Digital Policy Office

XML SCHEMA DESIGN AND MANAGEMENT GUIDE
PART II: XML SCHEMA DESIGN GUIDE

[G55-2]

Version 1.5

Jul 2024
The Government of the Hong Kong Special Administrative Region
of the People's Republic of China

Distribution of Controlled Copy	
Copy No.	Holder
1	Government-wide Intranet (itginfo.ccgo.hksarg)
2	Internet (www.digitalpolicy.gov.hk)

Prepared By: XML Coordination Group

Doc. Effective Date: 1 January 2006

1

Amendment History				
Change Number	Revision Description	Sections Affected	Revision Number	Date
	Updates to consultation draft issued in July 2003		1.0	24-Nov-03
1.	Clarified that this Guide is intended for B/Ds and contractors to define the data exchange interfaces in XML Schema for joined-up services.	1.1		
2.	Clarified that this Guide is complementary with and does not aim to replace existing system design or data modelling methodologies.	1.1		
3.	Added that the Business Information Modelling (BIM) methodology is optional for Bureaux and Departments (B/Ds) and contractors to follow when defining the data exchange interfaces of cross-departmental systems and the interface to external systems while the Business Process Modelling (BPM) methodology is optional.	1.1, 1.2, 3.1, 4.1		
4.	Emphasized that the modelling spreadsheet should be used instead of worksheets to ease capturing modelling information.	1.1, 1.2, 2, 4.4, 6.1		
5.	Clarified the duty of a business analyst.	1.2		
6.	Introduced the concept of Externally Defined Entity in BIM, which allows an Association Business Information Entity (ASBIE) to associate with an externally defined XML Schema.	4.3.2, 4.3.5		
7.	Added a subsection to clarify the aggregations and associations between Business Information Entities (BIEs) should refer to specific versions of these BIEs while multiple BIE versions exist.	4.3.9		
8.	Clarified that the usage rules field in the data dictionary can be used to capture any validation rules that cannot be supported by the XML Schema language but should be handled by system implementations.	4.4.1		
9.	Clarified that project teams may fine-tune the XSDs produced according to the methodology.	5.1, 5.4.5		
10.	Updated the XML Schema Definition (XSD) development to code the Externally Defined Entity in XML Schema.	5.3.3		
11.	Added more details about importing external schemas and assigning namespaces.	5.4.1, 5.4.2		
12.	Added Section 5.4.7 to analyze 2 different approaches of schema reuse: copy code versus import from external sources.	5.4.7		
	Major updates to version 1.0 issued in November 2003		1.1	01-Jul-04
13.	Renamed organization name from ITSD to OGCIO	Whole document		

Amendment History				
Change Number	Revision Description	Sections Affected	Revision Number	Date
	Major updates to version 1.1 issued in July 2004		1.2	2-Nov-04
14.	Modified Figure 2-1 to advise project teams to adopt industry standard for individual data element before considering to adopt Common Schemas.	2		
15.	Revised to document the enhancement of Business Information Modeling utility to allow making choice for data elements in XML Schema.	5.3.3, 5.4.5, 6.2.4		
	Major updates to version 1.2 issued in November 2004		1.3	4-Jan-06
16.	Minor version number upgraded to 1.3 according to annual review requirement of S&M [G57].	Whole document		
17.	Minor revision in light of OGCIO Circular No. 2/2015 regarding "Structured Systems Analysis and Design Methodology (SSADM)" and "Rapid Application Development (RAD)".		1.4	30-Mar-15
18.	Renamed organization name from OGCIO to DPO	Whole document		
	Major updates to version 1.4 issued in March 2015		1.5	25-Jul-24

1

Table of Contents

1 Introduction

1.1. Purpose of This Guide

This Guide aims to facilitate the process of designing and defining quality, consistent and reusable **XML Schema Definitions (XSDs)** in a systematic manner. Capturing international best practices and standards available, the Guide serves as an instruction manual for business analysts and developers to participate in the schema design process.

This Guide is intended for Bureaux and Departments (B/Ds) and their contractors to design XML Schema that defines the interface for data exchange between two or more B/Ds or between B/Ds and businesses in joined-up services. This Guide is complementary with and does not aim to replace existing system design or data modelling methodologies, e.g. OOM. In other words, a project team is still required to apply appropriate system analysis and design (SA&D) methodologies to design its entire software solution, including the development of a complete logical data model for the system.

After going through this Guide, readers should be able to:

- Understand the methodology and the sub-processes involved in the XML Schema Design Process;
- Be familiar with business and technical terminology related to the design methodology;
- Understand how to identify business documents from business processes systematically;
- Understand how to organize the information in the identified business documents into data elements and develop information models for data elements;
- Understand technical details in converting information models to XSDs; and
- Understand the provided schema design worksheets and able to use functionally-equivalent spreadsheets to apply the methodology.

This Guide describes the following two main methodologies in the schema design process:

- Business process modelling (BPM) methodology; and
- Business information modelling (BIM) methodology.

The BIM methodology is optional for B/Ds and contractors to follow when designing data exchange interfaces in order to comply with this Guide. The BPM methodology is optional for B/Ds and contractors to follow. B/Ds and contractors may apply other SA&D methodologies to model business processes and identify business documents for exchange.

1.2. Audience and Structure of This Guide

This Design Guide is intended for business analysts and programmers who participate in joined-up service project implementation. These individuals may be from the government B/Ds as well as contractors (e.g. system integrators), who provide professional IT services to the B/Ds.

Business analysts are project members who analyze the requirements on business process and information gathered from domain experts. They construct process and information models from business requirements, in particular for programmers to develop Project Schemas. In the context of this Guide, the duty of a business analyst can be fulfilled by a system analyst working with experienced business users who possess the domain knowledge.

Programmers are project members who are responsible for implementation of software solutions according to the software specifications, process and information models constructed by business analysts. In relation to Project Schema development, programmers are responsible for converting the models to XSDs using the provided conversion mechanism. Programmers who read this Design Guide are assumed to have the basic XML Schema programming skill. They convert information models into quality, consistent and reusable XSDs.

To get a complete view on the schema design and obtain best results, joined-up service project teams are recommended to go through **Part I: Overview** and read **all sections of this Design Guide** at least once. After members are familiar with the schema design process and terminology, they can refer to specific sections for quick reference.

This Guide consists of the following five main sections:

Section 2: XML Schema Design Process

This section outlines the process for designing XML Schema for implementation of HKSARG joined-up services. The process for Project Schema design is provided.

Section 3: Business Process Modelling

This section provides the BPM methodology for business analysts to systematically model business processes in joined-up services with a view to identifying the business documents exchanged across B/Ds in the business processes. It is based on the approaches suggested in the eBusiness eXtensible Markup Language (ebXML) framework. To help readers understand the methodology, this section also discusses technical details on terminology – Business Transaction and Business Collaboration. Business analysts may choose not to follow this BPM methodology if the SA&D methodologies they currently use can already serve the purpose of modelling business processes and identifying business documents for exchange.

Section 4: Business Information Modelling

This section describes the BIM methodology for business analysts to capture requirements on the business information to be exchanged in HKSARG joined-up services, and to specify the requirements as information models. It is based on the approaches suggested in ISO 11179, Core Components Technical Specification (CCTS)[1], and Universal Business Language (UBL), to derive a comprehensive and practical methodology. Section 4 also discusses technical details on terminology – Object Class, Property, and Representation, Core Component Type (CCT), Basic Business Information Entity (BBIE), Association Business Information Entity (ASBIE), Aggregate Business Information Entity (ABIE), Business Document, Business Context, data dictionary entry information, and Dictionary

[1]

Entry Name**. The BIM methodology is optional to follow when designing data exchange interfaces in order to comply with this Guide.**

Section 5: XML Schema Definition Development

Section 5 provides the mechanism that programmers should follow to convert information models, which are prepared by business analysts, into XSDs. XML naming rules are also discussed. In addition, this section provides guidelines on namespace assignment, versioning, and meta-data documentation of schema documents.

Section 6: Modelling Worksheets

This section describes seven modelling worksheets that provide a complete view on what modelling information to capture for the process and information models. They are intended to ease the understanding of the methodologies discussed in Sections 3, 4, and 5. In practice, the **data modelling spreadsheets** are recommended to substitute the data modelling worksheets to capture modelling information because the modelling spreadsheets allow easier data entry and can automate the conversion of modelling information into XSD code. The modelling spreadsheets are published in the Central Registry for download.

Supplementary information pertinent to this Design Guide is provided in Part IV Appendices. To illustrate the entire methodology described in the Design Guide, a case study is presented in Appendix 1.

1.3. Notations Used

Unified Modelling Language (UML)[2] is used to graphically illustrate the concepts throughout the document. Readers are expected to possess basic knowledge on UML. The URL of the UML official website is https://www.uml.org.

[2] Unified Modelling Language (UML) is an object-oriented visual language maintained by Object Management Group to model and specify business and software systems.

2 XML Schema Design Process

This section outlines the process for designing Project Schemas for implementation of HKSARG joined-up services. The XML Schema design process involves four main sub-processes: business process modelling, business information modelling, XML Schema Definition development, and schema management. The technical details of the first three sub-processes are elaborated in subsequent sections of this Guide. The last sub-process is discussed separately in the XML Schema Management Guide.

The flowchart shown in Figure 2-1 shows the Project Schema design process. The steps in the design process are described as follows:

1 Analyse and gather the high-level project requirements based on the software development methodologies such as OOM, conventionally used for systems analysis and design (SA&D) in e-government projects. This step should be done by business analysts.

1. Study the existing and emerging industry standards, such as UBL, to see if any standard can fulfil the project requirements or whether the business practices can be reengineered to align with the standard practice promoted in a relevant standard. If a standard is available to provide suitable Project Schemas for the project, adopt the standard and go to Step 10. This step should be done by business analysts in consultation with the domain experts.

2. Model the business processes by following a conventional SA&D methodology or the business process modelling methodology provided in Section 3 to identify all business documents to be exchanged among multiple business partners. This step should be done by business analysts.

3. Decompose the identified business documents into data elements according to the business information modelling methodology provided in Section 4. This step should be done by business analysts.

4. Search relevant industry standards for reusable data elements to represent the decomposed data elements. If suitable reusable data elements are found, adopt them in the Project Schema design. This step should be done by business analysts in consultation with the domain experts.

5. For remaining undefined data elements, search the Central Registry for the concertedly aligned Common Schemas that are suitable for reuse to represent the decomposed data elements. Human judgment is required for determining the suitability of a Common Schema. If suitable Common Schemas are found, adopt them in the Project Schema design. This step should be done by business analysts. The business analyst may do keyword search on the Common Schema Index Page or the Common Schema Data Dictionary (currently implemented as a spreadsheet) to locate the relevant Common Schema.

6. Develop the information models, according to Section 4, for the remaining data elements. These are the project-defined data elements. Existing Project Schemas, including those of other e-government joined-up projects, should be referenced in the modelling. This step should be done by business analysts. The data modelling spreadsheet provided in the Central Registry may be used to develop the information model.

7. Develop the XSD code for the information models according to the XML Schema Definition development methodology provided in Section 5. This step should be done by programmers.

8. Organize the information models and XSDs in the Project Registry. This step should be done by business analysts.

9. Register the project in the Central Registry on a centrally maintained list of XML projects. This step should be done by business analysts.

10. Identify the project-defined data elements that have potential for reuse by other projects and contribute those elements for concerted alignment through submission of Common Schema Creation / Change Requests[3]. This step should be done by business analysts.

This Design Guide provides the methodologies and tools for business analysts and programmers to specifically do Steps 3, 4, 7, 8, and 9 as summarized in Table 2-1. This Table corresponds to the "dotted box" of Figure 2-1.

Table 2-1: Use of the Design Guide in the schema design process.

Step	***Personnel who perform the step***	***Methodology / Tools provided in this Guide***
3. Model business processes to identify involved business documents	Business analysts	A conventional SA&D methodology or the business process modelling methodology provided in Section 3
4. Decompose the business documents into data elements	Business analysts	The business information modelling methodology provided in Section 4.
7. Develop information models for the project-defined data elements	Business analysts	The business information modelling methodology provided in Section 4.
8. Convert the information models into XSD code	Programmers	The XML Schema Definition development methodology provided in Section 5.
9. Organize the information models and XSDs (i.e. Project Schemas) in the Project Registry.	Business analysts	The modelling spreadsheet for organizing information models.

[3] If a project team can recognize the reuse potential of the data elements at an earlier stage (e.g. when developing information models in Step 7), it may raise the Common Schema Creation / Change Requests on those data elements before it finalizes the Project Schemas. The advantage of this is that through the Common Schema development process, which also involves other B/Ds outside the project, the resultant schemas (i.e. Common Schemas) for those data elements are likely to be more reusable. However, the project team should expect that it might take considerable time for multiple B/Ds to reach consensus on the target Common Schemas. Project teams should make their own judgement on whether to submit Common Schema Creation / Change Requests during or after project development.

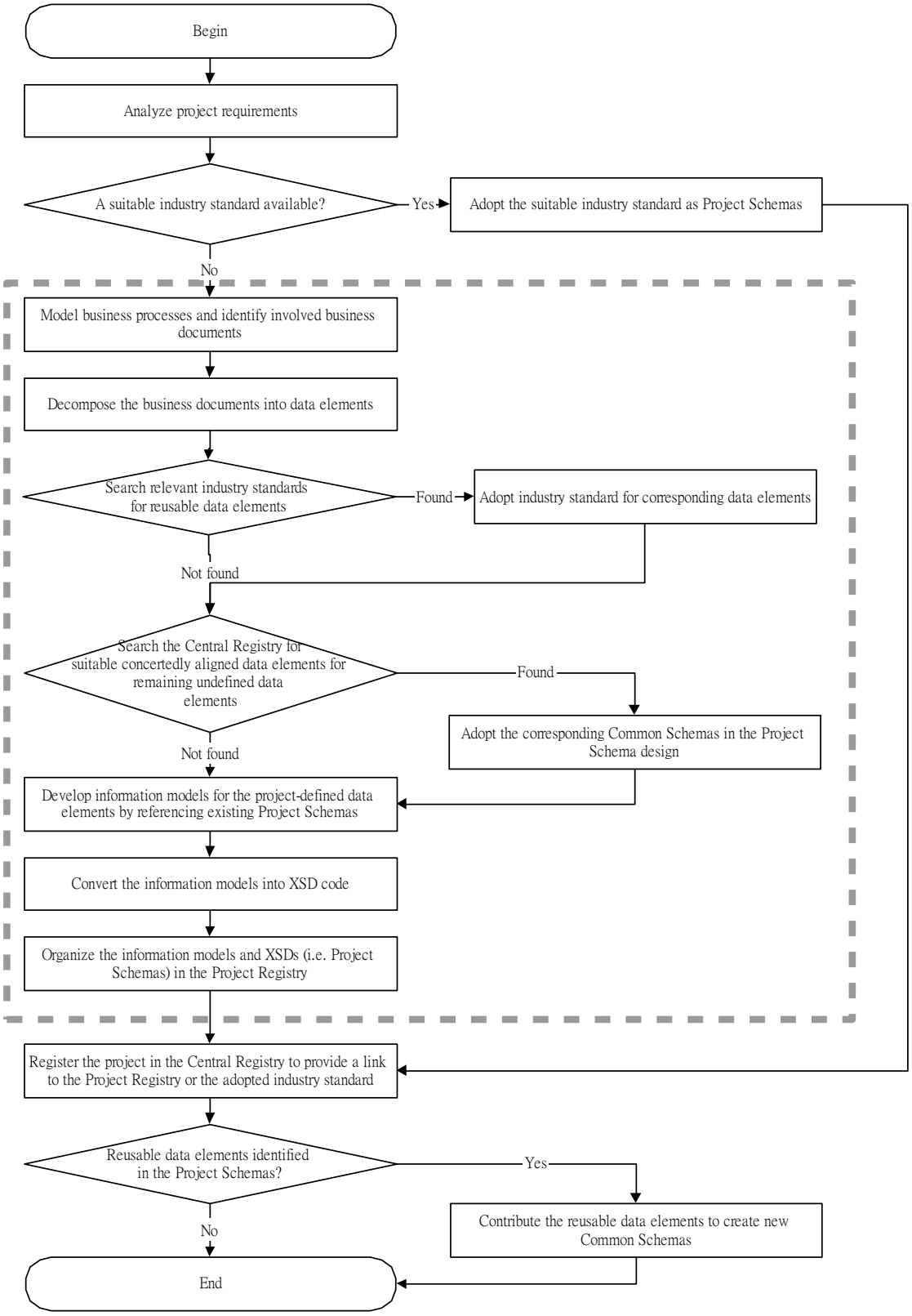


Figure 2-1: Project Schema design process.

3 Business Process Modelling (BPM)

3.1. Scope

This section provides the business process modelling (BPM) methodology for business analysts to model the business processes for joined-up services with a view to identifying the business documents involved in the interactions between business partners. The UN/CEFACT Modelling Methodology (UMM) refers to a business process as "the means by which one or more activities are accomplished in operating business practices".

When the business process of a joined-up service is considered, the business analyst needs to analyze how the business partners interact with each other. A series of activities conducted between two or more business partners is referred to as a **Business Collaboration** in the ebXML terminology. In other words, a Business Collaboration means a business process that involves multiple business partners.

This section recommends the use of UML activity diagrams to model business processes and in particular Business Collaborations. It also defines the concept of **Business Transaction** according to the ebXML business process framework, as an atomic component in a Business Collaboration realized by a single exchange of documents.

Most conventional SA&D methodologies provide techniques to analyse and model the conventional business processes that are internal to an organization. The techniques for modelling the Business Collaborations and Transactions in a joined-up service are provided in this section as supplementary techniques to be used with the conventional methodologies. For the purpose of schema design, the primary objective of using this methodology is to identify all business documents to be exchanged in the business process.

Business analysts may choose not to follow this BPM methodology if the SA&D methodologies they currently use can already serve the purpose of modelling business processes and identifying business documents for exchange.

3.2. Background of the BPM Methodology

The BPM methodology recommends the use of UML activity diagrams to model the business processes in a joined-up service. It is particularly useful for modelling Business Collaborations, i.e. business processes that involve multiple business partners.

The BPM methodology also recommends applying the concept of Business Transaction proposed in the ebXML business process framework to model an exchange of business documents (or simply called "documents"). By factoring every Business Collaboration into Business Transactions, business analysts can identify all documents that need to be exchanged in the Collaboration.

Derived from the UMM meta-model, the ebXML Business Process Specification Schema (BPSS) provides a solid framework with an XML-based language to specify business processes as Business Collaborations and Transactions. BPSS is aimed to specify business processes in a way that ebXML-based software engines can execute the BPSS specifications to automate business processes.

The BPM methodology is intended for business process modelling instead of business process execution. Therefore, it only adopts the high-level concepts of BPSS with the use of UML as good practices for analysing the business processes in a joined-up service. Besides, this methodology has also intentionally removed the ebXML-specific features to avoid tying the joined-up service implementation to the ebXML standard. However, if ebXML is used to implement a joined-up service, the process model captured with this methodology can be programmed into a BPSS specification for process execution.

3.2.1. Process Interoperability

To assure process interoperability, project teams need to well document their process-related agreement in a way that can be understood by all parties involved in the joined-up service. Developing such an agreement is not a simple linear activity, it usually takes many iterations to evolve, getting more complete as the domain experts and the business analysts learn and understand more about the domain.

The process-related aspects to be documented include, among other things, what parties are involved in a joined-up service, what specific services are provided by each party, what interactions can happen between the different parties, the business rules in performing the process (e.g. the payment terms and conditions), the manual procedures (e.g. exception handling procedures), the evidence required to prove a certain situation, the outcome of a certain interaction, etc.

In addition to describing the above aspects, project teams may use “Use Case” to capture the behavioural requirements of the business process. Each “Use Case” describes a specific way of using the joined-up service. Each “Use Case” constitutes a complete course of interactions (and their outcome) that will be triggered by a specific event. By documenting all the use cases, the complete functionality of the joined-up service can be defined.

This documentation that facilitates process interoperability, together with other collaboration agreements that facilitate technical interoperability and data interoperability, should form the basis for each party to automate their functions in the overall joined-up service.

3.3. BPM Methodology

3.3.1. Overview

A Business Collaboration is a kind of business process that consists of a series of business activities conducted between two or more business partners. A Collaboration is realized by a choreography of document exchanges between the involved partners’ business activities. In other words, a choreography specifies how document exchanges are sequenced to perform a Collaboration.

A Business Transaction is an abstraction of one exchange of documents and represents an atomic unit of work in a trading agreement between two business partners. A Business Transaction models a one-way or two-way Document Flows between two partners. A Document Flow carries one or more documents. A document is an atomic unit of business information exchange in a Business Transaction. Figure 3-1 summarizes the above concepts.

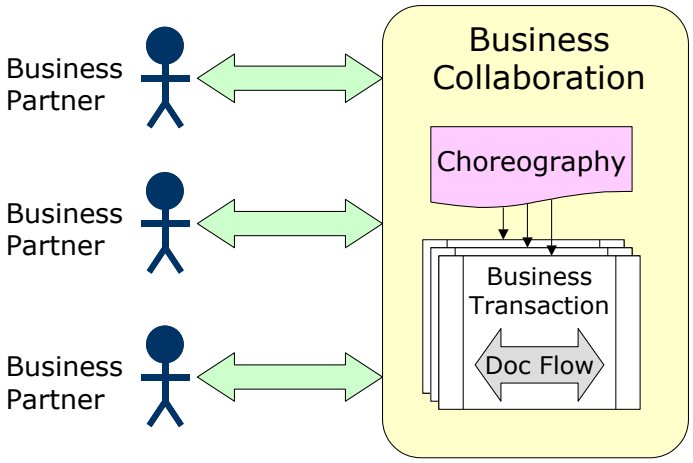


Figure 3-1: Key concepts of business process modelling.

This methodology can be applied to model general business processes and is particularly useful for modelling Business Collaborations. For the purpose of schema design, this methodology is specially aimed to identify all documents that need to be defined in XML Schema. By identifying all Business Transactions in a Business Collaboration, a business analyst can identify all documents that need to be exchanged in the Collaboration. For those identified documents that do not have suitable schemas from industry standards, the business analyst is required to follow the business information modelling and XSD development methodologies provided in subsequent sections to design project-specific schemas.

3.3.2. Business Collaboration

A Business Collaboration is a series of business activities conducted between two or more business partners. It is recommended to use a UML activity diagram to illustrate a Business Collaboration. Generally, the same technique can also be applied to modelling internal business processes. In relation to the modelling of business collaborations in a joined-up service, the objective is to identify all Business Transactions from that Collaboration.

Figure 3-2 shows an example of using a UML activity diagram to model a Business Collaboration. This Collaboration involves two business partners, “Buyer” and “Seller”, represented by two swimlanes. A **swimlane** is used to hold all activities performed by a business partner.

The Business Collaboration begins with a **start state** (a solid-circle symbol). Next, the “Buyer” decides whether to request for a price quote from the “Seller”. This decision is represented by a diamond symbol. If a price quote is needed, the “Buyer” sends a “Quote Request” message to the “Seller”. This activity is represented by the oval-shape **action state** symbol labelled “Send Quote Request”. The rectangle-shape **object** symbol denotes an object that flows from one activity to another, and the dotted arrow line denotes an **object flow**. (The solid arrow line denotes a **control flow**, i.e. transitions between action states.)

In the “Receive Quote Request” activity, the “Seller” receives the “Quote Request” message. Then, the “Seller” processes the message and prepares the requested quote in the “Prepare Quote” activity. After the quote is prepared, the “Seller” sends the “Price Quote” to the “Buyer” in the “Send Quote” activity.

The “Buyer” receives the “Price Quote” and decides whether to place an order. If a new order is needed, the “Buyer” prepares a purchase order in the “Prepare Order” activity. Then, the “Send Order” activity sends an “Order Request” to the “Seller”.

The "Seller" receives the "Order Request" in the "Receive Order" activity and processes the order in the "Process Order" activity. In the "Send Confirmation" activity, the "Seller" sends an "Order Response" to the "Buyer" to confirm whether the order request is accepted or rejected.

The "Buyer" receives the "Order Response" in the "Receive Confirmation" activity. Finally, the "Buyer" checks the response to see whether the order is accepted or rejected. An acceptance ends the Collaboration with a **final state** of success while a rejection gives a failure final state.

Drawing an activity diagram to model a Business Collaboration or a general business process is rather intuitive when the business analyst understands the basic concept and frequently-used symbols, such as start state, final state, action state, object, control flow, object flow, swimlane, etc. Some UML tools even provide sophisticated features to guide the user to draw well-formed activity diagrams. Readers who intend to obtain more information on this UML modelling technique may study the UML specification[4] and other reference materials.

[4] The UML specification is available at https://www.uml.org.

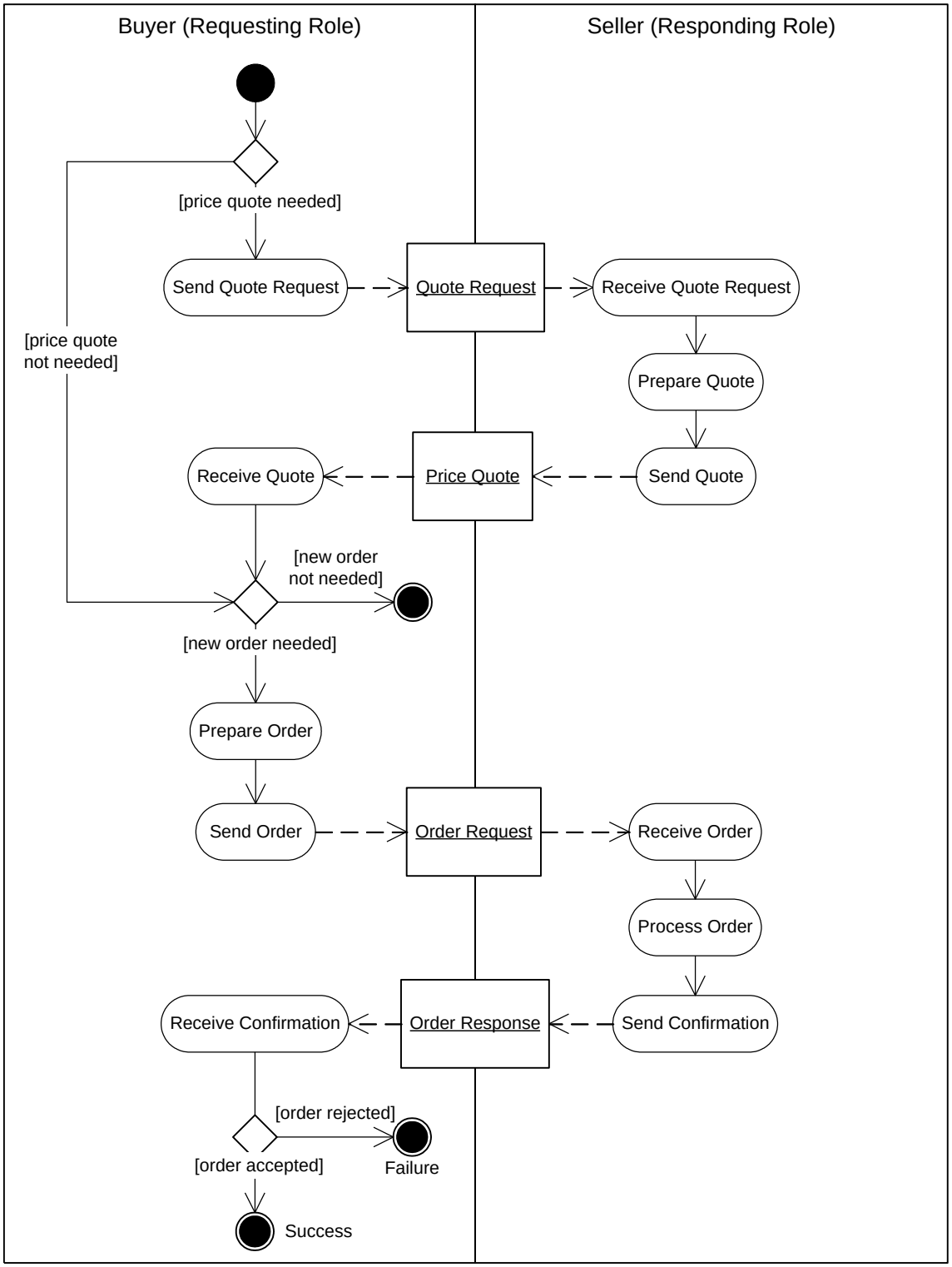


Figure 3-2: Example of using a UML activity diagram to model a Business Collaboration.

3.3.3. Business Transaction

A Business Transaction (BT) is the atomic unit of work in a trading arrangement between two business partners and is an abstraction of one exchange of documents.

A BT is conducted by a **Requesting Role** and a **Responding Role** that represent the two business partners. A BT is realized as one **Request Document Flow** and an optional **Response Document**

Flow. A **Document Flow (DF)** transmits a message, which packages one or more documents, between the Requesting Role, and the Responding Role. The Requesting Role initiates the Request DF to transmit the **Request Documents** to the Responding Role. After processing the Request Documents, the Responding Role may transmit the **Response Documents** to the Requesting Role via the optional Response DF.

Note that the Response Document Flow need not be synchronous and real-time. For example, the Response Document Flow may take place in a couple of days, particularly when human processing is required to process the Request Documents.

A BT is said to be one-way if it consists of only the Request DF but no Response DF. Otherwise, a BT is said to be two-way if it consists of both the Request DF and the Response DF. In a two-way BT, the Response DF can be either a Positive Response DF or a Negative Response DF. A Positive Response DF signifies that the Responding Role has accepted the BT and the BT is said to be a success. On the contrary, a Negative Response DF signifies that the Responding Role has rejected the BT and the BT is said to be a failure.

The above concepts are summarized in Figure 3-3.

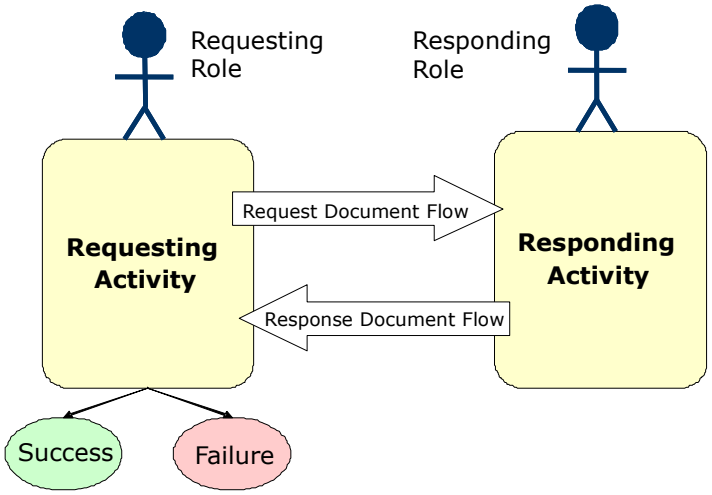


Figure 3-3: Business Transaction semantics.

3.4. Example

Figure 3-4 is the same activity diagram shown in Figure 3-2 while it highlights how BTs can be identified in a Business Collaboration. The two regions bounded by dotted rectangles in the diagram are abstracted into two BTs, namely “Request Quote” and “Create Order” in form of BT model shown in Figure 3-3. In the “Create Order” BT, the “Buyer” serves the Requesting Role while the “Seller” serves the Responding Role. The transmission of the “Order Request” message is the Request Document Flow and the transmission of the “Order Response” message is the Response Flow.

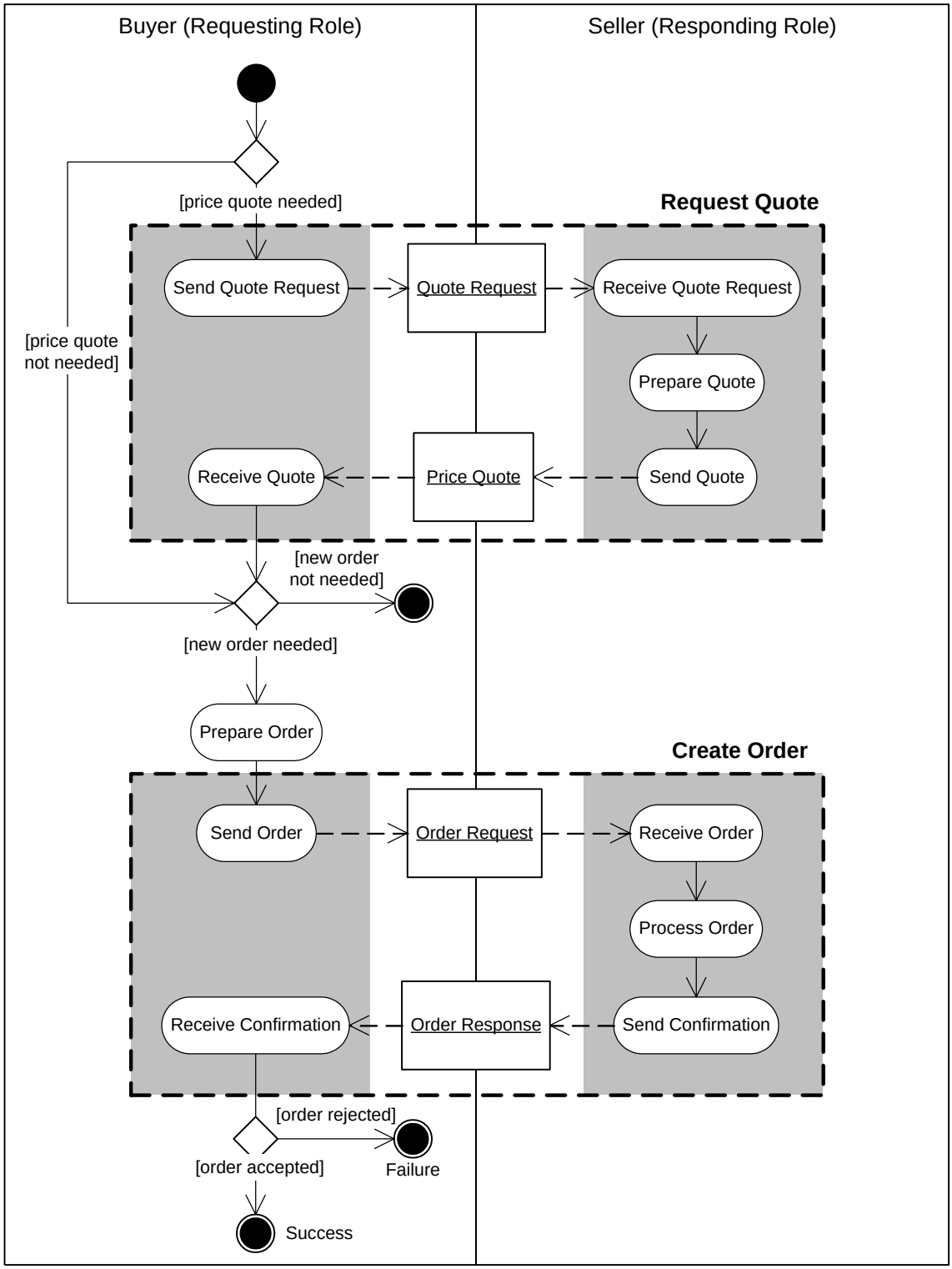


Figure 3-4: Example of 2 Business Transactions for “Request Quote” and “Create Order”.

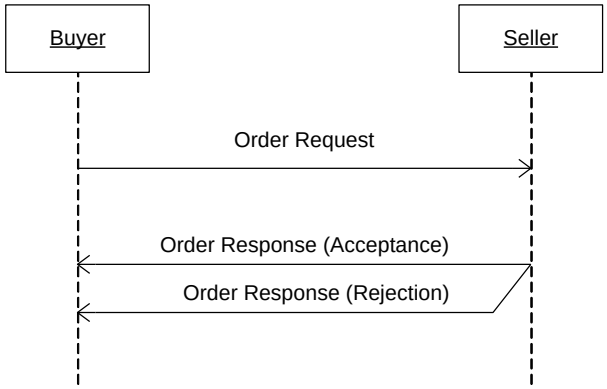


Figure 3-5: UML sequence diagram of the "Create Order" Business Transaction example.

A UML sequence diagram can also be used to present a simplified view of the Business Transaction as shown in Figure 3-5, in which only the DFs are indicated. The sequence diagram in Figure 3-5 shows that the "Create Order" BT is a two-way BT, consisting of both the Request DF and the Response DF. The "Buyer" first sends an "Order Request" message (Request DF) to the "Seller". After the "Seller" has processed the order request, it replies to the "Buyer" with an "Order Response" message that indicates either an acceptance" (Positive Response DF) or a rejection (Negative Response DF) for the order.

Table 3-1 shows the documents to be exchanged in the "Create Order" BT. The "Order Request" message (Request DF) will package two Requesting Documents, namely "Purchase Order" and "Price Quote", and the "Order Response" message (Response DF) will package two Responding Documents namely, "Order Confirmation" and "Purchase Order". The "Purchase Order" document in the "Order Response" message is the original purchase order document in the "Order Request" message. On the other hand, the "Order Confirmation" document will be used to indicate whether the "Order Response" message is a Positive Response DF or a Negative Response DF.

As a result, a total of three documents (i.e. "Purchase Order", "Price Quote" and "Order Confirmation") that need to be exchanged in this BT have been identified. By repeating this exercise to identify and analyze all BTs in a joined-up service, a business analyst can discover all documents for exchange.

Table 3-1: Specification of a Business Transaction example.

Business Transaction	
BT Name:	Create Order
Request Document Flow	
Requesting Documents:	1. Purchase Order: a request to place an order 2. Price Quote: the price quote referenced by the order request
Response Document Flow	
Positive Responding Documents:	1. Order Confirmation: an indication that the order request is accepted 2. Purchase Order: the original order request
Negative Responding Documents:	1. Order Confirmation: an indication that the order request is rejected 2. Purchase Order: the original order request

4 Business Information Modelling (BIM)

4.1. Scope

This section provides the business information modelling methodology for business analysts to model business information that needs to be exchanged with other parties. Business analysts should apply this methodology to capture the requirements on the business information that needs to be exchanged in joined-up services and specify the requirements as information models. The information models shall be used as the input for programmers to develop XSDs or for software tools (e.g. the modelling spreadsheets) to generate XSDs using the mechanism provided in Section 5.

This BIM methodology is based on industry good practices for data and schema standardization, including **ISO 11179**, **Core Components Technical Specification (CCTS)**[5], and **Universal Business Language (UBL)**, for construction of information models. The objectives of applying this BIM methodology are as follows:

- To model and specify data elements in an unambiguous, complete, organized and systematic manner to enhance their shareability; and

- To provide a methodology for programmers to develop quality, consistent, and reusable Project Schemas through a systematic mechanism.

Based on the ISO 11179 standard, this methodology identifies a data element in its three parts: the **object class**, the **property**, and the **representation**. (See Section 4.3.1.) It has also adopted CCTS and introduces the following types of information models:

- **Business Document**: a model that represents an electronic document as a unit for business information exchange; a root Aggregate Business Information Entity is used to provide the representation of the document.

- **Aggregate Business Information Entity (ABIE)**: a model that represents an object class and aggregates Basic and Association Business Information Entities as the properties.

- **Association Business Information Entity (ASBIE)**: a model that represents a complex property in an object class.

- **Basic Business Information Entity (BBIE)**: a model that represents a singular property in an object class.

- **Core Component Type (CCT)**: a model that provides the basic data structure to realize the representation of a Basic Business Information Entity.

This BIM methodology is optional to follow when designing data exchange interfaces of cross departmental systems in order to comply with this Guide.

[5] This BIM methodology is primarily adapted from the Core Components Technical Specification.

4.2. Background of BIM Methodology

This methodology provides a consistent and systematic way to (1) describe the characteristics of data elements as information models, and (2) decompose a complex document into modular components or data elements to enhance reusability of data elements. The ISO 11179 standard provides the principles on describing data elements while the CCTS provides the conceptual models to decompose complex documents into modular and reusable data elements. This methodology also references the UBL methodology to realize the CCTS conceptual models as concrete models for development of XSDs.

4.2.1. ISO 11179

The ISO 11179 standard, specification and standardization of data elements, serves as the framework for this methodology to describe data elements in an unambiguous and consistent way. Among the six parts of the standard, Part 1, Part 3, Part 4, and Part 6 are particularly applicable. Part 1[6] provides the high-level principles that suggest a data element be described in its three parts: the object class, the property, and the representation. Part 3[7] suggests the concepts and guidelines for specifying attributes of data elements. Part 4[8] provides additional rules for constructing definitions for data elements. Part 6[9] provides guidelines on registration of data elements for sharing.

4.2.2. Core Components Technical Specification (CCTS)

The Core Components Technical Specification (CCTS) provides the approach to decompose complex documents into modular and reusable data elements and to document these data elements as Business Information Entities.

CCTS suggests that business information be modelled as Core Components and Business Information Entities. A **Core Component (CC)** is a building block for creating a semantically correct and meaningful information exchange package. It contains only the information pieces necessary to describe a specific concept. CCs are business context neutral.

A **Business Information Entity (BIE)** is a piece of business data or a group of pieces of business data with a unique business semantic definition. The key differentiator between a CC and a BIE is the concept of business context. When a CC is used in a real business situation (business context), it serves as the basis of a BIE. The BIE is the result of refining a CC for a specific business context. For example, a generic “Postal Address” can be modelled as a CC. A BIE can be modelled for the “Hong Kong Postal Address” to have all properties in the “Postal Address” CC except the “Postal Code” property. In this case, the geopolitical context is “Hong Kong SAR”.

CCTS is a syntax neutral information modelling approach that can be used to generate different syntaxes of electronic messages, e.g. EDIFACT and XML. Also because of this, it does not provide any mechanism to generate a specific syntax schema, e.g. XML Schema, from the information models. When an organization applies CCTS to design electronic message formats, it must develop a syntax-binding mechanism to convert CCTS information models to concrete message formats. For example, this Design Guide has presented a syntax-binding mechanism (i.e. XSD development mechanism) in Section 5 to convert the information models into XSDs. Figure 4-1 shows how the syntax neutral models defined in CCTS are related to concrete electronic message formats.

[6] ISO 11179-1: Framework
[7] ISO 11179-3: Basic attributes of data elements
[8] ISO 11179-4: Rules and Guidelines for the Formulation of Data Definitions
[9] ISO 11179-6: Registration of data elements.

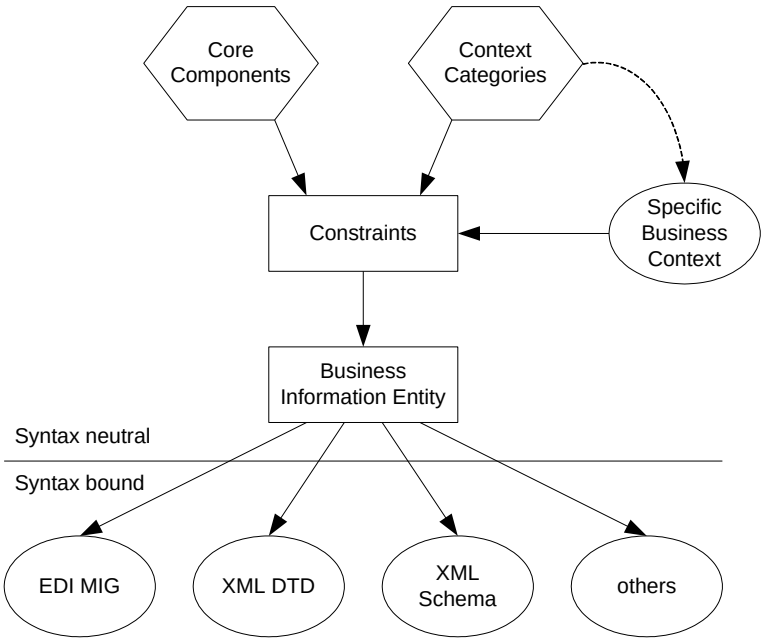


Figure 4-1: Application of CCTS.

The Copyright Statement of the CCTS is as follows :

4.2.3. Universal Business Language

The Universal Business Language (UBL) has developed the guidelines to apply CCTS and provide its syntax-binding mechanism to produce its XML Schema library. The UBL methodology has also simplified the CCTS approach by considering a CC being a context-free BIE (or a BIE without business context). This way, modellers need not distinguish between CCs and BIEs and only need to model BIEs. This has significantly simplified the understanding and the use of CCTS specifically for XML Schema design.

The business information modelling methodology provided in this section follows the UBL approach by substituting CCs by context-free BIEs. This BIM methodology is differentiated from the UBL methodology in its explicit segregation of business analyst and programmer duties. The UBL methodology is designed for use by the UBL Library Content team, which is a specialized team with both business and technical expertise. This BIM methodology explicitly divides the schema design process into different sub-processes. All tasks in each sub-process are specially designed for either business analysts or programmers. This way, non-technical business analysts can apply the methodology only to construct the information models without programming XML Schema. Then, programmers can use the models to develop XSDs using the syntax-binding mechanism provided in Section 5.

4.3. BIM Methodology

4.3.1. Object Class, Property, and Representation

Based on ISO 11179-1, a data element is represented in the following three parts:

- **Object class**: a set of ideas, abstractions, or things in the real world that can be identified with explicit boundaries and meaning, and whose properties and behaviour follow the same rules.
- **Property**: a peculiarity common to all members of an object class.
- **Representation**: how the data is represented, i.e. the combination of a value domain, data type, and, if necessary, a unit of measure or a character set.

Object classes are the things about which data is collected and stored. Examples of object classes are "Car", "Person", "Household", and "Employee". A Business Document (e.g. "Purchase Order") can also be modelled as an object class.

Properties are attributes humans use to distinguish or describe objects. Examples of properties are "Colour", "Model", and "Price" of a "Car", and "Name", "Address", and "Income" of a "Person".

The most important aspect of the representation part of a data element is the value domain. A value domain is a set of permissible values for a data element. A data element may have different representations. For example, the data element representing "Annual Household Income" may be a monetary amount. A monetary amount carries a non-negative decimal number as its primary content and a currency (e.g. "Hong Kong Dollar") as its supplementary content. On the other hand, the "Annual Household Income" may have another representation, which is a code indicating one of the predefined income levels, e.g. "high", "medium", and "low".

Figure 4-2 is the class diagram to illustrate the relationship among the concepts of object class, property, and representation. That is, an object class has one or more properties, and each property can have multiple representations.

For more information on the above concepts, please refer to ISO11179-1.

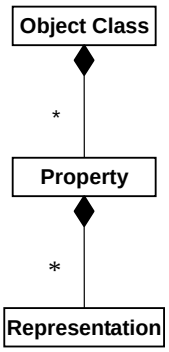


Figure 4-2: Relationship among the object class, the property, and the representation.

4.3.1.1. Singular Property and Complex Property

A property of an object class can be a singular property or a complex property. A singular property is a single piece of information that is bound to a data type and a value domain as its representation. A

singular property may carry some supplementary information that gives additional definition to the representation. For example, “Colour” can be a singular property of an object class called “Car”. The representation of the “Colour” property can be a “Code”. The “Code” has a character string as its data type and a code list as its value domain. One piece of the supplementary information may be the source of that code list (e.g. the URL that points to a Webpage that defines the code list). This example is illustrated graphically in Figure 4-3.

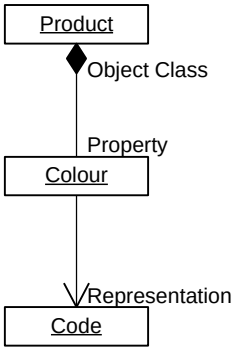


Figure 4-3: Singular property example.

A complex property of an object class is a collection of information pieces represented by another object class. For example, “Delivery Address” is a complex property of an object class called “Purchase Order”. The “Delivery Address” property itself can be represented by another object class called “Physical Address”, which has other properties such as “Street”, “City”, and “Country”. This example is illustrated graphically in Figure 4-4.

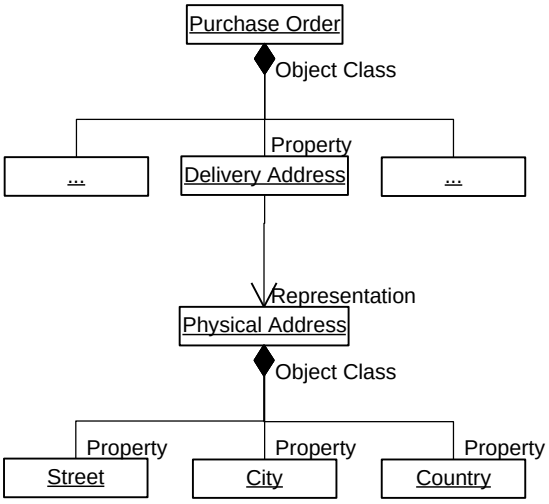


Figure 4-4: Complex property example.

4.3.2. Modelling a Business Document

A **Business Document** (or simply called “Document”) is a unit of business information exchange in a Business Transaction. A Document is an organization of data elements, each of which represents a piece of business information for exchange. A BIE is used to specify a data element described in ISO11179 standard. A Document can be organized as a hierarchy of data elements and in turn can be modelled as a hierarchy of BIEs.

There are three types of BIEs: **Aggregate BIE (ABIE)**, **Basic BIE (BBIE)**, and **Association BIE (ASBIE)**. An ABIE models an object class and aggregates one or more BBIEs or ASBIEs as the properties of that object class.

A BBIE models a singular property. An appropriate **Core Component Type (CCT)** is used to provide the data structure for the representation of the property. The CCT is selected from a registered set of pre-defined CCTs in the Central Registry.

An ASBIE models a complex property. Since the complex property is represented by another object class, the ASBIE is associated with another ABIE which models that object class. An ASBIE provides the association between an aggregating ABIE and an aggregated ABIE.

Figure 4-5 illustrates an example of using a hierarchy of BIEs to model a "Purchase Order" Document. The "Purchase Order" Document itself is represented by a root ABIE. That root ABIE contains a number of BBIEs, such as "Delivery Date", and ASBIEs, such as "Delivery Address". The "Delivery Date" BBIE is associated with a CCT called "Date". The "Delivery Address" ASBIE is associated with another ABIE called "Physical Address". The "Physical Address" ABIE contains other BBIEs, such as "Street", "City", and "Country", each of which is also associated with an appropriate CCT.

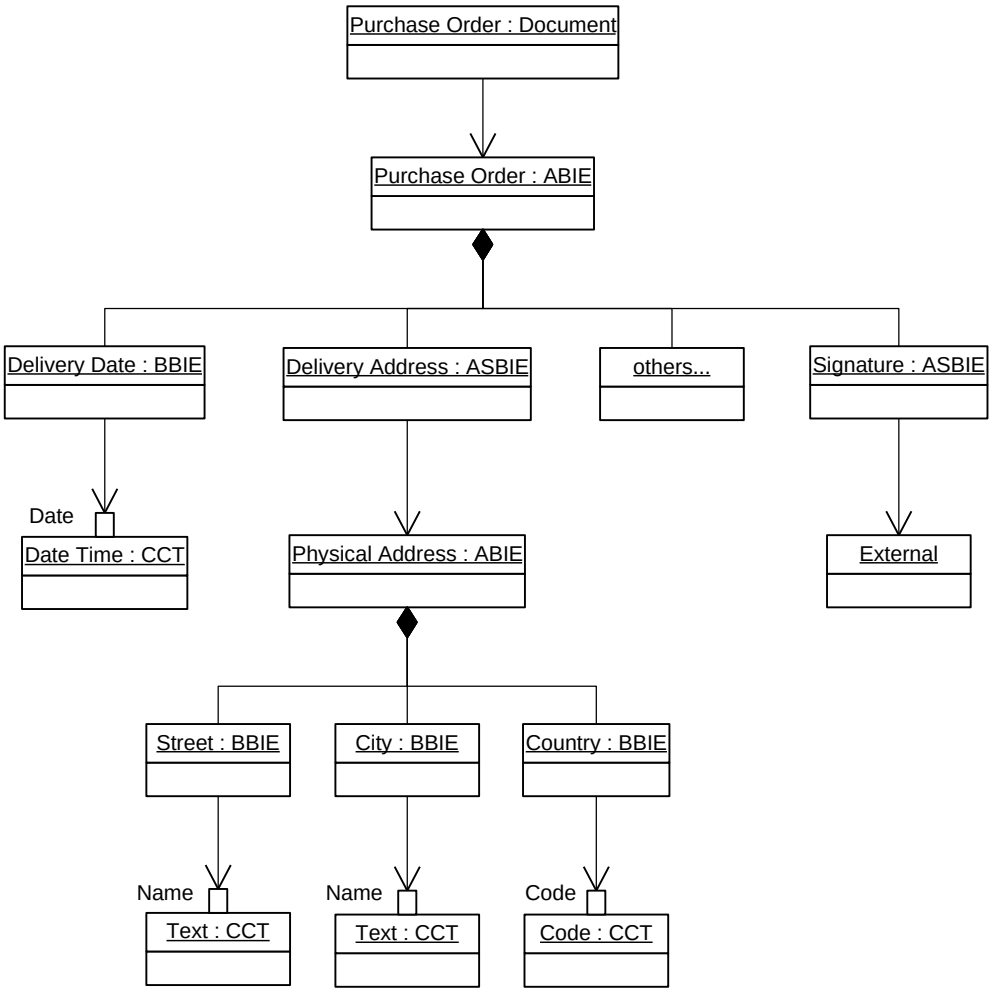


Figure 4-5: Example of modelling a purchase order document.

The illustration in Figure 4-5 is based on the UML object diagram. Each object (a document or a BIE or a CCT) is represented by a rectangle. The **composition association** (a line with a solid diamond) represents an aggregation of an ASBIE or BBIE model in an ABIE model. A **navigable association** (an arrow line) represents a relation between two information models, i.e. between a BBIE and a CCT, between an ASBIE and a Representation ABIE or an Externally-defined Entity (shown as 'External' in the above figure), or between the Document and the root ABIE. The **qualifier** (a small rectangle on

the object symbol) attached to a CCT object specifies the Representation Term (see Section 4.3.3.1) that is referenced when the CCT is used by a BBIE.

4.3.3. Core Component Type

A CCT is a model that provides the basic data structure to realize the representation of a singular property in an object class. A CCT has one Content Component and a specified list of Supplementary Components. The Content Component carries the actual content of the data element. A Supplementary Component gives an extra definition to the Content Component. For the "Amount" CCT that represents a monetary amount, the Content Component carries the number of the monetary units (e.g. "100"). A Supplementary Component is the "Currency Code", which defines the currency of the monetary units (e.g. "HKD" [Hong Kong Dollar]). A Supplementary Component can be mandatory or optional in the CCT. For example, the "Currency Code" is a mandatory Supplementary Component in the "Amount" CCT.

A recommended list of CCTs and corresponding Supplementary Components is given in Table 4-1. These recommended CCTs are created by customizing the CCTs provided by CCTS and UBL to meet the needs of modelling data elements for HKSARG joined-up services. This CCT list is registered and maintained in the Central Registry and may be updated to meet current modelling needs.

Table 4-1: Core Component Types and corresponding Supplementary Components.

Core Component Type Name	***Definition***	***Supplementary Components***	***Mandatory/ Optional***	***Definition***
Amount	A number of monetary units specified in a currency where the unit of currency is explicit or implied.	Currency Code	Mandatory	A 3-letter alphabetic currency code in the UN/ECE Rec. 9 code list.
		Code List Version	Optional	The version of the UN/ECE Rec. 9 code list.
Binary Object	A set of finite-length sequences of binary octets.	Character Set Code	Optional	The character set of the binary object if the mime type is text. Reference IETF RFC 2045, 2046, 2047.
		Encoding Code	Optional	The decoding algorithm of the binary object. Reference IETF RFC 2045, 2046, 2047.
		Filename	Optional	The filename of the encoded binary object. Reference IETF RFC 2045, 2046, 2047.
		Format	Optional	The format of the binary content.
		Mime Code	Optional	The mime type of the binary object. Reference IETF RFC 2045, 2046, 2047.
		Object URI	Optional	The Uniform Resource Identifier that identifies where the binary object is located.
Code	A character string (letters, figures or symbols) that for brevity and/or language	Agency ID	Optional	The identification of the agency that maintains the code list.

Core Component Type Name	*Definition*	*Supplementary Components*	*Mandatory/ Optional*	*Definition*
	independence may be used to represent or replace a definitive value or text of an attribute.	Agency Name	Optional	The name of the agency that maintains the code list.
		Code List ID	Optional	The identification of the code list, e.g. the URL of a source that publishes the code list.
		Code List Name	Optional	The name of the code list.
		Code List Version	Optional	The version of the code list.
		Code Name	Optional	The textual equivalent of the code content.
Date Time	A particular point in the progression of time.			
Identifier	A character string to uniquely identify and distinguish one instance of an object in an identification scheme from all other objects in the same scheme.	Agency ID	Optional	The identification of the agency that maintains the identification scheme.
		Agency Name	Optional	The name of the agency that maintains the identification scheme
		Scheme ID	Optional	The identification of the identification scheme, e.g. the URL of a source that publishes the identification scheme.
		Scheme Name	Optional	The name of the identification scheme.
		Scheme Version	Optional	The version of the identification scheme.
Indicator	A list of two mutually exclusive Boolean values that express the only possible states of a Property.			
Measure	A numeric value determined by measuring an object along with the specified unit of measure.	Code List Version	Optional	The version of the UN/ECE Rec. 20 measure unit code list.
		Unit Code	Mandatory	The unit code as defined in UN/ECE Rec. 20.
Numeric	Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.			
Quantity	A number of non-monetary units possibly including fractions.	Agency ID	Optional	The identification of the agency that maintains the quantity unit code list.
		Agency Name	Optional	The name of the agency which maintains the quantity unit code list
		Code List ID	Optional	The identification of the quantity code list, e.g. the URL of a source that publishes the code list.
		Code List Version	Optional	The version of the quantity code list.

Core Component Type Name	*Definition*	*Supplementary Components*	*Mandatory/ Optional*	*Definition*
		Unit Code	Optional	The quantity unit code.
Text	A character string (i.e. a finite set of characters) generally in the form of words of a language.	Language Code	Optional	The code of the language used in the corresponding text as defined in ISO 639.
Electronic Address	An address for electronic communication, such as email address, URL.	Protocol Code	Optional	The code that specifies the communication protocol used. Reference Official IANA Registry of URI Schemes.

4.3.3.1. Representation Term of Core Component Type

A CCT has one or more permissible Representation Terms (RTs). The particular Representation Term referenced determines the form in which the CCT is used. For example, the "Date Time" CCT has three RTs: "Date Time", "Date", and "Time". When a CCT is used by referencing the "Date Time" RT, the Content Component carries the information on both date and time. When the "Date" RT or "Time" RT is referenced, the Content Component carries only the date part or only the time part respectively. In other words, the selection of an RT can affect the value domain of the Content Component. However, this does not affect the list of Supplementary Components. A CCT has a fixed list of Supplementary Components regardless of which RT is referenced.

The Content Component is also bound to a primitive data type depending on which RT is referenced. The Content Component for the "Date Time" CCT is bound to the type "Date Time" when the "Date Time" RT is referenced. It is bound to the type "Date" or "Time" when the "Date" RT or the "Time" RT is referenced respectively. It is possible that the Content Component can be bound to the same primitive data type for all RTs. For example, the Content Component of the "Numeric" CCT is bound to the "Decimal" type for all RTs: "Numeric", "Percent", "Rate", and "Value". The available primitive data types for the Content Component are listed in Table 4-2. The permissible RTs of each CCT and the corresponding primitive data types for the Content Component are listed in Table 4-3.

Table 4-2: Primitive data types for the Content Component.

Primitive Data Type for Content Component	*Definition*
Binary	Binary data.
Boolean	Binary-valued logic of true or false.
Date	A specific date.
Date Time	A specific date and time.
Time	A specific time of day.
Decimal	A decimal number.
Integer	An integer
String	A character string.
URI	A Uniform Resource Identifier Reference (URI).

Table 4-3: Permissible Representation Terms of Core Component Types.

Core Component Type Name	*Permissible Representation Term*	*Primitive Data Type of Content Component*
Amount	Amount	Decimal
Binary Object	Binary Object	Binary
	Graphics	
	Picture	

Core Component Type Name	*Permissible Representation Term*	*Primitive Data Type of Content Component*
	Sound	
	Video	
Code	Code	String
Date Time	Date	Date
	Date Time	Date Time
	Time	Time
Identifier	Identifier	String
Indicator	Indicator	String
	Boolean	Boolean
Measure	Measure	Decimal
Numeric	Numeric	Decimal
	Percent	
	Rate	
	Value	
Quantity	Quantity	Decimal
	Count	Integer
Text	Name	String
	Text	
Electronic Address	Electronic Address	String
	URI	URI

4.3.4. Basic Business Information Entity

A BBIE is an information model that represents a singular property of an object class. A BBIE is defined based on an Object Class Term, a Property Term, and the Representation Term. These three Terms name the object class, the property, and the representation respectively. The Object Class Term of a BBIE is the same as the Object Class Term of the Aggregating ABIE which aggregates the BBIE. The Property Term is a meaningful name identifying the property. A BBIE uses a Core Component Type (CCT) to provide the representation and references one of the permissible RTs of CCT. The Representation Term is the name of the referenced RT.

A BBIE may apply some format restrictions on the Content Component of the CCT to constrain the value domain of the representation. A BBIE may also supply a list of possible values for each Supplementary Component.

In the example shown in Figure 4-5, the BBIE labelled “Delivery Date” models the first property, which is a singular property, of the “Purchase Order” object class. The Property Term of this BBIE is “Delivery Date” and the Object Class Term is “Purchase Order”. This BBIE uses the “Date Time” CCT and references the “Date” RT, which provides the Representation for this singular property. Similarly, the BBIE labelled “Street” models the singular property of a street description of the “Physical Address” object class. The Property Term and the Object Class Term are thus respectively “Street” and “Physical Address”. This BBIE uses the “Text” CCT, referencing the “Name” RT, as the representation of the property.

4.3.4.1. Format Restriction on Content Component

A set of format restrictions is provided to constrain the value domain of the Content Component of a CCT that provides the representation in the BBIE. A format restriction is applicable to a specific primitive data type as listed in Table 4-2.

For example, the "Length" format restriction can only be applied to the "String" primitive data type. The available format restrictions are listed in Table 4-4. For example, the possible format restrictions for the Hong Kong Identity Card Number may be applied as shown in Table 4-5.

Table 4-4: Format restrictions for different Primitive Data Types of Content Components.

Format Restriction	***Definition***	***Primitive Data Types***	***Remarks***
Expression	The restricted combination of characters to represent the string value.	• String	A textual description or a regular expression can be used to specify this format restriction.
Length	The required length of the string.	• String	This format restriction shall not be used in combination with the Minimum Length and Maximum Length Format restrictions.
Minimum Length	The minimum length of the string.	• String	This format restriction shall not be used in combination with the Length Format restriction.
Maximum Length	The maximum length of the string.	• String	This format restriction shall not be used in combination with the Length Format restriction.
Enumeration	The exhaustive list of the allowed values of the string or the URI.	• String • URI	
Total Digits	The maximum number of digits to be used in the numeric value.	• Decimal • Integer	
Fractional Digits	The maximum number of fractional digits to be used in the decimal value.	• Decimal	
Minimum Inclusive	The lower limit of the range of the allowed values of the numeric value, or date time. The lower limit is also an allowed value.	• Date Time • Date • Time • Decimal • Integer	This format restriction shall not be used in combination with the Minimum Exclusive format restriction.
Maximum Inclusive	The upper limit of the range of the allowed values of the numeric value, or date time. The upper limit is also an allowed value.	• Date Time • Date • Time • Decimal • Integer	This format restriction shall not be used in combination with the Maximum Exclusive format restriction.
Minimum Exclusive	The lower limit of the range of the allowed values of the numeric value, or date time. The lower limit is not an allowed value.	• Date Time • Date • Time • Decimal • Integer	This format restriction shall not be used in combination with the Minimum Inclusive format restriction.
Maximum Exclusive	The upper limit of the range of the allowed values of the numeric value, or date time. The upper limit is not an allowed value.	• Date Time • Date • Time • Decimal • Integer	This format restriction shall not be used in combination with the Maximum Inclusive format restriction.

Table 4-5: Possible format restrictions to specify Hong Kong Identity Card Number.

Data Type:	String
Format Restriction	***Value***
Expression	The first one or two characters are alphabets. The next 6 characters are numeric. The next character is "(". The next character is an alphanumeric. The last character is ")".
Maximum Length	11

Data Type:	String
Format Restriction	***Value***
Minimum Length	10

4.3.4.2. Possible Values on Supplementary Component

An optional list of possible values may be supplied as the valid values for a Supplementary Component of the CCT that provides the representation in the BBIE. The Supplementary Component may have one optional default value selected from the possible value list. When the Supplementary Component is not supplied, the default value shall be used.

For example, a BBIE that represents the height of a person may use the "Measure" CCT as the representation. It is possible to constrain the measure unit to be "meter" or "centimeter", where "meter" is the default unit. In this case, the possible values for the "Unit Code" Supplementary Component are "MTR" and "CMT" (i.e. measure unit codes for "meter" and "centimeter" in the UN/ECE Rec. 20 code list), and the default value is "MTR".

4.3.5. Association Business Information Entity

An ASBIE represents a complex property of an object class. An ASBIE is defined based on an Object Class Term, a Property Term, and a Representation Term. The three Terms name the object class, the property, and the representation respectively.

An ASBIE provides the association between two ABIEs, or between an ABIE and an Externally-Defined Entity. An Externally-Defined Entity is not modelled in BIM but is defined by an external schema, such as an industry standard (e.g. W3C XML Signature, W3C XHTML). (For details, see Section 5.)

When an ASBIE is used to associate two ABIEs. One ABIE is the Aggregating ABIE, which aggregates the ASBIE as the complex property. The other ABIE is the Representation ABIE, which provides the representation for the complex property. The Object Class Term uses the Object Class Term of the Aggregating ABIE. The Property Term is assigned with a meaningful name for the property represented by the ASBIE. The Representation Term uses the Object Class Term of the Representation ABIE.

In the example shown in Figure 4-5, the second property, labelled "Delivery Address", of the "Purchase Order" Aggregating ABIE is a complex property represented by an ASBIE. The ASBIE is associated with a Representation ABIE called "Physical Address" (Object Class Term) that provides the representation of this complex property. The Object Class Term, Property Term, and the Representation Term of that ASBIE are thus "Purchase Order", "Delivery Address", and "Physical Address" respectively.

When an ASBIE is used to associate an ABIE and an Externally-Defined Entity, the ABIE is the Aggregating ABIE, which aggregates the ASBIE as the complex property. The Externally-Defined Entity provides the representation of the complex property. The Object Class Term uses the Object Class Term of the aggregating ABIE. The Property Term is assigned with the name of the property represented by the ASBIE. The Representation Term should be assigned with "External".

In the example shown in Figure 4-5, the last complex property, labelled "Signature", of the "Purchase Order" Aggregating ABIE is represented by an ASBIE. The "Signature" property is associated with a data structure that encodes a digital signature for authenticating the issuer of the "Purchase Order". An Externally-Defined Entity is used to encode the digital signature, of which the schema is externally defined by the XSD of the W3C XML Signature. The Object Class Term, Property Term, and the

Representation Term of that ASBIE are thus “Purchase Order”, “Signature”, and “External” respectively.

4.3.6. Aggregate Business Information Entity

An ABIE represents an Object Class and is defined based on an Object Class Term. An ABIE aggregates one or more Basic Business Information Entities (BBIEs) and Association Business Information Entities (ASBIEs) that represent the properties of the object class.

Each aggregated BBIE or ASBIE has a cardinality that specifies the number of its occurrences in the ABIE. For example, the cardinality for a line item in a purchase order can have a range of 1-99, meaning that the purchase order has at least one and at most 99 line items.

An ABIE shall never aggregate—directly or at any nested level—an ASBIE as a mandatory (i.e. cardinality greater than 1) property that is associated with this ABIE itself. Otherwise, an infinite number of nested levels of referencing this ABIE would be created.

In the example shown in Figure 4-5, “Physical Address” is an ABIE. It aggregates three BBIEs, namely “Street”, “City” and “Country”. Each aggregated BBIE has a cardinality of 1.

4.3.7. Business Document

A Business Document (or Document) represents a basic unit of information exchange in a Business Transaction. A root Aggregate Business Information Entity (ABIE) is identified to provide the representation of the Document. The root ABIE directly and indirectly aggregates a hierarchy of BIEs that collectively represent all data elements included in the Document. A Business Document has a Document name, which is equivalent to the Object Class Term of the root ABIE.

In the example shown in Figure 4-5, the Business Document is called “Purchase Order” and is associated with the root ABIE called “Purchase Order” (as the Object Class Term). This ABIE aggregates a number of BBIE, such as “Delivery Date”, and ASBIE, such as “Delivery Address”, as the properties for the “Purchase Order” object class.

4.3.7.1. Graphical Presentation of a Document Model

It is useful to present the model of a Business Document graphically so that the project team can visualize the structural relations among the information models to ease modelling and understanding. There can be many possible formats and styles to graphically present a document model provided that the presentation is consistent with the BIM methodology. One recommended way is to use the UML class diagram or the object diagram to present a Document model.

Figure 4-5 is an example of using a UML object diagram to present the “Purchase Order” document model.

4.3.8. Different Languages of BIEs

One single BIE can be designed to represent information in two or more different languages when there is no need to have different representations, e.g. in terms of the data structure and format, restrictions for different languages of information. For example, the “Organization Name” BIE can be

modelled as a “Text” CCT of maximum length of 100 characters to represent both Chinese and English names.

However, there are cases that BIEs of two different languages require different representations. For example, the “Chinese Person Name” can be modelled as an ABIE that aggregates two BBIEs of the Text CCT, namely “First Name” (maximum length of 2 characters) and “Last Name” (maximum length of 2 characters). On the other hand, the “English Person Name” can be modelled as another ABIE that aggregates three BBIEs of the Text CCT, namely “First Name”, “Last Name”, and “Middle Name”, each of which have a maximum length of 25 characters.

Note that in XML, the maximum length of a piece of text specifies the maximum number of characters (either English or Chinese) that the data element can accommodate, regardless of the character encoding (BIG5 or UTF-8) used.

4.3.9. Multiple Versions of BIEs

Multiple versions of BIEs may exist because of the evolution of information models. A BIE is assigned with a version number, which identifies its version. (See Section 4.4.1.) Therefore, the aggregation of the BBIEs by an ABIE or the association between an aggregating ABIE and its Representation ABIEs by an ASBIE must refer to specific versions of BIEs. (See Section 4.4.1.) This can be done by referencing either the Dictionary Entry Unique Identifiers (UIDs) or the Dictionary Entry Names together with the Version identifiers of the BIEs involved in an aggregation or association. Similarly, a Business Document should also be associated with the root ABIE by UID.

4.3.10. Business Context

Every BBIE, ABIE, or Document has a business context that determines what business situation the information model should be used. The business context can be specified through assigning values (Context Values) to a suggested list of Context Categories listed in Table 4-6. This suggested list is formed by customizing the CCTS Context Categories. Each Context Category may be assigned multiple Context Values. For an ABIE or Business Document, the default Context Values listed in Table 4-6 are implicitly used for any Context Category not explicitly assigned any value. For a BBIE, the default Context Values are inherited from those of the ABIE that aggregates this BBIE.

The Context Values provide useful information for business analysts to understand in what business situation they should apply a BIE. They also serve as the meta-data for classifying and organizing BIEs in the Central Registry so that business analysts can locate and adopt suitable BIEs, and correctly apply the BIEs in a specific project. Thus, Context Values are the mandatory information of the registered BIEs of Common Schemas in the Central Registry.

For Project Schema design, it is not compulsory to assign Context Values to project-defined BIEs to ease the design process since the business context information is not required for programming XML Schema.

A Context Value shall be a controlled vocabulary. That is, a Context Value must be selected from a pre-defined code list registered in the Central Registry. See Section 4.3.11.

Table 4-6: Context Categories.

Context Category	*Definition*	*Default Value for ABIE*
Business Process Classification	The Business Process to which the information described by this BIE is specific.	In all contexts
Service / Product Classification	The classification of products or services to which the information described by this BIE is specific.	In all contexts
Industry Classification	The industry vertical to which the information described by this BIE is specific.	In all contexts
Geopolitical	The geographical location to which the information described by this BIE is specific.	In all contexts
Official Constraints	The legal and contractual constraint to which the information described by this BIE is specific.	None

4.3.11. Controlled Vocabulary

A **controlled vocabulary** refers to a pre-defined set of the values for a data element and the value set is controlled by an authority. A **code list** is a typical controlled vocabulary that defines a set of permissible values for a code-style data element. A code list is a list of [**code value**, **code name**] value pairs. A code value is a permissible value for a data element and a code name is the description that this code value stands for. For example, the BBIE of "Hong Kong District", which is based on a Code CCT, may use a code list to define its permissible values, e.g. ["WC", "Wanchai"], ["CT", "Central"], etc.

Other controlled vocabularies include Context Values and code-style Supplementary Components (e.g. quantity unit code).

When a business analyst needs to establish a code list for a BBIE or a Supplementary Component in a Project Schema, he/she should first consider to reference any suitable code list that is already defined by a relevant industry standard (e.g. UN/ECE Rec. 20 for quantity unit code) or is already registered in the Central Registry. If no such code list is available, the business analyst should define a project-specific code list for that BBIE. When a business analyst submits a BIE for concerted alignment, the business analyst should attach all associated code lists with the request.

All code lists for Common Schemas must be centrally defined and published in the Central Registry except the code lists that are industry standards. The controlled vocabulary for the Context Value in each Context Category as listed in Table 4-6 shall also be published and maintained in the Central Registry.

4.4. Data Dictionary

A data dictionary is a database for organizing the information models that define all relevant data elements for use within a specific scope. (The XSDs developed for the information models can also be referenced in the dictionary.) A data dictionary is either part of the Project Registry for Project Schema design or part of the Central Registry for Common Schema design.

The sophistication of a dictionary implementation varies from a modelling spreadsheet to a sophisticated database system that allows the information models to be searched and browsed. This sub-section is not intended to cover the physical implementation of a data dictionary but only to discuss its logical functions.

4.4.1. Dictionary Entry Information

Each information model is stored as a dictionary entry. In a dictionary, all entries must be uniquely defined and identified. An entry may only reference other entries within the same dictionary. Refer to Part III: XML Schema Management Guide, Section 5 for the information maintained for each dictionary entry.

Examples of the dictionary entry information for an information model include:

- **Unique Identifier (UID)**: an identifier that uniquely identifies a dictionary entry.
- **Dictionary Entry Name**: the official name of the dictionary entry. The naming rules provided in Section 4.4.2 apply.
- **Version**: the version identifier of the model. Evolution of a model may develop different model versions, which are stored in separate Entries and identified by different UIDs. The different model versions may share the same Dictionary Entry Name and definition. (See Section 5.4.3 for the numbering convention for specifying versions.)
- **Definition**: the semantic business meaning of the Entry.
- **Business Terms**: the synonym terms under which the model is commonly known as and used in the business. A model may have several business terms.
- **Usage Rules**: the constraints that describe specific conditions applicable to the model. Usage rules can also describe any validation rules that cannot be supported by the XML Schema language but should be handled by system implementations, e.g. the existence of the HKID number must be validated.

4.4.2. Dictionary Content Rules

The following rules are provided for preparing and organizing the dictionary content in an unambiguous and consistent manner.

4.4.2.1. General Rules

1. The dictionary content, with the exception of Business Terms, shall be in the English language following the primary Oxford English Dictionary English spellings to assure unambiguous spelling.
2. The Dictionary Entry Unique Identifier (UID) shall be unique in the dictionary.
3. The definition shall be consistent with the requirements of ISO 11179-4 and will provide an understandable meaning, which should also be translatable to other languages.
4. Whenever both the definite (i.e. “the”) and indefinite article (i.e. “a”) are possible in a definition, preference shall be given to an indefinite article (i.e. “a”).
5. The Dictionary Entry Name shall be concise and shall not contain consecutive redundant words.
6. The Dictionary Entry Name and all its components shall be in singular form unless the concept itself is plural.

7. The Dictionary Entry Name shall only contain verbs, nouns and adjectives (i.e. no words like “and”, “of”, “the”, etc.). This rule shall be applied to the English language, and may be applied to other languages as appropriate.
8. Abbreviations and acronyms that are part of the Dictionary Entry Name shall be expanded or explained in the definition.

4.4.2.2. Dictionary Entry Name for Core Component Type

[Note : Since all CCTs and their RTs are pre-defined and the corresponding XSDs have been generated and posted in the Central Registry for use by project teams, project teams should not create any CCT. The rule for naming CCT is described here for information only]

1. The Dictionary Entry Name of a CCT shall consist of two components. The first component shall be given by the Type name and the second component shall be the word “Type”. The two components shall be separated by a dot. The space character shall be used to separate multi-word Type name. To allow spell checking of the words in the Dictionary Entry Name, a space character shall follow the dot after the first component. For example, “Amount. Type”, “Measure. Type”.

4.4.2.3. Dictionary Entry Name for Basic or Association Business Information Entity

1. The Dictionary Entry Name of a BBIE or ASBIE shall consist of three components. The first, second, and third components shall be given by the Object Class Term, the Property Term, and the Representation Term respectively. The three components shall be separated by dots. The space character shall separate words in multi-word components. To allow spell checking of the words in the Dictionary Entry Name, a space character shall follow the dots after the first and second components. For example, “Postal Address. Street. Text”.
2. For the Dictionary Entry Name of a BBIE or ASBIE, if the second component (Property Term) uses the same or equivalent word or words as the third component (Representation Term), the redundant word(s) in the second component shall be removed from the Dictionary Entry Name. If no word remains in the second component, the whole second component shall be removed. For example, “Postal Address. Postal Code. Code” is modified to “Postal Address. Postal. Code”; “Country. Code. Code” is modified to “Country. Code”.

4.4.2.4. Dictionary Entry Name for Aggregate Business Information Entity

1. The Dictionary Entry Name of an ABIE shall consist of two components. The first component shall be given by the Object Class Term and the second component shall be the word “Details”. The two components shall be separated by a dot. The space character shall be used to separate the multi-word first component. To allow spell checking of the words in the Dictionary Entry Name, a space character shall follow the dot after the first component. For example, “Postal Address. Details”

4.4.2.5. Dictionary Entry Name for Business Document

1. The Dictionary Entry Name of a Business Document shall consist of two components. The first component shall be given by the Document name and the second component shall be the word “Document”. The two components shall be separated by a dot. The space character shall be used to separate the multi-word Document name. To allow spell

checking of the words in the Dictionary Entry Name, a space character shall follow the dot after the first component. For example, “Purchase Order. Document”.

4.5. Reuse of Common Schemas

When a Common Schema BIE is reused in a project, a Project BIE must be created based on that Common Schema BIE. The Project BIE should be placed in the Project Data Dictionary. This Project BIE shall be assigned a new UID according to the naming convention specified by the project[10]. The other dictionary entry information and modelling information of this Project BIE shall be copied from the reused Common Schema and amended according to the following rules.

When a Common Schema ABIE is reused, all other BIEs directly and indirectly aggregated by this ABIE shall also be reused

When a Common Schema BBIE is reused without its aggregating ABIE, the Project BBIE created from that reused BBIE may need to be assigned a new Object Class Term and/or a new Property Term because the Project BBIE is now aggregated in a different ABIE.

Table 4-7 shows an example of creating the Project BBIE for the “HKID” (Hong Kong Identity Number) for an applicant from the Common Schema BBIE “Identification”.

Table 4-7: Example of reusing a Common BBIE.

	Object Class Term	***Property Term***	***Dictionary Entry Name***
Common Schema BBIE	Hong Kong Identity Card	Identification	Hong Kong Identity Card. Identifier
Project BBIE	Applicant	HKID	Applicant. HKID. Identifier

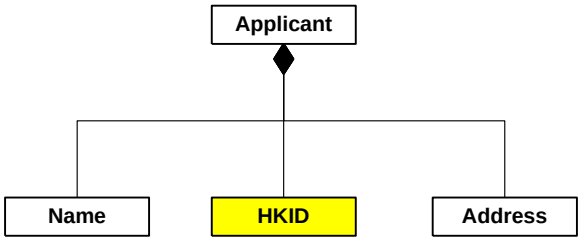


Figure 4-6: Example of Applicant ABIE (a Project BIE).

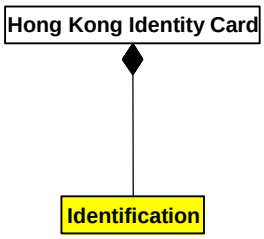


Figure 4-7: Example of Common Schema BBIE “Identification” and its aggregating Common Schema ABIE “Hong Kong Identity Card”.

[10] The UID of a Common Schema shall be assigned with a prefix of “COM” and the UID of a CCT shall be assigned with a prefix of “CCT”. Project teams should not use these reserved prefixes in assigning the UIDs to their Project Schemas.

A Common Schema ASBIE should not be reused without reusing its aggregating ABIE because a Common ASBIE only represents an association and contains no other meaningful information for reuse.

In reusing a Common Schema BIE in a Project or another Common Schema, a project team normally reuses the latest version of the Common Schema BIE. If this reused BIE is later updated to a new version, the project team need not update those Project or Common Schemas immediately to adopt the latest version of the updated BIE. The decision to update that Project or Common Schema to adopt a newer version of a BIE should be driven by business needs, e.g. when new business requirements come in.

When a project team reuses a Common Schema, it should register reuse and the reuse details (e.g. B/D, project, contact information) in the Central Registry.[11] This registration information allows the project team to be notified of subsequent changes to the Common Schema.

Project-specific CCTs should not be created. Approved CCTs in the Central Registry shall be used to construct Project BIEs.

[11] If a project team has customized the BIEs of a Common Schema in its Project Schema, it is formally not considered to be a reuse. Nevertheless, the project team is also recommended to report this partial reuse to the Interoperability Framework Coordination Group (IFCG) Standing Office. If the customization is commonly needed by different project teams to develop their Project Schemas, it may be necessary to update the Common Schema to reflect this common customization.

5 XML Schema Definition Development

5.1. Scope

This section provides the mechanism for programmers to develop the XML Schema Definitions (XSDs) based on the information models constructed by business analysts using the BIM methodology provided in Section 4. The mechanism is called a syntax-binding mechanism that binds the XML syntax to the information models discussed in Section 4. Targeted readers of this section are programmers who know XML Schema programming. Readers can follow the procedures and rules provided in this section to convert the information models into XSDs. This mechanism is intended to rely on minimal human decisions so that it is possible to develop software tools to automate the conversion.

Procedures and rules are provided for programmers to convert the information models of Business Document, Aggregate Business Information Entity (ABIE), Basic Business Information Entity (BBIE), and Core Component Type (CCT) into XSD code. No conversion procedure is needed for Association Business Information Entity (ASBIE) because the ABIE-to-XSD conversion has already covered the use of the ASBIE information.

The conversions of information models to XSD code are outlined as follows:

- A CCT has multiple Representation Terms (RTs). Each RT shall be converted into a schema complex type (i.e. `xs:complexType`[12]). The Content Component shall be coded as the schema simple content (i.e. `xs:simpleContent`) of that `xs:complexType`. Each Supplementary Component shall be coded as an XML attribute definition (i.e. `xs:attribute`) inside that `xs:simpleContent`.

- A BBIE shall be converted into an `xs:complexType`. The BBIE content shall be coded as the `xs:simpleContent` of that `xs:complexType`. The `xs:simpleContent` is a restriction (i.e. `xs:restriction`) based on `xs:complexType` of the used CCT. Each associated format restriction of the Content Component shall be coded as a schema facet (e.g. `xs:pattern`). Each Supplementary Component shall be coded as an XML attribute definition (i.e. `xs:attribute`) inside that `xs:simpleContent`. The possible values for a Supplementary Component shall be coded as schema enumerated values (i.e. `xs:enumeration`) of the `xs:attribute` that represents the Supplementary Component.

- An ABIE shall be converted into an `xs:complexType`. Each aggregated BBIE or ASBIE shall be coded as a child element definition (`xs:element`) or an `xs:any`, which allows any element to occur in the XML instance document.

- A Business Document is converted to an XML element definition (i.e. `xs:element`) based on the root ABIE.

[12] "`xs:`" is assumed to be the prefix that is associated with the XML Schema namespace through the declaration, `xmlns:xs="http://www.w3.org/2001/XMLSchema"`

This section also provides the procedure for packaging XSD code fragments into a schema document (or simply called schema in this section). Guidelines on namespace assignment, schema versioning, and meta-data documentation are also provided.

Programmers should follow the mechanism provided in this section to convert the information models constructed by business analysts into XSD schema documents. Software tools can also be produced to program the mechanism to automate the conversion. However, programmers are allowed to fine-tune the XSD code resulted from the manual or automatic execution of the conversion mechanism although this fine-tuning step is usually not necessary.

5.2. High-Level Procedure for Developing XSDs for Project Schemas

A programmer in a project team shall follow the procedure below to develop XSDs for Project Schemas:

1. Convert every BBIE into XSD code (see Section 5.3.2)
2. Convert every ABIE into XSD code (see Section 5.3.3)
3. Define a root element for every Business Document (see Section 5.3.4)
4. Create a schema document (see Section 5.4);

The conversion procedure for CCT is also provided (Section 5.3.1) for programmers to understand the conversion of CCTs into XSD code. However, programmers need not follow this procedure to do the conversion. A standard schema document of the approved CCTs (CCT schema) shall already be defined and published in the Central Registry for use by project teams. Programmers shall directly import the CCT schema to their Project Schemas.

5.3. Conversion Procedures

5.3.1. Core Component Type

A CCT has multiple RTs. Each RT shall be converted into a schema complex type `xs:complexType`. The Content Component shall be coded as the `xs:simpleContent` of that `xs:complexType`. Each Supplementary Component shall be coded as an `xs:attribute` inside that `xs:simpleContent`.

The procedure for converting an RT of a CCT into XSD code is as follows:

1. Define an `xs:complexType` for the RT. The `xs:complexType` name shall be translated from the RT name according to the naming rules in Section 5.5.1.
2. Declare an `xs:simpleContent` inside the `xs:complexType` to code the CCT content.
3. Extend the `xs:simpleContent` using `xs:extension` with an appropriate schema primitive data type based on the primitive data type of the Content Component. Table 5-1 should be referenced to choose the appropriate schema primitive data type.
4. Define an `xs:attribute` for each Supplementary Component. The `xs:attribute` name shall be translated from the Supplementary Component name according to Table 5-2. (See

also Section 5.5.3 for the naming rule for `xs:attribute`.) Table 5-2 should be referenced to assign an appropriate schema primitive data type to the `xs:attribute`. The occurrence of the `xs:attribute` is (specified by the `use` attribute) is either `required` or `optional`, which depends on whether the Supplementary Component is mandatory or optional (see Table 4-1).

Listing 5-1 shows the CCT schema file that contains "Quantity", "Count", and "Identifier" RTs.

Table 5-1: Recommended schema primitive data types for Content Component primitive data types.

Content Component primitive types	*XML Schema Primitive Data Type*
Binary	`xs:base64Binary`
Boolean	`xs:Boolean`
Date	`xs:date`
Date Time	`xs:datetime`
Time	`xs:time`
Decimal	`xs:decimal`
Integer	`xs:integer`
String	`xs:string`
URI	`xs:anyURI`

Table 5-2: Recommended `xs:attribute` representations for Supplementary Components.

Supplementary Component	*Attribute Name*	*Schema Primitve Datatype*
Agency ID	`agencyId`	`normalizedString`
Agency Name	`agencyName`	`normalizedString`
Character Set Code	`characterSetCode`	`token`
Code List ID	`codeListId`	`normalizedString`
Code List Name	`codeListName`	`normalizedString`
Code List Version	`codeListVersion`	`token`
Code Name	`codeName`	`normalizedString`
Currency Code	`currencyCode`	`token`
Encoding Code	`encodingCode`	`token`
Filename	`filename`	`normalizedString`
Format	`format`	`normalizedString`
Language Code	`languageCode`	`language`
MIME Code	`mimeCode`	`token`
Object URI	`objectUri`	`anyURI`
Protocol Code	`protocolCode`	`token`
Scheme ID	`schemeId`	`normalizedString`
Scheme Name	`schemeName`	`normalizedString`
Scheme Version	`schemeVersion`	`token`
Unit code	`unitCode`	`token`

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace=" http://www.xml.gov.hk/schemas/cct"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:cct="http://www.xml.gov.hk/schemas/cct"
version="1.0"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

   <xs:complexType name="Quantity.CT">
      <xs:simpleContent>
         <xs:extension base="xs:decimal">
            <xs:attribute name="agencyId" type="xs:normalizedString"
            use="optional"/>
            <xs:attribute name="agencyName" type="xs:normalizedString"
```

```
            use="optional"/>
            <xs:attribute name="codeListId" type="xs:normalizedString"
            use="optional"/>
            <xs:attribute name="codeListVersion" type="xs:token"
            use="optional"/>
           <xs:attribute name="unitCode" type="xs:token" use="optional"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>

   <xs:complexType name="Count.CT">
      <xs:simpleContent>
         <xs:extension base="xs:integer">
            <xs:attribute name="agencyId" type="xs:normalizedString"
            use="optional"/>
            <xs:attribute name="agencyName" type="xs:normalizedString"
            use="optional"/>
            <xs:attribute name="codeListId" type="xs:normalizedString"
            use="optional"/>
            <xs:attribute name="codeListVersion" type="xs:token"
            use="optional"/>
            <xs:attribute name="unitCode" type="xs:token" use="optional"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>

   <xs:complexType name="Identifier.CT">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute name="agencyID" type="xs:normalizedString"
            use="optional"/>
            <xs:attribute name="agencyName" type="xs:normalizedString"
            use="optional"/>
            <xs:attribute name="schemeID" type="xs:normalizedString"
            use="optional"/>
            <xs:attribute name="schemeName" type="xs:normalizedString"
            use="optional"/>
            <xs:attribute name="schemeVersion" type="xs:token"
            use="optional"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>

</xs:schema>
```

Listing 5-1: CCT schema that contains “Quantity”, “Count”, and “Identifier” RTs.

5.3.2. Basic Business Information Entity

A BBIE shall be converted into an `xs:complexType`. The BBIE content shall be coded as the `xs:simpleContent` of that `xs:complexType`. The `xs:simpleContent` is a `xs:restriction` based on `xs:complexType` of the used CCT. Each associated format restriction of the Content Component shall be coded as a schema facet (e.g. `xs:pattern`). Each Supplementary Component shall be coded as an XML attribute definition (i.e. `xs:attribute`) inside that `xs:simpleContent`. The possible values for a Supplementary Component shall be coded in `xs:enumeration` of the `xs:attribute` that represents the Supplementary Component.

The procedure for converting a BBIE into XSD code is as follows:

1. Define an `xs:complexType` for the BBIE. The `xs:complexType` name shall be translated from the Dictionary Entry Name of the BBIE according to the naming rules in Section 5.5.1.

2. Declare an `xs:simpleContent` inside the `xs:complexType` to code the BBIE content.

3. Restrict the `xs:simpleContent` using `xs:restriction` based on the `xs:complexType` for the RT referenced by the BBIE.

4. Define a facet for each format restriction on the Content Component according to Table 5-3. In most circumstances, the value of the format restriction can be directly used for the corresponding facet. For the expression format restriction, the business analyst may provide a textual description; in that case, the programmer should write an equivalent regular expression for the `xs:pattern` facet.

5. If there is any Supplementary Component that is given a default value and/or a list of possible values, declare an `xs:attribute` to specify the details. The steps are as follows:

    a. Declare the `xs:attribute` that represents the Supplementary Component and assign the `default` value if given. (See Table 5-2 and Section 5.5.3.)

    b. Declare an `xs:simpleType` for the `xs:attribute`.

    c. Declare an `xs:restriction` based on the same schema data type of the `xs:attribute` already defined in the RT `xs:complexType`.

    d. Declare a list of `xs:enumeration`s to specify the given possible values (including the default value).

Table 5-3: Use of schema facets for coding the format restrictions of a Content Component.

Content Component Format Restriction	***Facet***
Expression	`xs:pattern`
Length	`xs:length`
Minimum Length	`xs:minLength`
Maximum Length	`xs:maxLength`
Enumeration	`xs:enumeration`
Total Digits	`xs:totalDigits`
Fractional Digits	`xs:fractionDigits`
Minimum Inclusive	`xs:minInclusive`
Maximum Inclusive	`xs:maxInclusive`
Minimum Exclusive	`xs:minExclusive`
Maximum Exclusive	`xs:maxExclusive`

Listing 5-2 lists the sample XSD code for the BBIE used to represent "United Nations Dangerous Goods Number" ("UNDG Number") as an example. A UNDG Number is a unique serial number assigned within the United Nations to substances and articles on a list of most commonly carried dangerous goods. The BBIE is described as follows:

- The Dictionary Entry Name of the BBIE is "Dangerous Goods. UNDG. Identifier".
- The Content Component is a four-decimal-digit identifier. The equivalent regular expression to represent a four-decimal-digit string is "`\d{4}`".
- The "Agency ID" Supplementary Component is given only one possible value, which is also its default value: the URL of the United Nations (i.e. http://www.un.org).

```
<xs:complexType name="DangerousGoodsUndgIdentifier.CT">
   <xs:simpleContent>
      <xs:restriction base="cct:Identifier.CT">
         <xs:pattern value="\d{4}"/>
         <xs:attribute name="agencyId" default="http://www.un.org">
            <xs:simpleType>
               <xs:restriction base="xs:normalizedString">
                  <xs:enumeration value="http://www.un.org"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:restriction>
   </xs:simpleContent>
</xs:complexType>
```

Listing 5-2: Sample XSD code fragment for the BBIE for "United Nations Dangerous Goods Number".

5.3.3. Aggregate Business Information Entity

An ABIE shall be converted into an `xs:complexType`. Each aggregated BBIE shall be coded as a child `xs:element` of the `xs:complexType`. Each aggregated ASBIE shall be coded as a child `xs:element` if it is associated with a Representation ABIE (including those Externally-Defined ones) or `xs:any` if it is associated with an Undefined Entity as the representation of the property.

The procedure for converting an ABIE into XSD code is as follows:

1. Define an `xs:complexType` for the ABIE. The `xs:complexType` name shall be translated from the Dictionary Entry Name of the ABIE by applying the naming rules in Section 5.5.1.

2. Declare a sequence (i.e. `xs:sequence`) or a choice (i.e. `xs:choice`) of child `xs:element`s.

3. Define an `xs:element` or `xs:any` for every aggregated BBIE or ASBIE. Provide the values for `maxOccurs` and `minOccurs` to reflect the cardinality of the aggregated BBIE/ASBIE in the ABIE.

    a. If the aggregated BIE is a BBIE, declare an `xs:element` of which the `type` is the `xs:complexType` for the BBIE. Translate the `xs:element` name from the Property Term of the BBIE by applying the naming rules in Section 5.5.2.

    b. If the aggregated BIE is an ASBIE and the ASBIE is associated with a Representation ABIE defined in the same data dictionary, declare an `xs:element` of which the type is the `xs:complexType` for the Representation ABIE. Translate the `xs:element` name from the Property Term of the ASBIE by applying the naming rules in Section 5.5.2.

    c. If the aggregated BIE is an ASBIE and the ASBIE is associated with an Externally-Defined ABIE (e.g. defined by an industry standard schema), apply one of the following three conversion rules:

        i. Define an `xs:element` of which the type is assigned with an `xs:complexType` defined externally in an imported schema. Translate the `xs:element` name from the Property Term of the ASBIE by applying the naming rules in Section 5.5.2.

ii. Define an `xs:element` which is referenced to an element defined externally in an imported schema. Note that it is not necessary to assign an `xs:element` name.

iii. Declare an `xs:any` which allows any element to occur in the XML instance document. Refer to the W3C XML Schema specification to determine the values for the attributes `namespace` and `processContents`. Note that it is not necessary to assign an `xs:element` name.

The conversion from the "Physical Address" ABIE to XSD code is illustrated in Table 5-4 as an example. Listing 5-3 lists the converted XSD code[13].

Table 5-4: Sample conversion from the ABIE for "Foreign Physical Address" to XSD code.

Dictionary Entry Name	*BIE Type*	*Cardin-ality*	*Order*	*Complex Type Name*	*Element Name*	*Min-occurs*	*Max-occurs*
Foreign Physical Address. Details	ABIE	n/a	n/a	`ForeignPhysicalAddressDetails.CT`	n/a	n/a	n/a
Foreign Physical Address. Street. Text	BBIE	1	1	`ForeignPhysicalAddressStreetText.CT`	`Street`	1	1
Foreign Physical Address. City. Name	BBIE	1	2	`ForeignPhysicalAddressCityName.CT`	`City`	1	1
Foreign Physical Address. Country	ASBIE	1	3	`CountryDetails.CT` (the `xs:complexType` of the Representation ABIE with which this ASBIE is associated)	`Country`	1	1

```
<xs:complexType name="ForeignPhysicalAddressDetails.CT">
   <xs:sequence>
      <xs:element name="Street" type="ForeignPhysicalAddressStreetText.CT"/>
      <xs:element name="City" type="ForeignPhysicalAddressCityName.CT"/>
      <xs:element name="Country" type="CountryDetails.CT"/>
   </xs:sequence>
</xs:complexType>
```

Listing 5-3: Sample XSD code fragment for the ABIE for "Foreign Physical Addresss".

Table 5-5 and Listing 5-4 show the example of how XSD code shall be converted when an ABIE aggregates an ASBIE that is associated with an Externally Defined Entity..

Table 5-5: Sample conversion from the ABIE for "Document Header" to XSD code.

Dictionary Entry Name	*BIE Type*	*Cardin-ality*	*Order*	*Element Type or Element Reference or* xs:any	*Min-occurs*	*Max-occurs*
Document Header. Details	ABIE	n/a	n/a	`DocumentHeaderDetails.CT`	n/a	n/a
Document Header. First	ASBIE	1	1	`type="ds:SignatureType"`	1	1

[13] The XSD code for the `xs:complexTypes` of the child elements is not shown in the listing.

Signature. External						
Document Header. Second Signature. External	ASBIE	1	2	ref="ds:Signature"	1	1
Document Header. Third Signature. External	ASBIE	1	3	<xs:any namespace=" http://www.w3.org/2000/09 /xmldsig#" processContents="lax" />	1	1

```
<xs:schema targetNamespace="http://www.xml.gov.hk/schemas/example"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
version="1.0"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
…
   <xs:complexType name="DocumentHeaderDetails.CT">
      <xs:sequence>
         <xs:element name="FirstSignature" type="ds:SignatureType"/>
         <xs:element ref="ds:Signature"/>
         <xs:any namespace="http://www.w3.org/2000/09/xmldsig#"
processContents="lax"/>
      </xs:sequence>
   </xs:complexType>
…
</xs:schema>
```

Listing 5-4: Sample XSD code fragment for the ABIE for "Document Header"

5.3.4. Business Document

A Business Document is converted to an `xs:element` definition based on the root ABIE. The `xs:element` name shall be translated from the Document name by applying the naming rules in Section 5.5.2. The `xs:element` type shall be assigned with the `xs:complexType` of the root ABIE.

Listing 5-5 shows an example of declaring the root `xs:element` for a "Purchase Order" Document supposing the Dictionary Entry Name of the root ABIE is "Purchase Order. Details" (i.e. the `xs:complexType` name of the root ABIE is "`PurchaseOrderDetails.CT`").

```
<xs:element name="PurchaseOrder" type="PurchaseOrderDetails.CT"/>
```

Listing 5-5: Sample XSD code fragment for the ABIE for "Purchase Order".

5.4. Creating a Schema Document

The XSD code fragments for related BBIEs, ABIEs, and Business Documents in the Project Registry should be concatenated and packaged in a schema document (i.e. an XSD file) for use in software solutions.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema targetNamespace="http://www.xml.gov.hk/schemas/example"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:cct="http://www.xml.gov.hk/schemas/cct"
version="2.1"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xs:import namespace="http://www.xml.gov.hk/schemas/cct"
  schemaLocation="cct.xsd"/>

  <!-- The concatenation of the XSD code fragments for related Project BBIEs,
  ABIEs, and Documents -->

</xs:schema>
```

Listing 5-6: Declaring the `xs:schema` element in a valid XSD file.

The `xs:schema` element is the root element of a valid schema as shown in Listing 5-6. It should be declared to enclose all XSD code fragments converted from the information models in the Project Registry. (When some Common Schemas are adopted, their BIEs and XSD code should have already been imported to the Project Registry; therefore, the schema should have already included the XSD code from the adopted Common Schemas.) The sequence of concatenating the XSD code fragments is not significant but it is recommended to follow the sequence of the BIEs organized in the Project Registry, i.e. by UID.

On the one hand, it is common to organize the XSDs for multiple business documents in a single schema document when there are not too many XSDs. On the other hand, when the number of XSDs is large, it is also possible to organize the XSDs for a project in multiple schema documents by usage patterns. The `xs:include` schema element can be used to bring the XSDs defined in external schema documents into the main schema document. (The including and included schemas must share one effective target namespace. See Section 5.4.2.)

5.4.1. Importing External Schemas

External schema documents can be imported to a master schema document to allow external schema definitions to be referenced. External schema definitions from an imported schema document are referenced using a namespace different from the target namespace of master schema. It may be necessary to import external schema documents to a master schema document in the following cases:

- The schema for the approved CCTs (or CCT schema) should be imported to the schema of the Project Schemas as shown in Listing 5-6. This is necessary because the `xs:complexType` definitions for the Project BBIEs need to reference the `xs:complexType` definitions for CCTs in the CCT schema. The CCT schema will be maintained in the Central Registry and will be accessible at `http://www.xml.gov.hk/schemas/cct/cct.xsd`. The target namespace of the CCT schema is `http://www.xml.gov.hk/schemas/cct`.

- When a schema document contains XSD code converted from Externally-Defined Entities and these Entities are defined in some external schemas (e.g. the W3C XML Signature XSD at `http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd`), these external schema documents should be imported like the example shown in Listing 5-4.

- A large Project Schema may be specified by multiple external schema documents (with either different or same namespace) to ease maintenance of schema definitions, i.e. schema definitions for different business functions or domains are specified in different schema documents. If these schema documents are assigned different namespaces, a master schema document may import these external schema documents, each of which is assigned with a different namespace.[14]

5.4.2. Namespaces

5.4.2.1. Registration of Namespace URIs

A project team should request the IFCG Standing Office to assign a unique URI for defining the namespaces used in its Project Schema and all assigned namespace URIs will be registered in the Central Registry. The URI can be in the form of `http://www.xml.gov.hk/schemas/project_name` or the like. Using this registered URI, a project team may define different URIs for different namespaces in the form of `http://www.xml.gov.hk/schemas/project_name/domain_1`. An example namespace URI can be `http://www.xml.gov.hk/schemas/ICJS/prosecution`, where ICJS is the project name and prosecution identifies the business domain.

5.4.2.2. Declaring Namespaces in Schema Document

Proper namespace declarations should be included on the `xs:schema` element in a schema document.

The optional `targetNamespace` attribute of the `xs:schema` element can be used to assign a target namespace to a schema. The `targetNamespace` is the namespace of all schema components in this schema together with any schema documents included using the `xs:include`. (Unlike an imported schema documents, an included schema document must declare its target namespace same as the including (master) schema or declare no target namespace at all.) For example, the target namespace of the schema in Listing 5-6 is `targetNamespace="http://www.xml.gov.hk/schemas/example"`. The project team should request for a centrally assigned target namespace as specified in the previous sub-section.

The `xs:schema` element must also declare the namespaces of the W3C XML Schema (i.e. `xmlns:xs="http://www.w3.org/2001/XMLSchema"`), the CCT schema document (i.e. `xmlns:cct="http://www.xml.gov.hk/schemas/cct"`), and other imported external schema documents (e.g. `xmlns:pros="http://www.xml.gov.hk/schemas/ICJS/prosecution"`).

[14]. The primary objective of using multiple namespaces in a schema document is to avoid name clash in naming XML elements, attributes, and types. Since the project team has the control over the Project Schema design, the name clash can usually be avoided within a single namespace. For a regular-size project, it is recommended to have only the target namespace except the namespace of the imported CCT schema. (The namespace for the CCT schema will not appear in the XML instance documents since CCTs are only converted into types not elements.) For a large project, a few namespaces can be declared and used to segregate XML elements, attributes, and types representing different business functionalities or domains. If too many namespaces are used in a schema document, the instance documents conforming to that schema document could likely contain many different namespaces prefixed to the elements and attributes. This could make XML data processing difficult and complicate the software development.

5.4.3. Versioning

A schema should be assigned with a version number by using the `version` attribute of the `xs:schema` element as shown in Listing 5-6. This allows different versions of a schema to be referenced and prevents an incorrect version of a schema from being used.

A released schema is recommended to have its version number in the form `n.m` while a draft schema is recommended to have its version number in the form `n.ma`, where `n` and `m` is a decimal number and `a` is an alphabet. For instance, version `2.1` represents a released version while version `2.0b` represents a draft version.

`n` represents the major version number. It should be changed when the validation of the existing XML documents (compliant to the current schema) against the new schema fails. For example, if the new schema includes a new mandatory element, all existing XML documents become invalid with respect to the new schema.

`m` represents the minor version number. It should be changed when the validation of existing XML documents against the new schema passes while the validation of new XML documents against the existing schema may fail. For example, if the new schema includes a new optional element, all existing XML documents remain valid with respect to the new schema while some new documents may be invalid with respect to the old schema.

5.4.4. Documentation of Meta-Data

Meta-data, such as Dictionary Entry Name, UID, Context Values, can be documented in a schema. When documentation of meta-data is needed, the `xs:documentation` element within the `xs:annotation` element is much preferred to XML comments (e.g. "`<!-- this is an XML comment. -->`"). The meta-data documented within an `xs:documentation` can be processed by XML processors. For example, an XSL stylesheet can be used to present the meta-data in the schema. However, `xs:documentation` adds processing overhead when the schema is used to validate XML documents although it does not affect validation results. To strike a balance, the usage of `xs:documentation` should be limited to giving essential information, such as Dictionary Entry Name and UID.

5.4.5. Fine-tuning XSD Code

The mechanism for converting information models to XSD code is designed to be mechanical for potential software automation. However, programmers are allowed to fine-tune the XSD code resulted from the manual or automatic execution of the conversion mechanism although this fine-tuning step is usually not necessary.

5.4.6. Coding Reused Common Schemas

As discussed in Section 4.5, BIEs of reused Common Schemas are translated into Project Schema BIEs (where the original Object Class Terms and Property Terms may be renamed). Therefore, the XSD code converted from the Project Schema BIEs shall automatically include the XSD code for the reused Common Schemas. However, if the XSD code for the reused Common Schema has been fine-tuned (after conversion from the Common Schema BIEs), the XSD code converted from the Project Schema BIEs should be changed manually to incorporate the fine-tuning code.

When the Object Class Term or Property Term needs to be renamed during the translation of a Common Schema BIE to a Project Schema BIE, the converted Project Schema XSD shall be different from the Common Schema XSD in the type name or element name translated from that Term. Although two Project Schemas that have reused the same Common Schema may have different XML Schema type names or XML element names in their XSDs, this is not significant data interoperability problem. It is because the XML data structures and representations for the Common Schema in the two Project Schemas are always the same. In other words, the element for the Common Schema in one Project Schema can be easily mapped to the corresponding element in another Project Schema despite their different element names. This is already a very high level of data interoperability.

5.4.7. Different approaches to reuse schemas

There are different ways to reuse schemas, the most common ways are copy code and include / import.

Copy code is the ”cleanest” way of reusing common schemas or other project schemas for the following reason : A common schema is adopted only to save the repetitive data alignment effort during the design of the project schema before using it to exchange data. Once the business partners agree upon the project schema, that project schema should not be changed and the business partners will build their applications according to that schema. If the project schema references or imports an externally controlled schema which may be updated from time to time, the project schema may also change without anticipation. The unanticipated change of project schema may break down the existing application.

Having said that, if the externally controlled schemas have good version control (e.g. different schema versions are maintained in different files so that specific versions can always be located and imported, rather than having the new version overwrite the old version in the same file), then there is no harm importing those schemas.

To summarize, (1) when we can consistently locate specific versions of an externally defined schema, then we can either use the copy code approach or the import approach. In the case of Common Schemas where not only XSDs but BIEs are available, we can even copy specific versions of the BIEs into our project information model to get a more complete model for easy reference. (2) When the specific version of the external schema we refer to may not be available in future, e.g. replaced by its new version, we should use the copy code approach.

5.5. XML Naming Rules

5.5.1. Name a Complex Type

Ignoring the impact caused by different versions of a BIE, the name of the `xs:complexType` that represents a BBIE or ABIE shall be translated from the Dictionary Entry Name following the procedure below:

1. Remove all dot and space characters from the Dictionary Entry Name.

2. Apply the upper camel case convention: all words shall be concatenated with the first letter in every word in upper case and the other letters in lower case.

3. Append "`.CT`" at the end to indicate it is an `xs:complexType`.

For example, the `xs:complexType` name for the Dictionary Entry Name "Postal Address. Street. Text" is "`PostalAddressStreetText.CT`".

When there are multiple dictionary entries for different versions of a BBIE or ABIE, it is possible that two or more different versions are used **within a single schema document**. Since two versions of the same BBIE or ABIE share the same dictionary entry name, they will be converted to two `xs:complexType`s of the same name, which is not allowed. To cater for different versions of a BIE, `xs:complexType` names should be made unique by incorporating version number information as follows:

1. Remove all dot and space characters from the Dictionary Entry Name.

2. Apply the upper camel case convention: all words shall be concatenated with the first letter in every word in upper case and the other letters in lower case.

3. Append the version number of the BBIE or ABIE in form of "V*n.ma*", where "*n.ma*" is the version number.

4. Append "`.CT`" at the end to indicate it is an `xs:complexType`.

For example, the `xs:complexType` name for the Dictionary Entry Name "Postal Address. Street. Text" and version "1.2a" is "`PostalAddressStreetText.V1.2a.CT`".

[Note : for simplicity, all `xs:complexType`s illustrated in this Guide will ignore the version information. Individual project teams should decide whether they incorporate version information in project-defined `xs:complexType`s consistently or only when different versions exist. For the Common Schemas, version number information will be incorporated consistently when generating XSDs.]

The name of the `xs:complexType` that represents an RT of a CCT shall be translated from the RT name following the procedure below:

1. Apply the upper camel case convention: all words shall be concatenated with the first letter in every word in upper case and the other letters in lower case.

2. Append "`.CT`" at the end to indicate it is an `xs:complexType`.

For example, the `xs:complexType` name for the RT name "Binary Object" is "`BinaryObject.CT`".

5.5.2. Name an Element

The name of the `xs:element` that represents the aggregation of a BBIE or ASBIE in an ABIE shall be translated from the Property Term of the BBIE/ASBIE by applying the upper camel case convention. For example, the `xs:element` name for the "Delivery Date" Property Term is "`DeliveryDate`".

The name of the root `xs:element` of a Business Document shall be translated from the Object Class Term of the root ABIE by applying the upper camel case convention. For example, the `xs:element` name for the "Purchase Order" Object Class Term is "`PurchaseOrder`".

5.5.3. Name an Attribute

The name of the `xs:attribute` that represents a Supplementary Component shall be translated from the Supplementary Component name by applying the lower camel case convention: all words shall be concatenated with the first letter in every word in upper case, except the first word, and the other letters in lower case. For example, the `xs:attribute` name for the "Currency Code" Supplementary Component is "`currencyCode`".

6 Use of Modelling Worksheets

6.1. Scope

This section provides seven process and information modelling worksheets to facilitate business analysts to understand and apply BPM and BIM methodologies provided in Sections 3 and 4. Each worksheet serves a concrete presentation as well as a complete view on the business requirement details to be captured for each of the process and information models. Theoretically, business analysts can construct a particular model by filling in the corresponding worksheet with the necessary details. The worksheets created by business analysts are to be used by programmers to implement the software solution.

During business information modelling, the business analysts may need to capture massive modelling information for a large number of data elements. It would be very tedious for them to fill in a big pile of worksheets. Therefore, the **data modelling spreadsheets** are recommended to substitute the modelling worksheets to capture modelling information for BIM[15] because the modelling spreadsheets allow easier data entry and can automate the conversion of modelling information into XSD code. The modelling spreadsheets are published in the Central Registry for download.

Each BIM worksheet has two parts: Part I – Business Information Modelling, and Part II – XML Schema Definition. Part II serves to guide a programmer to convert the information model specified in Part I to XSD code using the mechanism provided in Section 5.

6.2. Modelling Worksheets

The following two BPM worksheets and five BIM worksheets are provided in this section:

1. Business Collaboration worksheet
2. Business Transaction worksheet
3. Business Document worksheet
4. Basic Business Information Entity (BBIE) worksheet
5. Aggregate Business Information Entity (ABIE) worksheet
6. Association Business Information Entity (ASBIE) worksheet
7. Core Component Type (CCT) worksheet

[15] There is no BPM spreadsheet. Business analysts should use the first three worksheets included in this section to capture BPM information.

The Section A of each worksheet captures general worksheet information, including worksheet identifier, project identifier, technical and administrative contacts.

The other sections differ in different worksheets and are described in the following sections.

6.2.1. Business Collaboration Worksheet

The Business Collaboration worksheet shown in Table 6-1 is provided for a business analyst to specify a Business Collaboration (see Section 3.3.2).

Section B specifies the general information of this Collaboration.

Section C specifies the roles in this Collaboration. A Business Collaboration has at least two or more roles.

Section D specifies the Business Transactions in this Collaboration. A Business Collaboration has at least one Business Transaction.

Section E lists all Business Documents to be exchanged in this Collaboration. This is a collection of all Positive and Negative Response Documents in the Transactions identified in Section D.

Section F specifies the UML activity diagram to illustrate the choreography of the Business Transactions (document exchanges) in this Collaboration.

Table 6-1: Business Collaboration worksheet.

BUSINESS COLLABORATION WORKSHEET

A. Worksheet Information	
Worksheet ID: BCWS-ORDERENTRY	**Project ID:** XMLGL
Technical Contact: Josia Chan / CECID	**Administrative Contact:** Thomas Lee / CECID

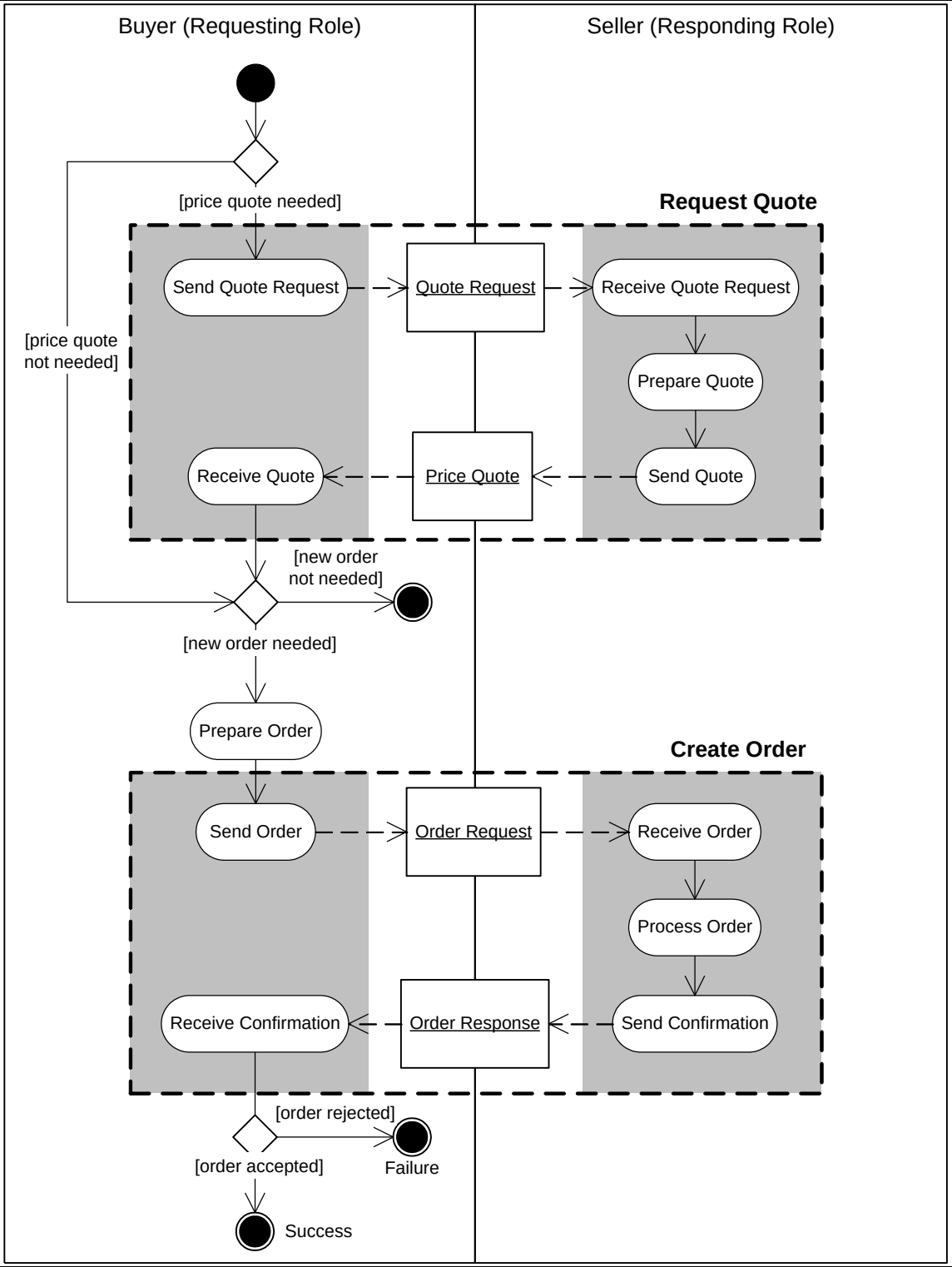
B. Business Collaboration Properties
Name: Order Entry
Description: A buyer and a seller exchange price quotes and create an order.
Scope: The buyer requests the seller to quote the offered prices of the goods for ordering. If the buyer agrees on the prices, it can place an order to the seller.
Pre-conditions: 1. The buyer and the seller have agreed to transact with each other.

C. Roles	
Name	***Description***
Buyer	The firm that places an order.
Seller	The firm that fulfills the order.

D. Business Transactions	
Name	***Description***
Request Quote	The buyer requests a price quote from the seller.
Create Order	The buyer places a purchase order to the seller.

E. Business Documents	
Name	***Description***
Quote Request	Request that the buyer sends to the seller for a price quote.
Price Quote	Price quote that the seller replies to the buyer on a quote request.
Purchase Order	Request by the buyer to place an order to the seller.
Order Confirmation	Indication whether the order request is accepted or rejected.

F. Activity Diagram



6.2.2. Business Transaction Worksheet

The Business Transaction worksheet shown in Table 6-2 is provided for a business analyst to specify a Business Transaction (see Section 3.3.3).

Section B specifies the general information of this Transaction.

Section C specifies the Request Document Flow and the Request Documents in the flow. The non-repudiation and data confidentiality requirements indicate whether Request Documents require digital signature and data encryption.

Section D specifies the Response Document Flow and the Positive and Negative Response Documents. The success conditions are the conditions that must be fulfilled for the Responding Role to initiate a Positive Document Flow; if any of these conditions cannot be fulfilled, the Responding Role will initiate a Negative Document Flow.

Table 6-2: Business Transaction worksheet.

BUSINESS TRANSACTION WORKSHEET

A. Worksheet Information	
Worksheet ID: BTWS-CREATE-ORDER	**Project ID:** XMLGL
Technical Contact: Josia Chan / CECID	**Administrative Contact:** Thomas Lee / CECID

B. Business Transaction Properties	
Name: Create Order	**One/Two-Way:** Two-way
Description: A buyer places a purchase order to a seller.	
Scope: The buyer sends an order request to the seller. The seller either accepts or rejects the order request.	
Pre-conditions: 1. The buyer and the seller have agreed to transact with each other. 2. The buyer and the seller have agreed on the prices of the goods to order.	
Requesting Role: Buyer	**Responding Role:** Seller

C. Request Document Flow		
Description: The buyer sends an order request to the seller.		
Non-Repudiation Required: Yes	**Data Confidentiality Required:** Yes	
C1. Request Documents		
No.	***Document Name***	***Business Information Carried***
1	Purchase Order	A request to place an order
2	Price Quote	The price quote referenced by the order request

D. Response Document Flow		
Description: The seller sends an order confirmation to the buyer after processing the order request.		
Success Conditions: 1. The order information is valid. 2. The order can be fulfilled.		
Non-Repudiation Required: Yes	**Data Confidentiality Required:** Yes	
D1. Positive Response Documents		
No.	***Document Name***	***Business Information Carried***
1	Order Confirmation	An indication that the order request is accepted
2	Purchase Order	The original order request
D2. Negative Response Documents		
No.	***Document Name***	***Business Information Carried***
1	Order Confirmation	An indication that the order request is rejected
2	Purchase Order	The original order request

6.2.3. Business Document Worksheet

The Business Document worksheet is shown in Table 6-3. Part I is provided for a business analyst to specify a Document (see Section 4.3.7). Part II is provided for a programmer to convert the specification in Part I into XSD code (see Section 5.3.4).

Section B specifies the dictionary entry information of this Document.

Part I:

Section C specifies the Document name and the root ABIE that provides the representation for this Document. The Document name is the Object Class Term of the root ABIE.

Part II:

Section D specifies the XSD code programmed for this Document. The element name translated from the Document name and the complex type representing the root ABIE should be specified.

Table 6-3: Business Document worksheet.

BUSINESS DOCUMENT WORKSHEET

A. Worksheet Information	
Worksheet ID: BDWS-00402	**Project ID:** XMLGL
Technical Contact: Josia Chan / CECID	**Administrative Contact:** Thomas Lee / CECID

B. Dictionary Entry Information	
UID: POS001101	
Dictionary Entry Name: Postal Service Order. Document	**Version:** 1.0
Definition: Purchase order of a postal service.	
Business Terms:	
Usage Rules:	

PART I – BUSINESS INFORMATION MODELLING

C. Document Name
Document Name (Object Class Term of Root ABIE): Postal Service Order
UID / Dictionary Entry Name of Root ABIE: POS001102 / Postal Service Order. Details

PART II – XML SCHEMA DEFINITION

D. XML Schema Code
Element Name: PostalServiceOrder
Complex Type: PostalServiceOrderDetails.CT
<xs:element name="PostalServiceOrder" type="PostalServiceOrderDetails.CT"/>

6.2.4. Aggregate Business Information Entity Worksheet

The Aggregate Business Information Entity (ABIE) worksheet is shown in Table 6-4. Part I is provided for a business analyst to specify an ABIE (see Section 4.3.6). Part II is provided for a programmer to convert the specification in Part I into XSD code (see Section 5.3.3).

Section B specifies the dictionary entry information of the ABIE.

Part I:

Section C specifies whether a Common Schema is reused or what existing schemas or standards are referenced. If this ABIE reuses a Common Schema, the UID and Dictionary Entry Name of the Common Schema should be specified. Otherwise, any existing schemas (e.g. other Project Schemas) and industry standards (e.g. UBL) that have been referenced to construct this ABIE should be specified.

Section D specifies the Object Class Term of this ABIE.

Section E specifies the BBIEs and ASBIEs aggregated by this ABIE.

Section F specifies the business context in which this ABIE should be used.

Part II:

Section G specifies the complex type name translated from the dictionary entry name of the ABIE (see Section 5.5.1).

Section H specifies the child elements for the aggregated BIEs specified in Section E (see Section 5.5.2).

Section I provides the XSD code for this ABIE.

Table 6-4: Aggregate Business Information Entity worksheet.

AGGREGATE BUSINESS INFORMATION ENTITY WORKSHEET

A. Worksheet Information	
Worksheet ID: ABIEWS-00448	**Project ID:** XMLGL
Technical Contact: Josia Chan / CECID	**Administrative Contact:** Thomas Lee / CECID

B. Dictionary Entry Information	
UID: BIE001120	
Dictionary Entry Name: Mail Item. Details	**Version:** 1.0
Definition: Details about a mail item.	
Business Terms:	
Usage Rules:	

PART I – BUSINESS INFORMATION MODELLING

C. Reused Common Schema / Referenced Schemas and Standards
Reused Common Schema:
Referenced Schemas and Standards:

D. Object Class
Object Class Term: Mail Item

E. Aggregated BIEs

Sequence Order or "Choice" [16]	***UID***	***Dictionary Entry Name of the aggregated BIE***	***Dictionary Entry Name of the Representation ABIE or "External" (for ASBIE only)***	***Property Term***	***Cardinality***
1	POS001121	Mail Item. Weight. Measure		Weight	1
2	POS001123	Mail Item. Dimension	Dimension. Details	Dimension	1
3	POS001122	Mail Item. Count		Count	1

F. Business Context	
Context Category	***Values***
Business Process Classification	In all context
Service / Product Classification	Postal service
Industry Classification	In all context
Geopolitical	Hong Kong
Official Constraints	None

[16] For ABIE with a choice of aggregated ASBIE and BBIE, fill-in the word "Choice" in the leftmost column instead of 1, 2, 3… as shown in the above example.

PART II – XML SCHEMA DEFINITION

G. Naming
Complex Type Name: MailItemDetails.CT

H. Child Elements				
Sequence Order or "Choice" [17]	***Element Name or xs:any***	***Element Type or Element Reference or xs:any***	***minOccurs***	***maxOccurs***
1	Weight	MailItemWeightMeasure.CT	1	1
2	Dimension	DimensionDetails.CT	1	1
3	Count	MailItemCount.CT	1	1

I. XML Schema Code

```
<xs:complexType name="MailItemDetails.CT">
   <xs:sequence>
      <xs:element name="Weight" type="MailItemWeightMeasure.CT"
minOccurs="1" maxOccurs="1"/>
      <xs:element name="Dimension" type="DimensionDetails.CT"
minOccurs="1" maxOccurs="1"/>
      <xs:element name="Count" type="MailItemCount.CT" minOccurs="1"
maxOccurs="1"/>
   </xs:sequence>
</xs:complexType>
```

[17] For ABIE with a choice of aggregated ASBIE and BBIE, fill-in the word "Choice" in the leftmost column instead of 1, 2, 3… as shown in the above example.

6.2.5. Association Business Information Entity Worksheet

The Association Business Information Entity (ASBIE) worksheet is shown in Table 6-5. Part I is provided for a business analyst to specify an ASBIE (see Section 4.3.5).

Section B specifies the dictionary entry information of the ASBIE.

Part I

Section C specifies whether a Common Schema is reused. An ASBIE of a Common Schema shall only be reused together with its aggregating ABIE.

Section D specifies the Object Class Term of this ASBIE, which is the Object Class Term of the aggregating ABIE.

Section E specifies the Property Term of this ASBIE.

Section F specifies the representation of this ASBIE. The Representation Term and Representation ABIE should be specified.

Part II

Section G specifies either the `xs:element` or `xs:any` definition. In the `xs:element` definition, the element `type` or `reference` should be specified. In the `xs:any` definition, the values for the `namespace` and `processContents` attributes should be specified.

It is not necessary to specify the XSD code in the ASBIE worksheet. The XSD code should be specified in the worksheet for the aggregating ABIE.

Table 6-5: Association Business Information Entity worksheet.

ASSOCIATION BUSINESS INFORMATION ENTITY WORKSHEET

A. Worksheet Information	
Worksheet ID: ASBIEWS-00450	**Project ID:** XMLGL
Technical Contact: Josia Chan / CECID	**Administrative Contact:** Thomas Lee / CECID

B. Dictionary Entry Information	
UID: POS001123	
Dictionary Entry Name: Mail Item. Dimension	**Version:** 1.0
Definition: Dimension of a mail item.	
Business Terms:	
Usage Rules:	

PART I – BUSINESS INFORMATION MODELLING

C. Reused Common Schema
Reused Common Schema:

D. Object Class
Object Class Term: Mail Item

E. Property
Property Term: Dimension

F. Representation
Representation Term (Object Class Term of Representation ABIE): Dimension
UID / Dictionary Entry Name of the Representation ABIE: POS001123 / Dimension. Details

PART II – XML SCHEMA DEFINITION

G. Child Element (Complex Type Name or Element Reference or xs:any)		
Element Name: Dimension	**Type:** DimensionDetails.CT	
Element Reference:		
xs:any	**namespace:**	**processContents:**

Note: this worksheet need not specify the XML Schema. The XML Schema code should be specified in the aggregating ABIE worksheet.

6.2.6. Basic Business Information Entity Worksheet

The Basic Business Information Entity (BBIE) worksheet is shown in Table 6-6. Part I is provided for a business analyst to specify a BBIE (see Section 4.3.4). Part II is provided for a programmer to convert the specification in Part I into XSD code (see Section 5.3.2).

Section B specifies the dictionary entry information of the BBIE.

Part I:

Section C specifies whether a Common Schema is reused or what existing schemas or standards are referenced. If this BBIE reuses a Common Schema, the UID and Dictionary Entry Name of the Common Schema should be specified. Otherwise, any existing schemas (e.g. other Project Schemas) and industry standards (e.g. UBL) that have been referenced to construct this BBIE should be specified.

Section D specifies the Object Class Term of this BBIE, which is the Object Class Term of the aggregating ABIE.

Section E specifies the Property Term of this BBIE.

Section F specifies the representation of this BBIE. The CCT used by this BBIE, the Representation Term (RT) of the CCT (see Table 4-3) referenced by this BBIE, and the primitive data type for the Content Component associated with the RT (see Table 4-2) should be specified.

Section F1 specifies the format restrictions, if any, for the Content Component based on its primitive data type (see Table 4-4).

Section F2 specifies the default value and possible values, if any, for each Supplementary Component associated with the CCT (see Table 4-1).

Section G specifies the business context in which this BBIE should be used.

Part II:

Section H specifies the complex type name translated from the Dictionary Entry Name (see Section 5.5.1).

Section I specifies the facet values that represent format restrictions specified in Section F1 (see Table 5-3).

Section J specifies the default and enumerated values of each XML attribute definition in the complex type to code the default and possible values of Supplementary Components specified in F2 (see Section 5.5.3 and Table 5-2).

Section K provides the XSD code for this BBIE.

Table 6-6: Basic Business Information Entity worksheet.

BASIC BUSINESS INFORMATION ENTITY WORKSHEET

A. Worksheet Information	
Worksheet ID: BBIEWS-00458	**Project ID:** XMLGL
Technical Contact: Josia Chan / CECID	**Administrative Contact:** Thomas Lee / CECID

B. Dictionary Entry Information	
UID: POS001122	
Dictionary Entry Name: Mail Item. Count	**Version:** 1.0
Definition: Number of mail items.	
Business Terms:	
Usage Rules:	

PART I – BUSINESS INFORMATION MODELLING

C. Reused Common Schema / Referenced Schemas and Standards
Reused Common Schema:
Referenced Schemas and Standards:

D. Object Class
Object Class Term: Mail Item

E. Property
Property Term: Count

F. Representation	
Core Component Type: Count	**UID:** CCT000020
Representation Term: Count	**Primitive Data Type:** Integer

F1. Format Restrictions	
Restriction	***Value***
Expression	
Length	
Minimum Length	
Maximum Length	
Enumeration	
Total Digits	10
Fractional Digits	
Minimum Inclusive	
Maximum Inclusive	
Minimum Exclusive	
Maximum Exclusive	

F2. Supplementary Components		
Supplementary Component	***Default Value***	***Other Possible Values***
Agency ID	http://www.unece.org	
Agency Name	UNECE	
Code List ID	Rec. 20: 3A	
Unit Code	PCE	

G. Business Context	
Context Category	***Values***
Business Process	In all context
Service / Product Classification	Postal service
Industry Classification	In all context
Geopolitical	In all context
Official Constraints	None

PART II – XML SCHEMA DEFINITION

H. Complex Type
Complex Type Name: MailItemCount.CT

I. Facet of Simple Content	
Facet	***Value***
pattern	
length	
minLength	
maxLength	
enumeration	
totalDigits	10
fractionDigits	
minInclusive	
maxInclusive	
minExclusive	
maxExclusive	

J. Enumerated Attribute Values		
Attribute	***Default Value***	***Enumerated Values (Including Default Value)***
agencyId	http://www.unece.org	http://www.unece.org
agencyName	UNECE	UNECE
codeListId	Rec. 20: 3A	Rec. 20: 3A
unitCode	PCE	PCE

K. XML Schema Code

```
<xs:complexType name="MailItemCount.CT">
  <xs:simpleContent>
    <xs:restriction base="cct:Count.CT">
      <xs:totalDigits value="10"/>
      <xs:attribute name="agencyId" default="http://www.unece.org">
        <xs:simpleType>
          <xs:restriction base="xs:normalizedString">
            <xs:enumeration value="http://www.unece.org"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="agencyName" default="UNECE">
        <xs:simpleType>
          <xs:restriction base="xs:normalizedString">
            <xs:enumeration value="UNECE"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="codeListId" default="Rec. 20: 3A">
        <xs:simpleType>
          <xs:restriction base="xs:normalizedString">
            <xs:enumeration value="Rec. 20: 3A"/>
          </xs:restriction>
```

```
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="unitCode" default="PCE">
          <xs:simpleType>
            <xs:restriction base="xs:token">
              <xs:enumeration value="PCE"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:restriction>
    </xs:simpleContent>
</xs:complexType>
```

1

6.2.7. Core Component Type Worksheet

The Core Component Type (CCT) worksheet is shown in Table 6-7. Part I is provided for a business analyst to specify a CCT (see Section 4.3.3). Part II is provided for a programmer to convert the specification in Part I into XSD code (see Section 5.3.1).

Section B specifies the dictionary entry information of the CCT.

Part I:

Section C specifies the Type name, and the Representation Terms (RTs). An RT shall be based on a primitive data type (see Table 4-2).

Section D defines each Supplementary Component in this CCT, and whether that Supplementary Component is mandatory or optional.

Part II:

Section E specifies the attribute definition for each Supplementary Component specified in Section D. (See Section 5.5.3 and Table 5-2.) Each XML attribute should be defined based on a schema primitive data type (see Table 5-1). If the Supplementary Component is mandatory, the attribute should be defined to be required; otherwise the attribute should be defined to be optional.

Section F provides the XSD code for this BBIE.

Table 6-7: Core Component Type worksheet.

CORE COMPONENT TYPE WORKSHEET

A. Worksheet Information	
Worksheet ID: CCTWS-QUANTITY	**Project ID:** XMLGL
Technical Contact Josia Chan / CECID	**Administrative Contact:** Thomas Lee / CECID

B. Dictionary Entry Information	
UID: CCT000009	
Dictionary Entry Name: Quantity. Type	**Version:** 1.0
Definition: A number of non-monetary units possibly including fractions.	
Business Terms: N/A	
Usage Rules: Use the Quantity RT if the quantity can be fractional (e.g. quantity counted in dozen pieces) Use the Count RT if the quantity is always an integer (e.g. quantity counted in pieces)	

PART I – BUSINESS INFORMATION MODELLING

C. Representation		
Type Name: Quantity		
Representation Term	***Primitive Data Type of Content Component***	***Definition***
Quantity	Decimal	A quantity possibly including fractions.
Count	Integer	An integral count.

D. Supplementary Components		
Supplementary Component Name	***Definition***	***Mandatory/ Optional***
Agency ID	The identification of the agency which maintains the quantity unit code list.	Optional
Agency Name	The name of the agency which maintains the quantity unit code list	Optional
Code List ID	The identification of the quantity code list, e.g. the URL of a source that publishes the code list.	Optional
Code List Version	The version of the quantity code list.	Optional
Unit Code	The quantity unit code.	Optional

1 **PART II – XML SCHEMA DEFINITION**
2

E. Attributes		
Attribute Name	***Schema Primitive Datatype***	***Use (required/optional)***
agencyId	normalizedString	optional
agencyName	normalizedString	optional
codeListId	normalizedString	optional
codeListVersion	token	optional
unitCode	token	optional

3

F. XML Schema Code

Representation Term: Quantity

Complex Type Name: Quantity.CT | **Schema Primitive Datatype:** decimal

```
<xs:complexType name="Quantity.CT">
   <xs:simpleContent>
      <xs:extension base="xs:decimal">
         <xs:attribute name="agencyId" type="xs:normalizedString"
   use="optional"/>
         <xs:attribute name="agencyName" type="xs:normalizedString"
   use="optional"/>
         <xs:attribute name="codeListId" type="xs:normalizedString"
   use="optional"/>
         <xs:attribute name="codeListVersion" type="xs:token"
   use="optional"/>
         <xs:attribute name="unitCode" type="xs:token" use="optional"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

Representation Term: Count

Complex Type Name: Count.CT | **Schema Primitive Datatype:** integer

```
<xs:complexType name="Count.CT">
   <xs:simpleContent>
      <xs:extension base="xs:integer">
         <xs:attribute name="agencyId" type="xs:normalizedString"
   use="optional"/>
         <xs:attribute name="agencyName" type="xs:normalizedString"
   use="optional"/>
         <xs:attribute name="codeListId" type="xs:normalizedString"
   use="optional"/>
         <xs:attribute name="codeListVersion" type="xs:token"
   use="optional"/>
         <xs:attribute name="unitCode" type="xs:token" use="optional"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

4