CHAPTER 6
XML-BASED METADATA DICTIONARY

In order to ensure that the ontology-based metadata dictionary can be applied to any real world implementation, the ontology-based metadata dictionary components will be transformed into XML-based metadata dictionary representation. Since XML descriptive strengths in well-formedness, validity, and schema denotation, whereby choosing it as a language for expressing the metadata dictionary contents, which in turn support maximal interoperability across the heterogeneous systems in a web-based environment. Besides, XML also provides the flexibility and scalability in building and manipulating the metadata dictionary contents. As such, metadata dictionary content management can be achieved by means of flexible XML data model. The fact that XML-based metadata dictionary consisting of XML-DTD incorporated with XML documents makes it imperative for an XML document needs to be validated by rules defined through XML-DTD (document type definition) representing XML data schematic description. The XML-DTD structure can be modeled from domain ontology components represented through the object-oriented and set theory discussed in chapter 5. In this chapter, the XML-DTD will be structured from the domain ontology components with a list of legal elements and attributes to maintain the conceptual and physical level of abstraction design. Next, the construction rules are set up to administer the correctness and consistency of the conceptual level formulation. Finally, the resulting XML-DTD is illustrated for the metadata dictionary validation.

6.1 Structural Design of XML-DTD from Domain Ontology Components

The structural design of XML-DTD is set up to maintain their conceptual and physical correspondence and consistency through two levels of abstraction, namely, the conceptual level of design abstraction and the physical level of design abstraction as follows.

6.1.1 The Conceptual Level of Design Abstraction

In order to capture the semantic element and conceptual formulation established in the domain ontology components, the XML-DTD at this level must encompass all virtual

concepts and their corresponding virtual properties and their relationships. The resulting structure of the XML-DTD is depicted in Figure 6.1, where each rectangle denotes an XML element or sub-element and each double-lined rounded rectangle represents an XML attribute within an element. White areas of nodes contain the element or attribute names and shaded areas of nodes hold the data elements or attribute values. A bracket symbol of an element indicates that there are sub-elements within that element. A (+) symbol in front of an element indicates that there are one or more instances of that element, whereas a (?) symbol indicates zero or one instance of the element. Any element without a symbol represents exactly one instance of that element. The data elements and their attributes can be atomic values or string; each unique identifier is denoted by *id* and the corresponding identifier reference is denoted by *idref*. The identifier reference refers to an *id* of another element shown as a dashed arrow.

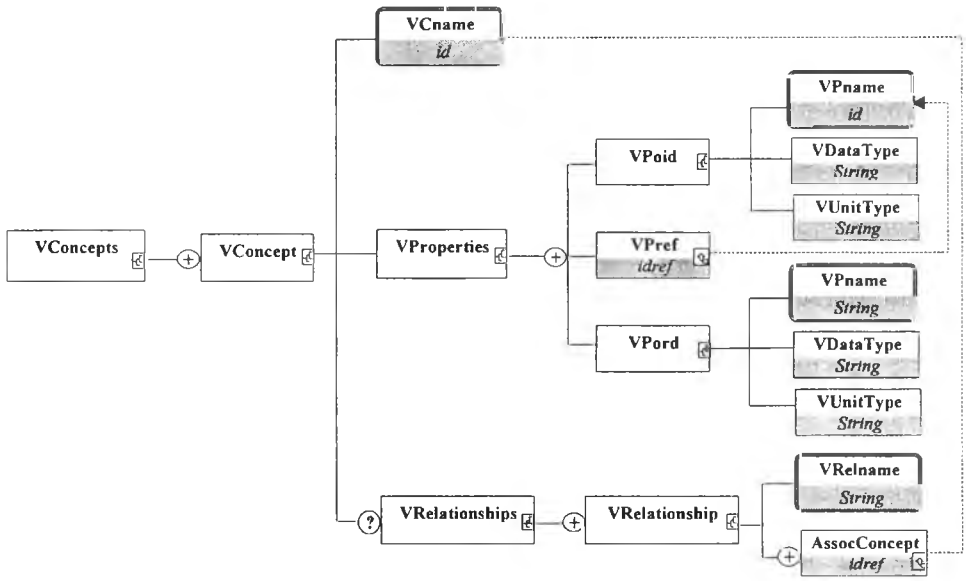

Figure 6.1 The XML-DTD structure at the conceptual level of design abstraction.

The XML-DTD structure at this level can be described as follows:

(1) Virtual concepts

- The root element `VConcepts` consists of one or more sub-elements `VConcept` of the domain ontology, whose names, $n(vc_i)$, $\forall i = 1..n$ are stored in the attribute

VCname of VConcept. Each VConcept in turn consists of a sub-element VProperties, and zero or one sub-element VRelationships.

(2) Virtual properties

The element VProperties of each VConcept contains one or more sub-elements VPoid, VPref, and VPord.

- Each VPoid denotes an object identifier property or key. The name of VPoid, denoted by $n(vp_{kc})$ of each $vp_{kc} \in OID(vc_k)$, is designated to the attribute VPname of VPoid.
- Each VPref denotes an object reference property or foreign key designated to store a virtual property name $n(vp_{kt})$ of each $vp_{kt} \in REF(vc_k)$ to its data element. Each data element of VPref is defined as *idref* to reference the virtual property name defined as *id* in VPoid.
- Each VPord denotes an ordinary property whose value is atomic value (e.g., integer, string). The name of VPord, denoted by $n(vp_{kd})$ of each $vp_{kd} \in ORD(vc_k)$, is designated to the attribute VPname of VPord.
- Each VPoid and VPord consists of sub-elements, VDataType and VUnitType, whose data elements are designated to store the virtual data type, $v(d_1)$, and unit type, $v(d_2)$, respectively. Note that a NULL value in a VUnitType element designates the property that is not of unit of measure.

(3) Relationships

- Each VConcept can associate with zero or more concepts, whose names, $n(vc_j)$ $(\forall j \subset \forall i=1..n) \wedge (j \neq i)$, are designated to the data elements of AssocConcept. The associated concept name of AssocConcept is defined as *idref*, pointing back to the already defined concept name in VConcept.
- The types of relationship, that is, associative, IS-A, and IS-PART-OF relationships, between VConcept and AssocConcept are designated to the attribute VRelname of VRelationship.

6.1.2 The physical level of design abstraction

The principal constituents of this level consist of the physical property names and other related physical information that are connected with the virtual properties and concepts in the conceptual level. This physical level is designed to incorporate the physical source configurations describing the configurations of the physical concepts and sources. The overall XML-DTD structure is illustrated in Figure 6.2, where bold pictures denote the physical level of abstraction.

(1) Physical properties and other physical information

- Each of `VPoid` and `VPord` at the conceptual level contains one or more synonymous physical properties whose names, $n(p_{kcl})$ of each $vp_{kc} \in OID(vc_k)$ and $n(p_{kdt})$ of each $vp_{kd} \in ORD(vc_k)$, are designated to the attribute `PPname` of `PProperty` of `VPoid` and `VPord`, respectively.
- Other physical information related to the physical property names of `VPoid` and `VPord`, such as the physical data type $v(I_1)$ and unit type $v(I_2)$ are designated to the data elements `PDataType` and `PUnitType`, respectively. Similarly, the concept name $v(I_3)$, source name $v(I_4)$, and corresponding virtual concept name $v(I_5)$ are designated to the attribute `PCname`, `PSname`, and `CVCname`, respectively. The attribute `PCname` and `PSname` are defined as *idref* to reference the physical concept and source name defined as *id* in the physical source configuration. The attribute `CVCname` is defined as *idref* to reference the corresponding virtual concept name defined as *id* of `VCname`.

(2) The physical source configurations

- The element `PhysicalSourceConfs` consists of one or more sub-elements `PSource` whose names, $n(S_i)$, $i = 1...n$, are designated to the attribute `PSname` of `PSource`.
- Each `PSource` consists of one or more sub-elements `PConcept` whose names, $n(pc_{kc})$ of each $pc_{kc} \in P(S_k)$, are designated to the attribute `PCname` of `PConcept`. The values of other physical configurations, *cnf* of pc_{kc} that associate to each

physical concept (e.g. physical data model, permission, owner) are designated to the data elements of `PDataModel`, `Permission`, and `Owner`, respectively.

The XML-DTD structure at this level is described below.

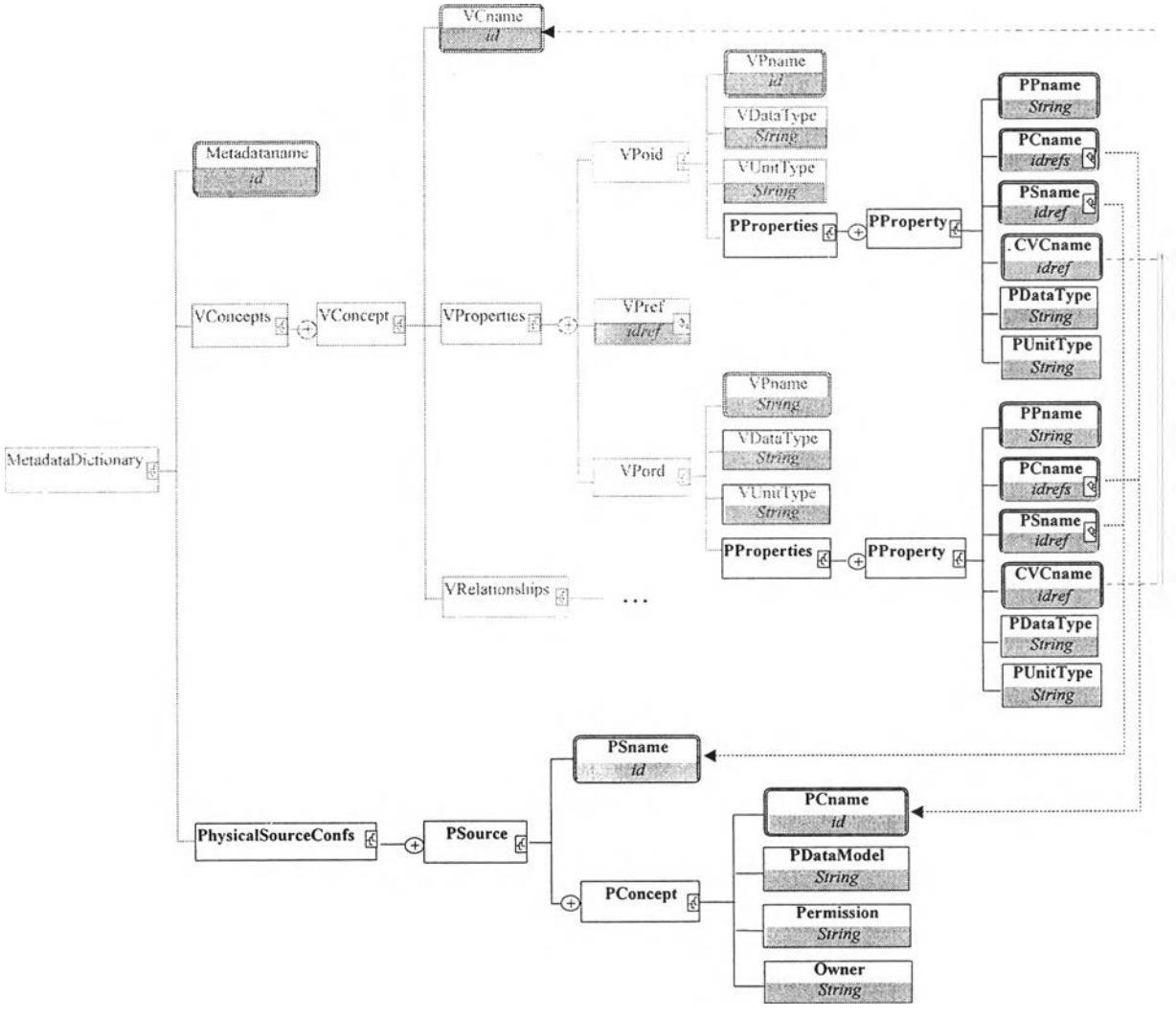

Figure 6.2 The XML-DTD structure at the physical level of design abstraction.

6.2 XML-DTD Metadata Dictionary Structure

The structure of XML-DTD metadata dictionary, depicted in Figure 6.3, is constructed based on the design abstractions described in the earlier section and ordered according to Figure 6.1 and 6.2.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE  MetadataDictionary [
  <!ELEMENT  MetadataDictionary   (VConcepts, PhysicalSourceConfs)>
  <!ATTLIST  MetadataDictionary   MetadataName        ID      #REQUIRED>
  <!ELEMENT  VConcepts            (VConcept)+>
  <!ELEMENT  VConcept             (VRelationships?, VProperties)>
  <!ATTLIST  VConcept             VCname              ID      #REQUIRED>
  <!ELEMENT  VRelationships       (VRelationship)+>
  <!ELEMENT  VRelationship        (AssocConcept)+>
  <!ATTLIST  VRelationship        VRelname    (IS-A|IS-PART-OF|Associative)
#REQUIRED>
  <!ELEMENT  AssocConcept         (#PCDATA)>
  <!ATTLIST  AssocConcept         VConcept    IDREF       #IMPLIED>
  <!ELEMENT  VProperties          (VPoid|VPord|VPref)+>
  <!ELEMENT  VPoid                (VDataType, VUnitType, PProperties)>
  <!ATTLIST  VPoid                VPname      ID          #REQUIRED>
  <!ELEMENT  VPord                (VDataType, VUnitType, PProperties)>
  <!ATTLIST  VPord                VPname      CDATA       #IMPLIED>
  <!ELEMENT  VPref                (#PCDATA)>
  <!ATTLIST  VPref                VPoid       IDREF       #IMPLIED>
  <!ELEMENT  VDataType            (#PCDATA)>
  <!ELEMENT  VUnitType            (#PCDATA)>
  <!ELEMENT  PProperties          (PProperty)+>
  <!ELEMENT  PProperty            (PDataType, PUnitType)>
  <!ATTLIST  PProperty            PPname      CDATA       #REQUIRED
                                  PCname      IDREFS      #REQUIRED
                                  PSname      IDREF       #REQUIRED
                                  CVCname     IDREFS      #REQUIRED>
  <!ELEMENT  PDataType            (#PCDATA)>
  <!ELEMENT  PUnitType            (#PCDATA)>

  <!ELEMENT  PhysicalSourceConfs  (PSource)+>
  <!ELEMENT  PSource              (PConcept)+>
  <!ATTLIST  PSource              PSname              ID      #REQUIRED>
  <!ELEMENT  PConcept             (PDataModel, Permission, Owner)>
  <!ATTLIST  PConcept             PCname              ID      #REQUIRED>
  <!ELEMENT  PDataModel           (#PCDATA)>
  <!ELEMENT  Permission           (#PCDATA)>
  <!ELEMENT  Owner                (#PCDATA)>
]>
```

Figure 6.3 The XML-DTD metadata dictionary structure.

6.3 Construction Rules

In order to govern the update operations of the well-formed and valid XML document, a set of construction rules is set up to administer the correctness and consistency. Since XML

document is tree-structured, the metadata dictionary contents are represented in conventional tree structure. The following formulations outline the rules in constructing the XML metadata dictionary document.

Let vc_m and vc_n be virtual concepts in domain ontology. The virtual concepts and relationships make up the nodes and links of the XML document tree as follows:

(a) Virtual Concepts

Referring to the XML-DTD structure in Figure 6.1, the `VConcepts` starts as the root of all virtual concepts. Other associated concept names stored in data elements of `AssocConcept` become the child nodes. The relationship between `VConcept` and `AssocConcept` denotes a one-way traversal. In other words, if $n(vc_m)$ and $n(vc_n)$ are designated to `VCname` and `AssocConcept`, respectively, it is not necessary to store $n(vc_n)$ and $n(vc_m)$ to `VCname` and `AssocConcept` in the reverse direction, that is,

$$Belong\_To(n(vc_m), VConcept.VCname) \wedge Belong\_To(n(vc_n), AssocConcept) \Rightarrow \neg(Belong\_To(n(vc_n), VConcept.VCname) \wedge Belong\_To(n(vc_m), AssocConcept))$$, where

- $Belong\_To(n(vc_m), VConcept.VCname)$ denotes the concept name of vc_m stored in the attribute `VCname` of `VConcept`.
- $Belong\_To(n(vc_n), AssocConcept)$ denotes the concept name of vc_n stored in the data element of `AssocConcept`.

(b) Relationships

To properly link the virtual concepts, the following formal guidelines are employed to preserve the structural design established in Section 6.1 as follows:

- If a concept vc_m associates with vc_n through the IS-A, IS-PART-OF, or Associative relationships, the concept names of vc_m and vc_n are designated to the attribute `VCname` and the data element `AssocConcept`, respectively, that is,

$$IS\text{-}A(vc_m, vc_n) \vee IS\text{-}PART\text{-}OF(vc_m, vc_n) \vee R_{11}(vc_m, vc_n) \vee R_{1N}(vc_m, vc_n) \Rightarrow Belong\_To(n(vc_m), VConcept.VCname) \wedge Belong\_To(n(vc_n), AssocConcept)$$

- For the N:M relationship, it is important to note that the N:M relationship holding its own properties apart from the participating concepts must be separately accounted for as a virtual concept in the ontology structure. Therefore, if a concept vc_k is a separate concept representing a relationship between the participating concepts vc_m and vc_n, the concept names of vc_k , vc_m, and vc_n are designated as follows:

  $R_{NM}(vc_m, vc_n) \Rightarrow (Belong\_To(n(vc_k), VConcept.VCname) \wedge Belong\_To(n(vc_m), AssocConcept)) \wedge (Belong\_To(n(vc_k), VConcept.VCname) \wedge Belong\_To(n(vc_n), AssocConcept))$