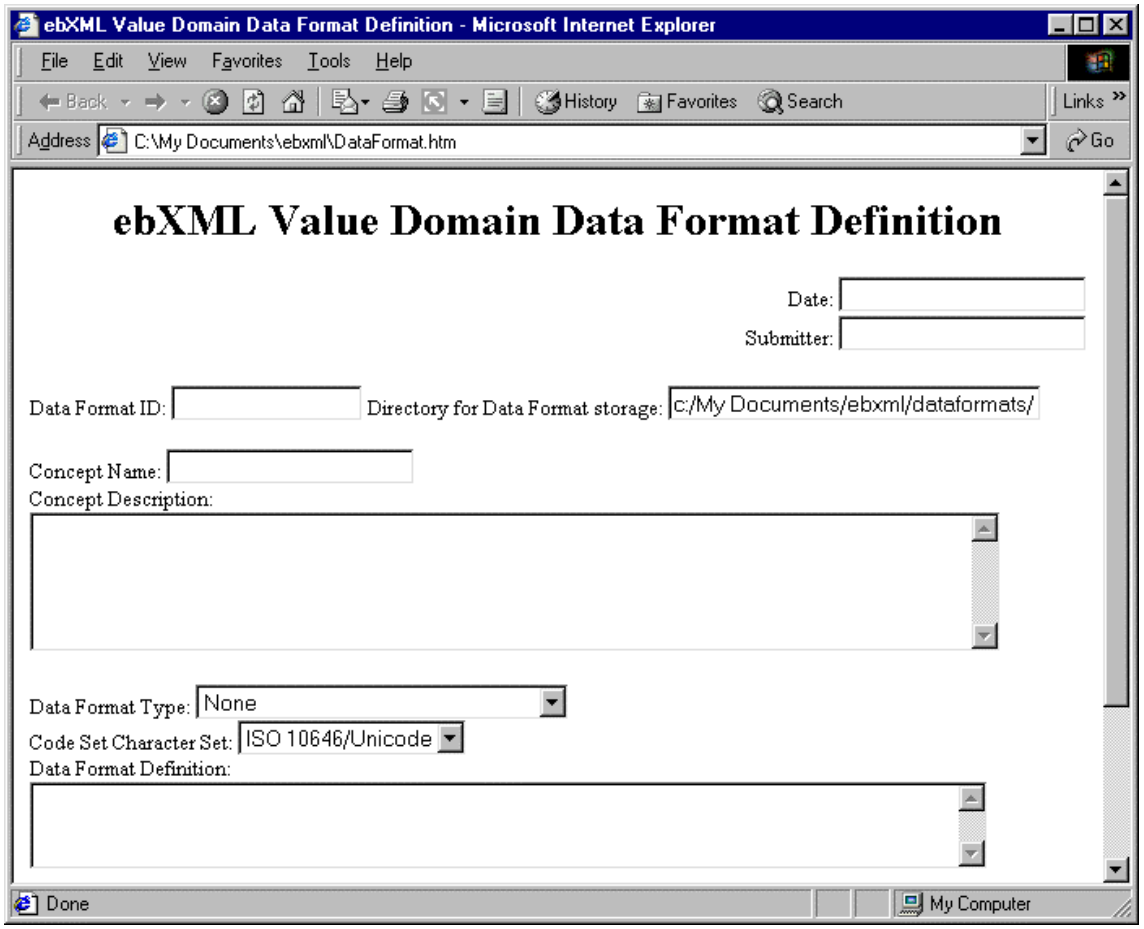
- Submitting Organization: enter the name of the submitting organization here
- Contact E-mail Address: enter e-mail address to which questions can be submitted
- Date Representation ID: Enter a unique identifier for the data representation. (This will form the file name of the data representation definition.)
- Directory for Representation storage: Should indicate the path required to reach the required directory. (When the default directory set up is being used this entry should not need to be updated from the pre-assigned values.)
- Data Representation Name: Name to be used to identify the data representation. (May or may not be the same as the Date Representation ID.)
- Data Representation Description: Enter description of role of data representation that allows it to be distinguished from other representations.
- Type: Select Code Set if the representation is to reference a code set, or Data Format if it is to reference a data format
- Code Set or Data Format ID: Enter the unique identifier to be assigned to the referenced definition.

The form ends with the following buttons:

- Submit this Representation: Creates XML record of form and displays this in a separate window. (Dismiss window if record is accurate: otherwise return to source and use the Back button to return to your entries so that they can be corrected and resubmitted.)
- Define another Representation: Resets the form so that details of another representation can be defined.
- Define Code Set: Calls up the form needed to record a Code Set. (Each format defined using Code Set in the Type field requires completion of a Domain Value Code Set form if the identified format has not previously been defined.)
- Define Data Format: Calls up the form needed to record a Data Format. (Each format defined using Data Format in the Type field requires completion of a Domain Value Data Foramt form if the identified format has not previously been defined.)

7.4 Defining a Data Format

The following figure shows the fields used to define a code set for use within a core component representation. The status line at the foot of the form will change as you move from field to field to provide you with information about the type of data to be entered in each field.



The fields provided on this form are:

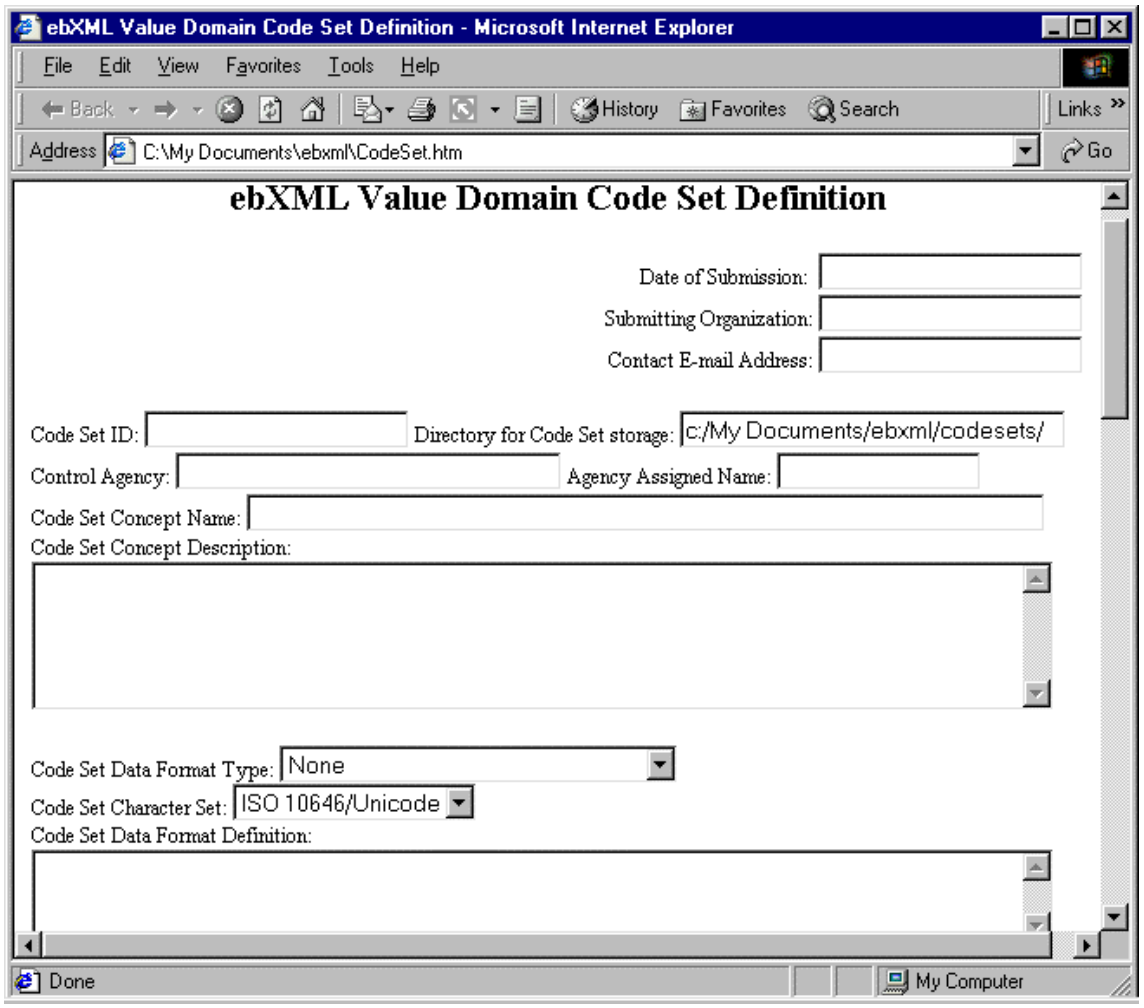
- Date of Submission: enter an ISO 8601 conformant date using the format CCYY-MM-DD here
- Submitting Organization: enter the name of the submitting organization here
- Contact E-mail Address: enter e-mail address to which questions can be submitted
- Data Format ID: Enter a unique identifier for the data format. (This will form the file name of the data format definition.)
- Directory for Data Format Set storage: Should indicate the path required to reach the required directory. (When the default directory set up is being used this entry should not need to be updated from the pre-assigned values.)
- Data Format Concept Name: Enter name to be used to identify the concept behind code list. (May or may not be the same as the Data Format ID.)
- Data Format Concept Description: Enter description of role of data format that allows it to be distinguished from other data formats.
- Data Format Type: If code set has pattern based on one of the known data format languages select relevant entry from list. Otherwise select None
- Character Set: If relevant, select controlling character set from list supplied.
- Data Format Definition: If relevant, enter pattern that defines format of codes set values using language identified in Code Set Data Format field

The form ends with the following buttons:

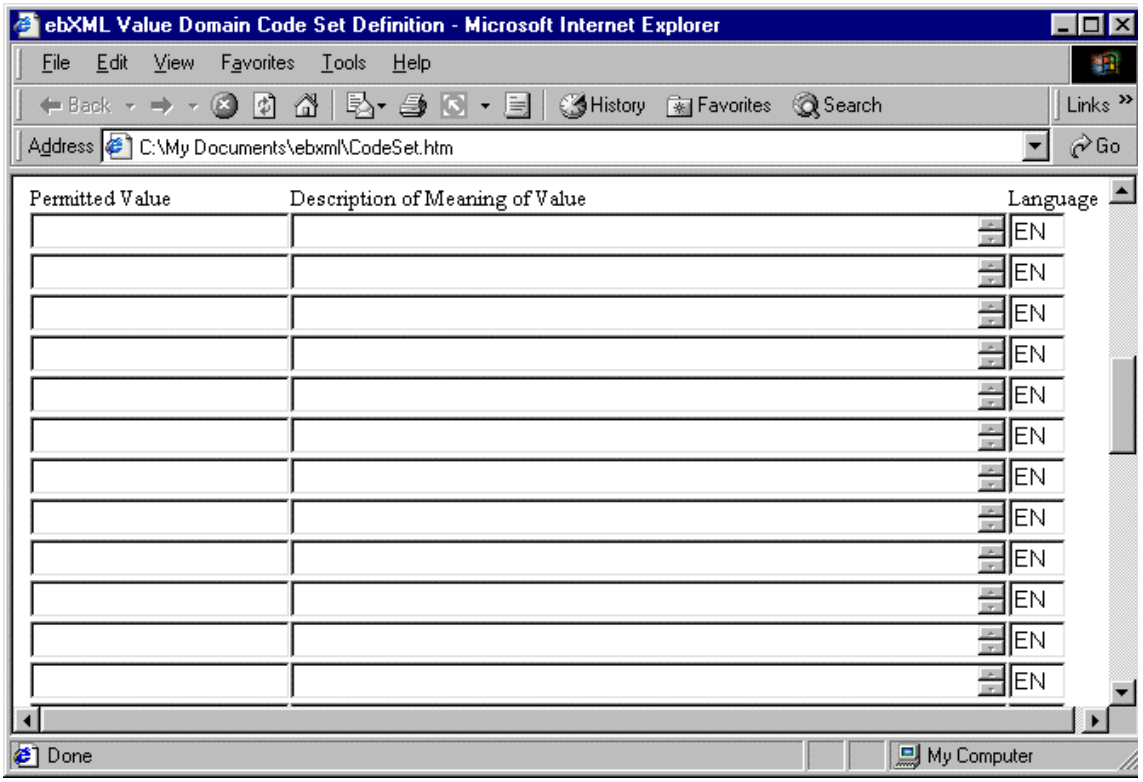
- Submit this Data Format: Creates XML record of form and displays this in a separate window. (Dismiss window if record is accurate: otherwise return to source and use the Back button to return to your entries so that they can be corrected and resubmitted.)
- Define another Data Format: Resets the form so that details of another code set can be defined.
- Define Code Set: Calls up the form needed to define a Code Set.
- New Entity Definition: Calls up the form needed to define an Entity.
- Create Data Representation Definition Calls up the form needed to define a Data Representation.

7.5 Defining a Code Set

The following figures show the fields used to define a code set for use within a core component representation. The status line at the foot of the form will change as you move from field to field to provide you with information about the type of data to be entered in each field.


ebXML Value Domain Code Set Definition - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back History Favorites Search Links
Address C:\My Documents\ebxml\CodeSet.htm
Go
ebXML Value Domain Code Set Definition
Date of Submission:
Submitting Organization:
Contact E-mail Address:
Code Set ID:
Directory for Code Set storage: c:/My Documents/ebxml/codesets/
Control Agency:
Agency Assigned Name:
Code Set Concept Name:
Code Set Concept Description:
Code Set Data Format Type: None
Code Set Character Set: ISO 10646/Unicode
Code Set Data Format Definition:
Done
My Computer

1336



The fields provided on this form are:

- Date of Submission: enter an ISO 8601 conformant date using the format CCYY-MM-DD here
- Submitting Organization: enter the name of the submitting organization here
- Contact E-mail Address: enter e-mail address to which questions can be submitted
- Code Set ID: Enter a unique identifier for the code set. (This will form the file name of the code set definition.)
- Directory for Code Set storage: Should indicate the path required to reach the required directory. (When the default directory set up is being used this entry should not need to be updated from the pre-assigned values.)
- Code Set Control Agency: Enter name of agency responsible for adding new entries to code set.
- Code Set Concept Name: Enter name to be used to identify the concept behind code list. (May or may not be the same as the Code Set ID.)
- Code Set Concept Description: Enter description of role of code set that allows it to be distinguished from other code sets.
- Code Set Data Format Type: If code set has pattern based on one of the known data format languages select relevant entry from list. Otherwise select None
- Code Set Character Set: If relevant, select controlling character set from list supplied.
- Code Set Data Format Definition: If relevant, enter pattern that defines format of codes set values using language identified in Code Set Data Format field
- Permitted Value: Enter permitted code set values in the fields in this column.

- Description of Meaning of Value: Enter description of how value is to be interpreted.
- Language: If description in not in English, enter ISO 639 code for language used for description.

The form ends with the following buttons:

- Submit this Code Set: Creates XML record of form and displays this in a separate window. (Dismiss window if record is accurate: otherwise return to source and use the Back button to return to your entries so that they can be corrected and resubmitted.)
- Define another Code Set: Resets the form so that details of another code set can be defined.
- Define Data Format: Calls up the form needed to record a Data Format.
- New Entity Definition: Calls up the form needed to define an Entity.
- Create Data Representation Definition Calls up the form needed to define a Data Representation.

Troubleshooting

The forms described in this document have not been fully tested to date. If you encounter any problems using them please contact their author, Martin Bryan, via email at mtbryan@sgml.u-net.com (please be patient as I am traveling a lot during May/June so may not be able to respond as fast as you would like.)