Bolero Document Modeling Conventions

The following submission documents the modeling conventions used by Bolero in phase one of UML to XML conversion. They are submitted to the group in the knowledge that somethings could be done better. Currently we are working on phase 2 which involves schemas and changes to the conventions, as such, this document may be regarded as a prototype – but even prototypes have their uses. Many of the conventions used in this document will be changed in phase 2. The submission is submitted in the hope that it will stimulate discussion, and allow us to submit our thoughts on the future of the important topic of UML to XML production rules.

Phil Goatly October 2001

1.0 Background

1.1 Definition

Bolero$_{XML}$ makes a distinction between a message and a document. A document contains only data, whereas a message contains data and involves a sender and one or many receivers.
This report is only concerned with data modelling for documents. The interface between this document model and the message model is construed via the parties who send the messages and by the data which they send. The document model is the 'data model' and the message model is the 'data flow model'.

1.2 Analysis

In many cases, an electronic format for a document does not already exist. The modelling involves the creation of electronic documents definitions based on an emulation of currently available paper documents and details provided by the user community. This approach is intended to ease the introduction of electronic versions of the documents within the user community by an evolutionary process.

1.3 Uncertainty

The electronic document definitions are modelled in conditions of uncertainty, i.e. the scope of the documents to be provided is not known at the outset. The scope will be dictated by the user community and their business requirements, both of which will grow over time. The modelling is therefore different from the design of a 'computer system' or an existing messaging system, where the scope of the

data is explicitly defined at the design phase. It has been necessary to invent conventions that allow for a degree of uncertainty, in particular the convention of variations. (please refer to xx.xx)

1.4 Extensions

Unlike other providers of electronic document formats who have a homogeneous customer base, i.e. banks, Bolero$_{XML}$ must support the entire trade community and thus has to cater for industry specific extensions to documents. Due to the international nature of trade, country specific extensions are also needed.

1.0 Introduction

This document is intended to serve as a reference for any party involved in modelling the Bolero$_{XML}$ document definitions.

2.1 Background

Bolero$_{XML}$'s key objective is for the document models to have no dependency on the technology solution used for implementation. This safeguards the use of document definitions against any impact from changing technology where changes can be addressed without having to redefine the underlying business requirements.

In support of this objective, Bolero$_{XML}$ has chosen to model the requirements for these document definitions in Unified Modelling Language (UML), with a set of rules and conventions described in this document. UML was selected to support Bolero$_{XML}$'s vision of being able to house all definitions in a single repository, ensuring unambiguous data definitions and consistency across all documents.

The goal is to be able to extract the document definition, by means of an automated conversion, from UML to any desired implementation tool.

2.2 Key Principles

In the development of these document definitions, Bolero$_{XML}$ has chosen to focus on the following principles:

- Use 'best of breed' from existing EDI standards
- The scope is limited to the core 80% of business transaction needs whilst ensuring that the deliverable is operational
- Distinguish between 'automatable' and 'non-automatable' information
- Focus on user-friendliness rather than technical excellence
- Base all definitions on a single common business model

Bolero Document Modeling Conventions

Due to the dependent relationship between all document definitions, the methodology and tools used by Bolero$_{XML}$, any amendment to any feature of this document or the concepts described will have to be ratified by the Bolero$_{XML}$ team.

Any detail deemed necessary to support understanding for the conventions is included in a grey box.

2.3 Glossary of Terms

Abstract Class	An abstract class is a type of super class. It does not have any data attributes itself and is distinguished by the different functions performed by the classes associated to it.
Multiplicity	An expression to show the possible occurrences of an object. (1..1) – (minimum occurrences..maximum occurrences)
Conditionality	An expression to show a possible choice between objects. (OR)
Data Attribute	Objects which describe the properties of a class with one aggregate relationship, i.e. the leaf of the tree.
Data Class	A description of a group of objects with common properties and business meaning.
Object	A generic term for all attributes, classes and relationships.
Super Class	A class from which another class inherits the data attributes.
UML Attribute Notation	Objects which describe the properties of a class for an application or system. This explains technical detail such as the length of the attribute as opposed to business data.
UML Class Notation	The way in which a data class is represented in UML.

3.0 Rational Rose

The following outlines the specificities relating to the use of the Rational Rose modelling tool.

For the Rose Modeler Edition 2000 version in current use at Bolero, please note that these are the default folder settings, which cannot be changed:

- Use Case View
- Logical View
- Component View
- Deployment View
- Model Properties

3.1 Rose Folder

The entire Bolero model structure is to be housed within the **Logical View** folder. It is not envisaged at this time that any other Rose folder will be used.

3.2 Model Types

All 3 models described below are to be housed in one model in the Logical View folder within Rose.

- Business Process Model (no entry at present)

This comprises Use Case Diagrams, Activity Diagrams and workflow etc. to describe the business processes. Although some inaugural work has been done by Bolero, there is no immediate demand for work to be finalised on the business process model at this time.

- Business Data Model (no entry at present)

This is a relational data model defining all the relationships between the objects covered by the scope of the business activities in the trade documents.

- Document Model

This is a hierarchical data model for each trade document. Each document model must be hierarchical in order to avoid ambiguity and circular references.

3.3 Logical View Model Structure

This model structure follows a hierarchy within the different Rose packages. The foundation layer is the data typing package providing the primary building blocks. Each subsequent layer is an increment for firstly the definition of the business attributes, the business classes and culminating in the document models. The following describes how to suit different object definitions to the correct package within the model structure.

The Logical View model is divided into the following sections:

1.0 Data Typing
1.1 Basic Data Types
1.2 Composite Data Types

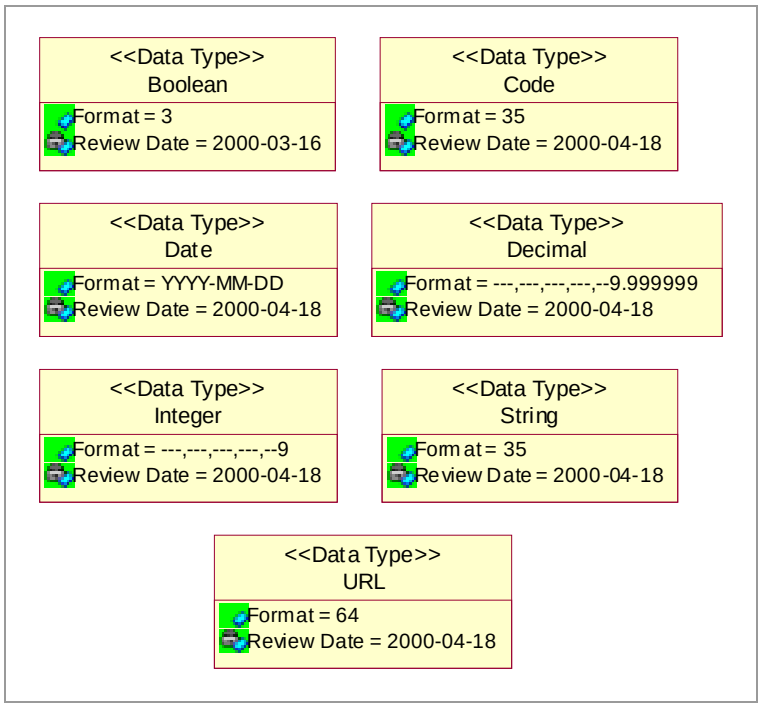
2.0 Business Semantics
2.1 Business Elements
2.2 Business Components
2.3 Un-modelled Abstract Classes

3.0 Business Data Model
3.1 Relationships

4.0 Documents
4.01 Bill of Lading
4.02 Packing List
4..n etc.

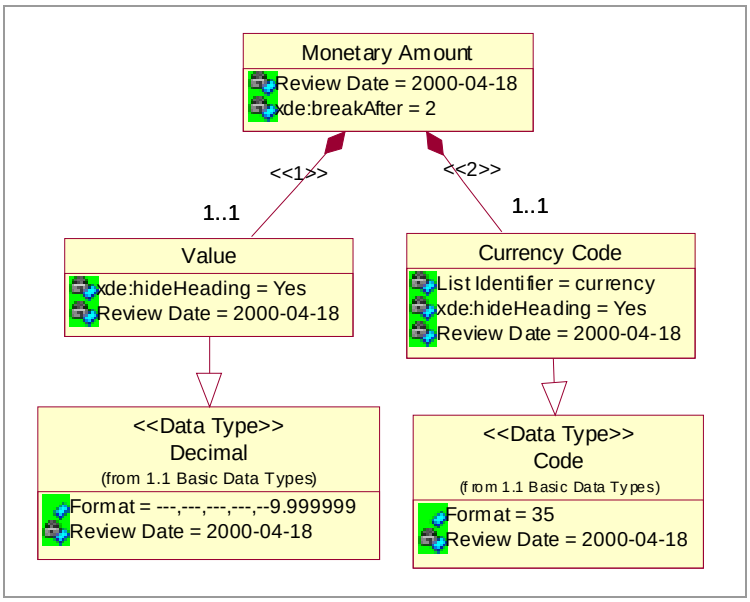
3.4 Data typing - Basic Data Types

Basic data types comprise the fundamental objects upon which all others are constructed. These objects describe the structure of the data, often for the benefit of technical understanding and they do not infer any business meaning.
All data attributes of a class must be defined by one of the basic data types. These basic data types cannot be data attributes themselves.



3.5 Data typing - Composite Data Types

Composite data types are derived from the basic data types. These are normally used where two or more data attributes are required to represent a concept. Composite data types do not infer business meaning. Composite data types, which are generalisations of each basic data type should be held in their own sub-package and diagram for ease of use and reference.



Explanation

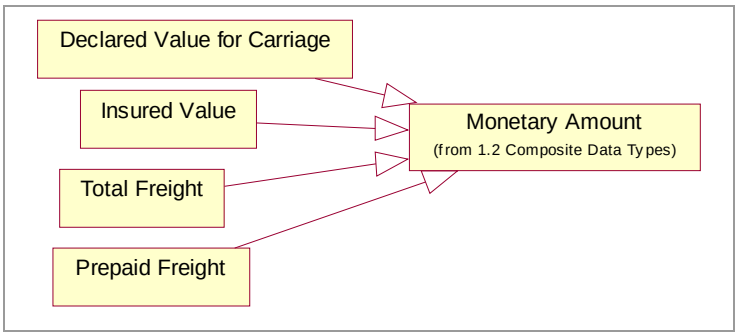
In the above diagram the class <Monetary Amount> is made up of two mandatory attribute classes; <Value> of Basic Data Type <<Decimal>> and a <Currency Code> of Basic Data Type <<Code>>.

3.6 Data typing – Correct Usage

None of the data type objects can appear on any trade document. The data types will only ever be used in the Business Elements package. The data attributes of these business elements will be a “type of” or generalisation of the basic and composite data types.

3.7 Business Semantics - Business Elements

The business elements are the first level in the hierarchy at which business meaning is derived. They can be considered to be “business attributes” and can only be re-used to formulate the business components.

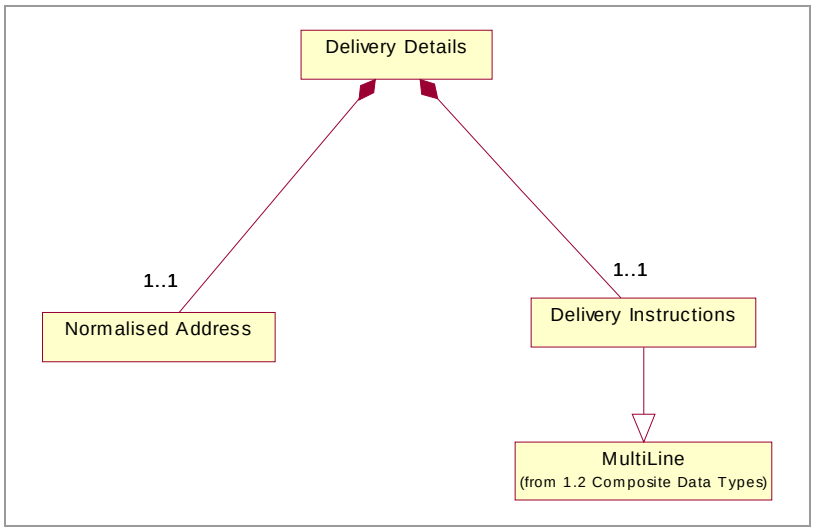


Explanation

The above diagram shows that all the classes on the left of the diagram are of type <<Monetary Amount>>. As such they inherit classes <Value> and <Currency Code>.

3.8 Business Semantics - Business Components

The business components are constructed from the business elements and can be considered as “business classes”. A business component reflects a concept which can have a number of variations which may appear on different documents.
All business components must have an abstract class and have at least one variation of this abstract class. Each variation is modelled as a generalisation or “type of” the abstract class. The business attributes are contained within each variation. Common business attributes must be associated to each variation separately. It is envisaged that new elements may need to be defined as further document definitions are modelled. Business components are the only objects that can appear on the trade documents. Please refer to CV2.3 for further detail on abstract classes and the example in Figure I.



Explanation

Delivery Details consists of two mandatory classes <Normalised Address> and <Delivery Instructions>, which is of type <<Multi Line>>.

3.9 Business Semantics – Non-Document Classes

The non-document classes are found in the business data model only. These classes describe a concept, a term or an assumption which has meaning for the business context but which would not be directly conveyed in the document models. These classes are important as they denote a relationship to other classes in use, but they are not used themselves in the trade documents. An example is <Consignment>, a valid concept but which is not found on a trade document itself. It appears on the business data model related to <Commodity> and <Container>, which can be a part of a <Consignment>.

3.10 Business Data Model- Relationships

This section sets out the abstract classes in use across all trade documents and illustrates the relationships and dependencies between them. Abstract classes are used on the business model to illustrate a concept without any document or context specificities. The abstract class <Commodity> is used on the business model to denote the high level concept, whereas <Commodity (1)> is one of the variations needed directly on a document model for a specific context.
An abstract class has no “contained within” or aggregation relationship with another object. As this model is depicting high-level relationships between objects, it is not relevant to describe generalisations, aggregations or Multiplicity rules. These are applied in the Business Semantics section, which must include the documentation of all associated data attributes.

3.11 Documents

This section comprises the document models used to describe each trade document. These models will include the specific variations of the abstract classes defined in the business model.

It will be necessary to follow the existing numbering scheme for all subsequent documents, i.e. 4.15, 4.16 etc.

All documents have a number comprising a document type code and description, pre-defined by the Bolero Core Messaging Platform, derived from EDIFACT.

The document header will always be the same and is defined in the Bolero Document message. The document body comprising a document class and a body class are the only classes that can be defined in this package.

4.0 UML Specificities

In accordance with the current modelling symbols used by Bolero, the following points must be adopted:

4.1 Object Naming

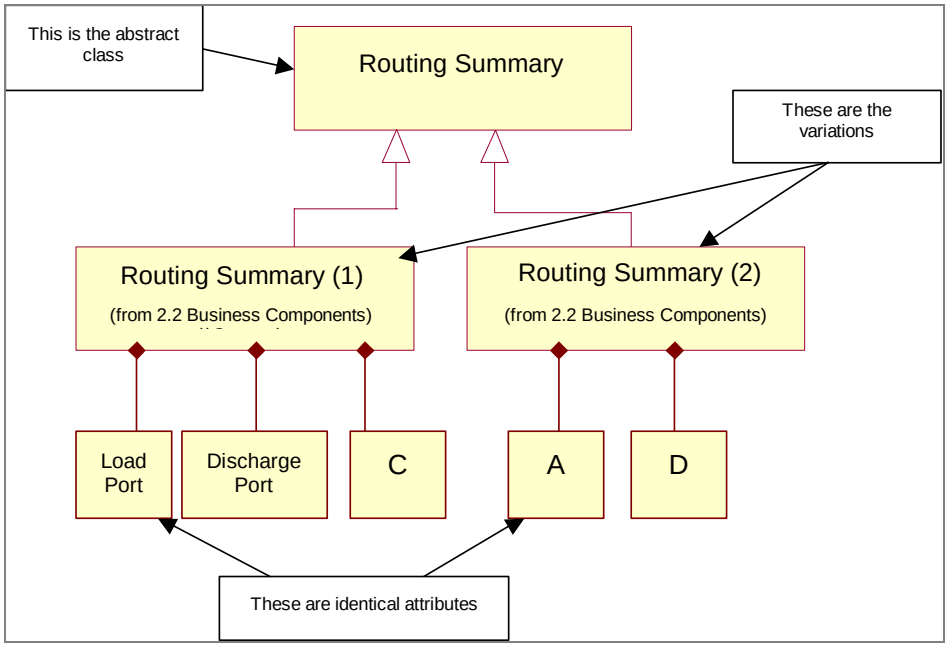
All objects must be named using upper or lower case letters or numbers only. No numbers can appear at the beginning of the name. Matching brackets ( ) can only be used for the naming of variations of abstract classes. The brackets must not be used at the beginning of the name.

4.2 Classes

All data classes and data attributes are defined as UML classes. This is to facilitate the re-use of attribute definitions across business components.

4.3 Abstract Classes

Abstract classes are defined in the Business Components package and are used only in the business model. An abstract class may not have any specific business attributes.



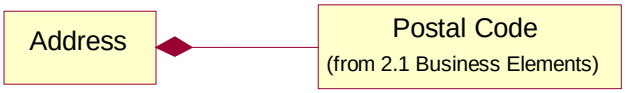
4.4 UML Attribute Notation

It must be noted that the UML attribute notation is not used in the Bolero models to convey data attributes. This is to permit the re-use and ease of update for objects when the UML attribute notation restricts them to the class to which they are specifically associated. Rather, the attribute notation can only be used to describe specific detail such as attribute length to support technical implementation.

4.5 Business Attribute

Business attributes must be defined in the Business Elements package. Objects defined in this package cannot have other objects in an aggregate relationship to them.
Bolero applies the rule that an object is considered to be a business attribute when it only has one aggregate relationship, i.e. the leaf of the tree.

The object <Postal Code> has a single, direct relationship to <Address>



4.6 Meta Data Attributes

Where relevant, the recommended length or format for each business attribute is defined in one UML attribute named <format>. This detail will in some cases be inherited from the related super class.
Please note that <<Date>> must be defined in one, universal format; YYYY-MM-DD.
For classes of type <<Code>>, a <List Identifier> attribute is required for the code list name. This name must be an abbreviation of the class name.

1. Documentation (Semantics) Inheritance

The following convention will allow the inclusion of Documentation from other classes than the current class.

In the documentation for the 'current class' one may place the following:

%Include ClassName;

The statement must start on a new line and should have nothing after the semi-colon.
The above syntax will have the effect of replacing the %Include with the documentation for the class <*ClassName*>. The replacement is part of the functionality of the Bolero$_{XML}$ Generator suite of Programs.

2. Format will be included in documentation

If the Format is totally numeric the following will be generated:

Recommended Length: *n*

If the Format is not Totally Numeric the following will be generated:
Required Format: *aaaaaaa*

3. Bolero Document Types and Names

The fixed Document Type Codes and Document Type Names will be included in the model as follows:

Each class with a stereotype of <<Document>> will have two additional attributes as follows:

<DocumentTypeName> with an Initial Value appropriate to the Document Class.
i.e. <Bill of Lading>

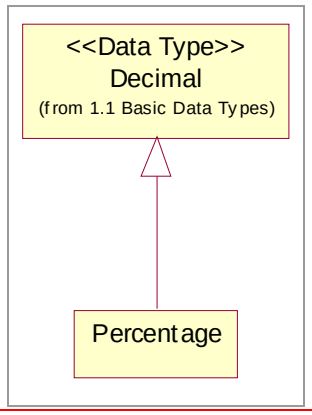
<DocumentTypeCode> with an Initial Value appropriate to the Document Class.
i.e. <705>

4.7 Relationships

To facilitate document standards development, Bolero$_{XML}$ does not need to utilise the full software development aids in Rational Rose. Bolero$_{XML}$ has adopted a simplified approach and only needs to illustrate two relationships.
For the document models, relationships between classes are illustrated by generalisation (Superclass Indicator) or aggregation (Strong Aggregation) only.

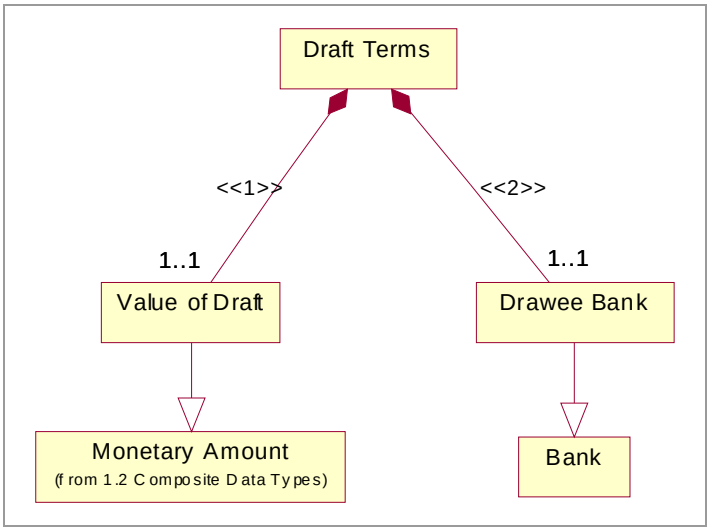
Generalisation
This relationship shown by the arrow on the diagram below infers that an object is a *type of* another object.

Example: The object <Percentage> is defined as a <<Decimal>>.



Aggregations

This relationship shown by the diamond arrow on the diagram below denotes that an object is *contained within or part of* another object and is used to show composite relationships.



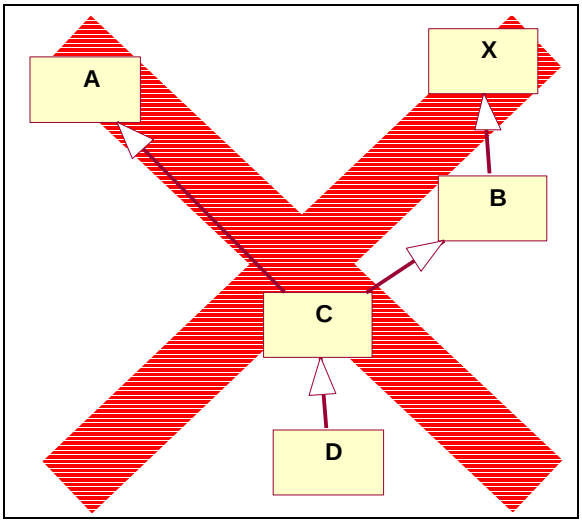
Explanation

<Draft Terms> contains <Value of Draft> and <Drawee Bank>.
<Value of Draft> is a type of <Monetary Amount> and <Drawee Bank> is a <Bank>.
The 1..1 tells as the *minimum occurrences .. maximum occurrences.* This expression is known as 'Multiplicity' or 'multiplicity'.

4.8 Multiple Inheritance

Please note that multiple inheritance, where common data attributes can be inherited and replicated among numerous classes, is not used in the Bolero models as described in the diagram below. Bolero has not experienced the need for this construct and expects that it might be difficult to manage.

Example: Object D is a type of object C. Object C is also a type of object B and A, with object B being a type of object X. This means that by inference, object D may inherit all the data attributes that are associated to the other objects. The effect of introducing new data attributes on each object is complex to trace and manage effectively.



4.9 General Document Construct

The document has two key sections, the header and the body.

- *The header* is standard across all Bolero documents and includes all document identification information such as *Document Reference, Document Type and Document Type Code.* This section includes the user defined document number such as the Bill of Lading number which is not modelled as a business attribute and therefore will not appear again in the main body section.

- *The body* is the main sub-section and at present comprises all the document specific objects. It is envisaged that it may be favourable to decompose this section into further sub-groupings for summary detail and calculation of totals for example. This construct could be used to indicate the generation sequence of each sub-section.

4.10 Code Lists

- Validated code lists are pre-defined and universally referenced. They have a <List identifier> UML attribute to indicate validation from an external source.

- Unvalidated code lists may be user-defined or have localised meaning. They have a UML attribute for <Valid Values> for maintenance within the model.

4.11 Stereotypes

Stereotypes are used to extend the meaning and qualify the function of the object. This meaning is otherwise not explicitly provided in the model. A stereotype name is located within brackets: << >>.

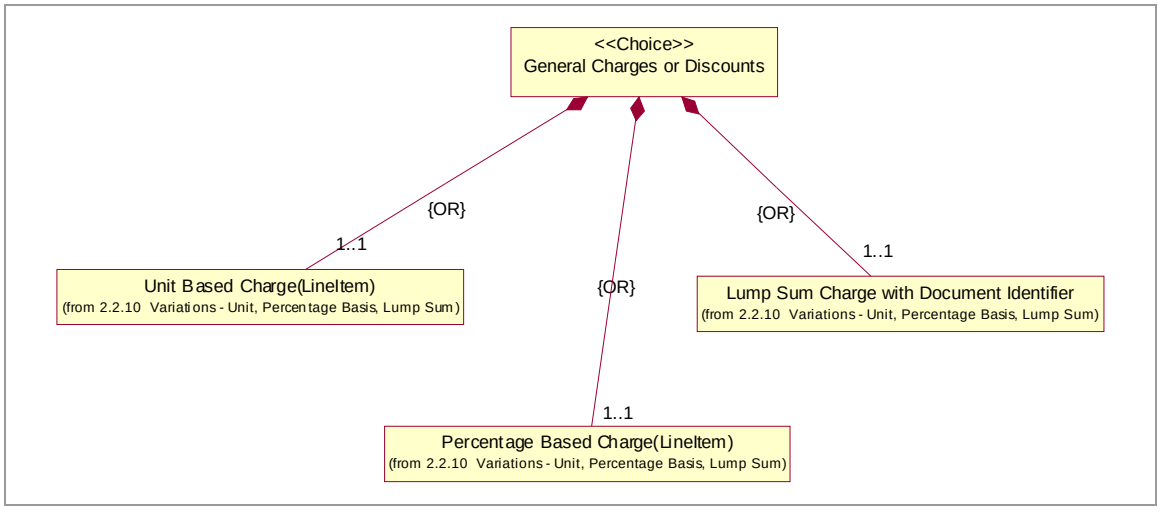
The following stereotypes have been defined:

<<Common>> Stereotypes on packages
This is a method employed to indicate that every object within the package stereotyped <<common>> is visible to any other class in the model. Objects not in common packages are only visible in their own package. This visibility is used by the Bolero$_{XML}$ Generator Program

<<Document>> Stereotypes on classes
This is used to represent an object in the model as a trade document.

<<Data Type>> Stereotypes on classes
Within the basic data type package, this stereotype is used to indicate that these objects describe the nature of the data itself. Therefore, when these basic data types are used in the composite data types or business elements packages, users can see that they describe the data without having to trace them back and consult the basic data type package.

<<Choice>> *Stereotypes* on classes
This stereotype is used to indicate conditionality. that the classes aggregated into the super class are Choices i.e. OR.



Explanation
The ‘General Charges or Discounts’ class represents a Choice between the aggregate relationships of ‘Unit Based Charge’ OR ‘Percentage Based Charge’

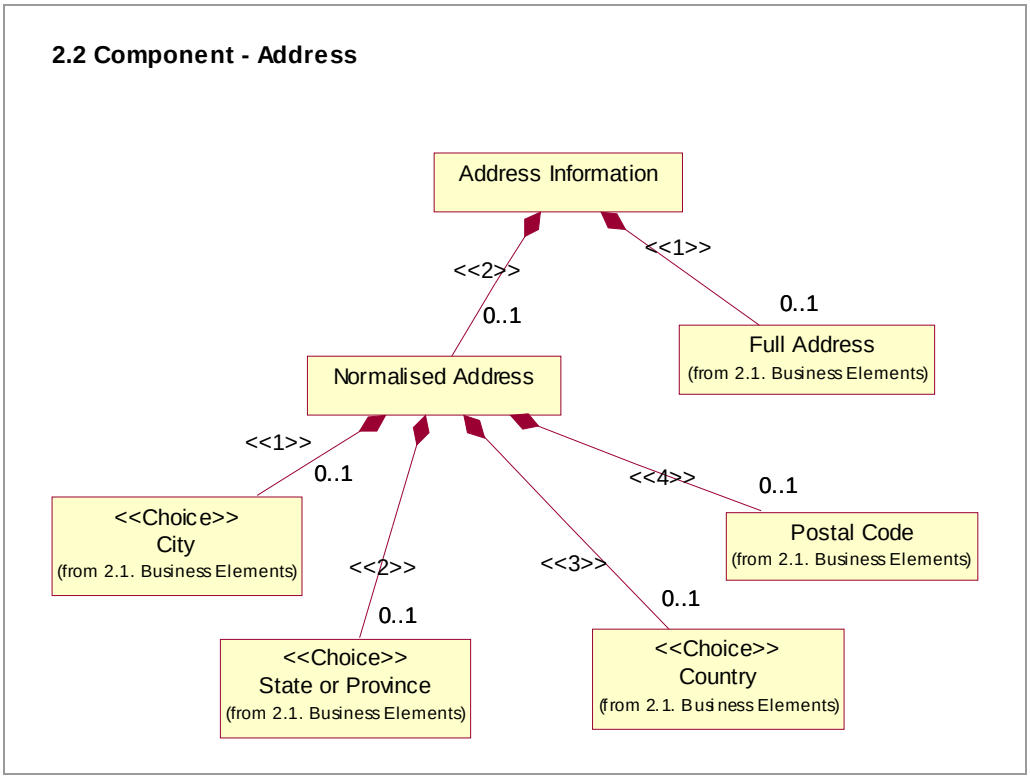
OR ‘Lump Sum with Document Identifier’ classes. (The OR represents an exclusive or XOR).

4.12 Circular References

Circular references between classes must be avoided due to the ambiguity inferred on the document definitions. Such references invalidate the integrity of a hierarchical model. Intentional recursive, also called reflexive references, may be modelled.
Object constraint language can be used to indicate these desired, reflexive circular references.

4.13 Generation sequencing

A number shown on the object relationships to indicate the order in which objects will appear on a trade document. This is information may be required for the process of converting the model for a form of technical output such as a DTD (document type definition).



Explanation

The number on the Stereotypes on the associations between 'Address Information' class i.e. <<1>> and <<2>> indicate the order in which the generator should generate XML information for the classes at the end of the association i.e. 'Normalised Address' and 'Full Address'.

4.14 Reflexive circular references

This is a construct used to allow, specifically for the often confusing scenario of cartons held within boxes on bigger boxes on a palette, and their relevant counting basis to be modelled on a document effectively.

5.0 Data Dictionary

The Bolero data dictionary comprises all the definitions of objects contained within the models. In order to ensure consistency, all UML documentation of data classes and data attributes must include the following properties;

- Multiplicity – the multiplicity of relationships between objects
- Business rules – mandatory, optional or conditional rules
- Recommended attribute length and format in a UML attribute
- Data typing
- Semantics – "plain English" explanations as detailed below

Semantics Rules

This section refers to use of the documentation feature within the Rational Rose modelling tool where all objects can be described with a free text description.

5.1 Structure

All documentation must follow the construct of a textual description and an example comprising sample data.

5.2 Comprehension

The naming and descriptions must support understanding for a "reasonable man". It is necessary to avoid examples such as <Conveyance ID> as a standard term where others such as vessel ID or flight number may be more universally familiar. It is necessary for the definitions to be explicit, unambiguous, comprehensive and generic as possible. In most cases, a single line description will not be suitably exhaustive. It is required that no abbreviations are used. Use of existing universal standard terms is advocated where possible. It must also be recognised that the wording and explanations used must not prevent or deter accurate translation into other languages nor invoke an inappropriate interpretation within the model or a document context.

5.3 Definition/context

Definitions must not be technology specific or biased. They must be entirely distinct and must not include or rely on any reference to a document or process, except where a unique business concept exists in a document. An example of this is the object <Insurance Policy Section>, which is relates to a concept that will only ever appear on an insurance document.

5.4 Examples Format

Examples are always required to support documentation. The use of examples must only be for the purpose of elaborating on the textual description. Examples must never be used alone to describe an object.

To support a description, an example can be in textual form or sample data. It is necessary to follow the construct – Example: <example> as opposed to <e.g.>. The example must appear below the description, after a blank line.

Example:

<Location>
A geographical entity that is clearly identifiable even when no street address is given. It may be a town, a city or a place with a named description. It is not a country.

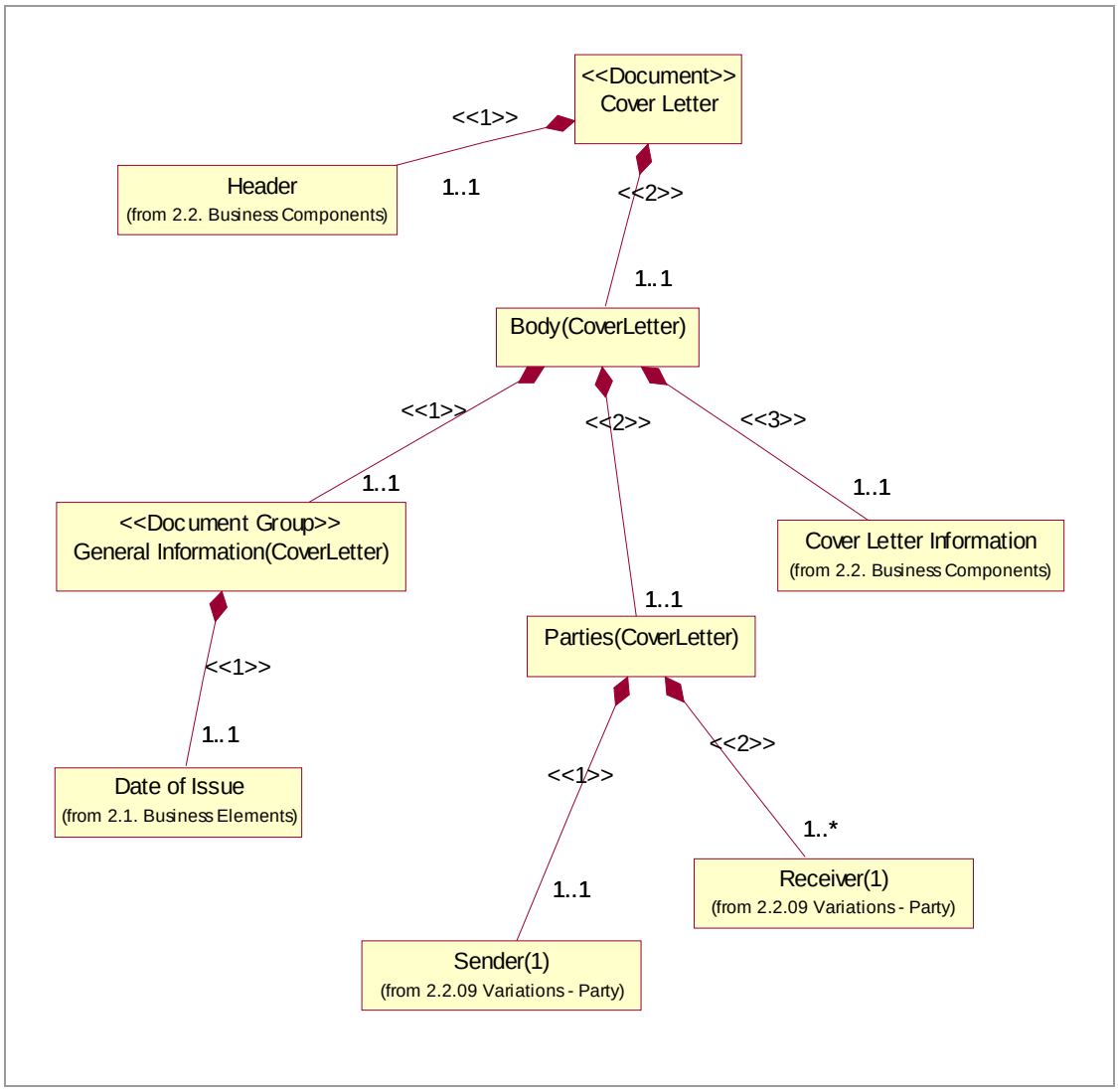
Example: Heathrow Air Cargo Terminal.

5.5 Relationships

Please note that relationships between objects will not be specifically described. It is felt that there is no immediate need given the integrity of the documentation provided.

6.0 Relationship of UML Model to XML

6.1 Processing of Document Hierarchies



In the above hierarchy there are a number of points to note:

1. The root or top of the hierarchy starts with a class with <<Document>> as the stereotype. The generator will only start at classes stereotyped as documents.

2. The root of the <<Document>> is considered to be Level 0 of the hierarchy. The 'Header' and 'Body' classes are at Level 1. 'General Information' , 'Parties' and 'Cover Letter Information' are at Level 2 etc.

3. The brackets at the end of a class indicate a variation as described above in CV3.3. On finding a variation the generator removes any text in the class name past the "(". Thus Sender(1) becomes Sender as a 'tag' and Body(CoverLetter) becomes Body as a 'tag'.

4. The output generated from the above diagram in the DTD is shown below. The grayed out area is part of the DTD, but the 'tags' produced are not shown on the diagram, but are to be found on other diagrams which have associations with these classes shown in the diagram.

5. As will be seen from the generated DTD, the generator starts at the top of the hierarchy processing down to the 'bottom' of each node encountered and producing the requisite DTD code.

6.1.1 Generated DTD

```
<!ELEMENT CoverLetter (Header,Body)>
<!ELEMENT Header (cmp:DocumentID,cmp:DocType,cmp:Status)>
<!ELEMENT cmp:DocumentID (cmp:RID,cmp:GeneralID,cmp:Version?)>
<!ELEMENT cmp:DocType (cmp:DocTypeCode,cmp:DocTypeDescription?)>
<!ELEMENT Body (GeneralInformation,Parties,CoverLetterInformation)>
<!ELEMENT GeneralInformation (dateOfIssue)>
<!ELEMENT Parties (Sender,Receiver+)>
<!ELEMENT Sender
                (organizationName,OrganizationIdentification*,AddressInformation?,Conta
                ctDetails?)>
<!ELEMENT OrganizationIdentification
                (organizationReference,organizationReferenceType?)>
<!ELEMENT AddressInformation (FullAddress?,NormalisedAddress?)>
<!ELEMENT FullAddress (line+)>
<!ELEMENT NormalisedAddress (City?,StateOrProvince?,Country?,postalCode?)>
<!ELEMENT City ((locationCode|locationName))>
```

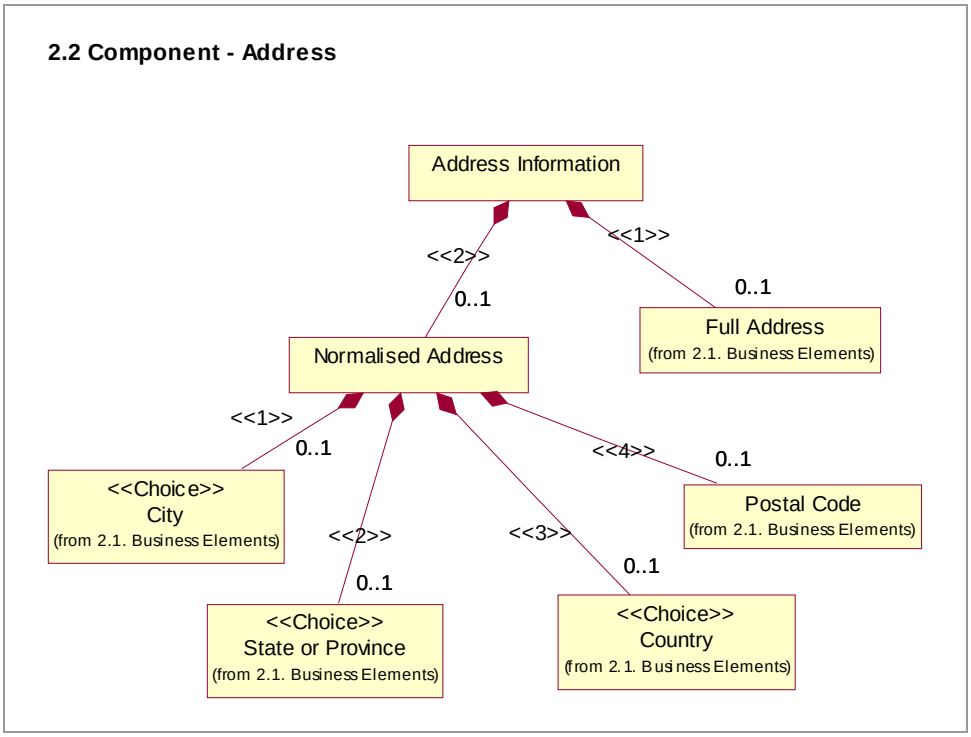
<!ELEMENT StateOrProvince ((stateOrProvinceCode|stateOrProvinceName))>
<!ELEMENT Country ((countryCode|countryName))>
<!ELEMENT ContactDetails
(personName?,departmentName?,telephoneNumber?,faxNumber?,e-MailAddress?)>
<!ELEMENT Receiver
(organizationName?,OrganizationIdentification*,AddressInformation?,ContactDetails?)>
<!ELEMENT CoverLetterInformation (line+,CoverLetterDocumentsAttached*)>
<!ELEMENT CoverLetterDocumentsAttached
(documentsIncluded,AdditionalDocumentInformation)>
<!ELEMENT AdditionalDocumentInformation (line+)>

Explanation

In the above fragment the ‘tag’ ‘Receiver is followed by a plus sign “+” in XML terms this means that there must be at least 1 occurrence of the ‘tag’ and it can repeat. This is generated from the “1..*” Multiplicity in the Model. Any ‘tag’ without a qualifier such as “ ?” or “ *” or “+” must be there i.e it is mandatory, and is generated from the Multiplicity “1..1”

6.2 Processing of Classes

When the Generator ‘encounters’ a class in the model it generates an XML ‘tag’.
The first letter in each word is converted to upper case. The spaces in the class name are then removed, and if the class is a leaf element i.e. a Data Node as opposed to a structural node, the first letter of the name is lowercased.



In the above diagram the ‘tag’ for the ‘Normalised Address’ class is ‘NormalisedAddress’ and the tag for ‘Postal Code’ class is ‘postalCode’.

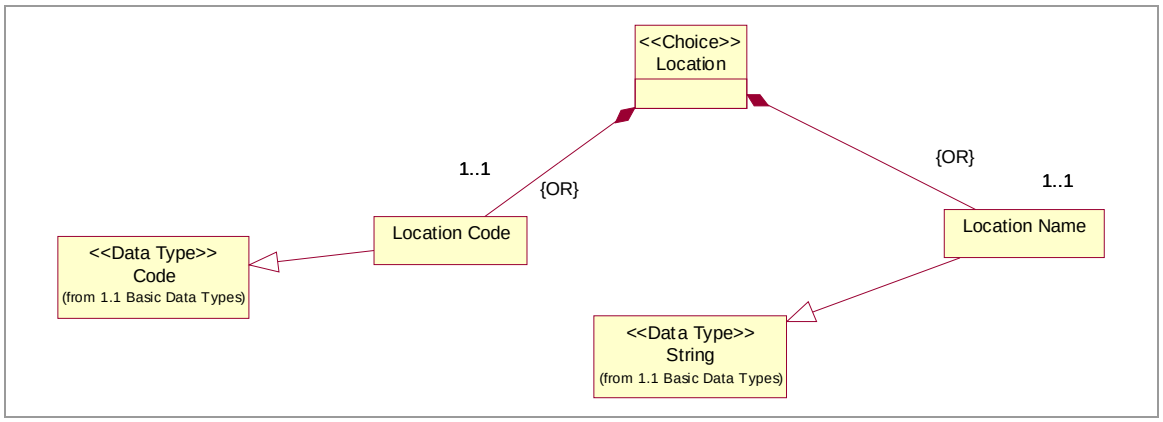
6.2.1 Generated DTD

```
<!ELEMENT AddressInformation(FullAddress, NormalisedAddress?) >
 <!ELEMENT NormalisedAddress(
City?,StateOrProvince?,Country?,postalCode?)>
```

Explanation

The ? in the DTD fragment above indicates that the data element is optional in line with **0..1** Multiplicities in the diagram above. The lowercase first letter of 'postalCode' shows that it is a data element. The order of the 'tags' is related to the numeric Stereotypes shown on the aggregation relationships in the diagram.

6.3 Processing Choices



The {OR}constraint is used to provoke the generation of the XML OR i.e. '|' (vertical bar). Given the <<Choice>> Stereotype on the Location class the {OR} is not strictly necessary.

6.3.1 Generated DTD

```
<!ELEMENT Location(locationCode|locationName) >
```