Context Representation, Transformation and Comparison for Ad Hoc Product Data Exchange

Jingzhi Guo and Chengzheng Sun
School of Computing and Information Technology
Griffith University, Brisbane, QLD 4111, Australia
Tel: +61-7-38753765/38756540
{J.Guo, C.Sun}@cit.gu.edu.au

ABSTRACT
Product data exchange is the precondition of business interoperation between Web-based firms. However, millions of small and medium sized enterprises (SMEs) encode their Web product data in ad hoc formats for electronic product catalogues. This prevents product data exchange between business partners for business interoperation. To solve this problem, this paper has proposed a novel concept-centric catalogue engineering approach for representing, transforming and comparing semantic contexts in ad hoc product data exchange. In this approach, concepts and contexts of product data are specified along data exchange chain and are mapped onto several novel XML product map (XPM) documents by utilizing XML hierarchical structure and its syntax. The designed XPM has overcome the semantic limitations of XML markup and has achieved the semantic interoperation for ad hoc product data exchange.

Categories and Subject Descriptors
H.5 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *Web-based interactions.* H.3.5 [**information Storage and Retrieval**]: Online Information services – *Sharing Data; Web-based services*.

General Terms
Documentation, Design, Languages

Keywords
Concept, semantics, product data integration, context representation, context transformation, context comparison, ad hoc product data exchange, electronic commerce, XML product map, XPM, electronic product catalogue

1. INTRODUCTION
Product data exchange is the precondition of business interoperation between Web-based firms [9][13]. However, millions of small and medium sized enterprises (SMEs) encode their Web-based product data in ad hoc formats for their *electronic product catalogues* (EPCs). The reason is simple: they are financially or technically difficult to join in any existing product standards [14]. This situation largely prevents SMEs from participating the emerging global electronic marketplaces and seriously weakens SMEs' survivability and competitiveness [11]. A realistic issue is thus how our researches can help SMEs participate the emerging global electronic marketplaces. In another word, how can we enable SMEs to successfully exchange their ad hoc product data?

Central to the issue of ad hoc product data exchange is how to represent, transform and compare the contexts of these ad hoc data from various SMEs [14].

Ad hoc product data have several important characteristics. (1) *Inconstant*: most ad hoc product data are encoded in heterogeneous EPCs following no standards, and easy to change without notice. (2) *Small-scale*: most ad hoc product data exist in heterogeneous EPCs of SMEs that only have several or tens of products. (3) *Irregular*: ad hoc product data are stored in different storage formats, which, in general, can be classified as XML product files, relational databases and coded Web pages. (4) *Heterogeneous*: firms often adopt their local languages or dialects to encode their ad hoc EPCs even though the data semantics might be the same. (5) *Numerous*: there are millions of heterogeneous EPCs distributed around the world such as in SMEs, of which each is a "semantic community" [14][27].

The characteristics of ad hoc product data indicate that a solution to the issue of context representation, transformation and comparison for ad hoc product data exchange is urgently needed.

Nevertheless, it is a great challenge to achieve such a solution. Most current approaches for product data exchange are mainly focused on enabling data exchange between various kinds of product standards. For example, Omelayenko and his collaborators [22][24][25] propose to integrate de facto industrial product standards by building the map relationships in the levels of documents, objects and vocabularies. Dogac et al advocate integrating several standards by exploiting RosettaNet's public processes and UDDI's public registries [6]. Bergamaschi et al [3] adopt semiautomatic approach to associate/map product terms of product classification standards that are differently classified or coded. Few researchers specialize in providing solutions to ad hoc product data exchange except some of our previous researches [12][13][14][15].

This paper aims to provide a solution to the above issue and is devoted to discussing how contexts can be represented, transformed and compared through a novel concept-centric catalogue engineering approach for ad hoc product data exchange between SMEs. This approach decomposes EPCs into sets of hierarchically represented concepts, which construct contexts for ad hoc product data exchange. The concept-centric catalogues are mod-

DocEng'03, November 20–22, 2003, Grenoble, France.

eled in a set of specially designed XML documents called XML PRODUCT MAP (XPM).

The remaining part is arranged as following: Section 2 discusses the previous works related to the theme of this paper. Section 3 lays the foundation of concept-centric context representation. Section 4 discusses XML representation strategy for global product catalogue. In Section 5, context transformation is discussed based on a novel XML product map. Section 6 discusses context comparison. An example is given to overview the involved operations in Section 7. The final section discusses some related works in catalogue engineering, concludes the paper and proposes several future works.

2. PREVIOUS WORKS

2.1 Concept-Centric Document Analysis

Conventionally, there are three kinds of analytical approaches for document engineering: document-centric analysis, data-centric analysis and document/data combined approach. "Document analysis is often conducted with the goal of abstracting a logical model from heterogeneous instances and encoding it as an SGML or XML schema. The schema enables the replacement of ad hoc, inconsistent or incomplete formatting with a stylesheet that applies presentation semantics in a consistent fashion to any instances that conforms to the schema" [9]. However, schema-based document analysis cannot detect semantic conflicts that cause data interoperation problems even though two schemas are structurally the same (cf. [10][17]). For example, we cannot guarantee that two refrigerators in two companies are semantically the same even though they apply for a same schema in which <!ENTITY refrigerator "An appliance, a cabinet, or a room for storing food or other substances at a low temperature"> is given. One firm may refer "refrigerator" as "domestic refrigerator" and the other may refer it to the meaning of "electronic portable cooler for traveling purpose", because we have no mechanisms to guarantee that two independent firms commit the schema definitions in exact meaning interpretation. More seriously, most SMEs develop their electronic product catalogues following no standard schemas. It is difficult and nearly impossible to apply for any stylesheets to transform millions of catalogue documents that are heterogeneously represented.

Data analysis inherits the same problems from document analysis, except that the granularity of semantic heterogeneity is smaller. Data analysis focuses on object levels, such as product representations but not catalogue representations. Detailed problems of semantic conflicts between product representations are documented in the works of [13][23].

To reduce semantic conflicts to achieve flexibility, exactness and evolvability for ad hoc product data exchange between various EPCs, we have proposed a concept-centric approach for catalogue engineering [12][13]. This approach regards a catalogue as a set of meta concepts uniquely identified by vectors in a vector tree defined in Definition 1. Catalogue representations are based on semantic concepts but not on schemas of documents or objects.

Definition 1: *Meta Concept*

A meta concept is a vector on a vector tree $(A_i^1, \ldots, A_i^k)$ where (1) *level* is $k \in \{1,\ldots,n\}$, (2) *position* is $i \in \{1,\ldots,n\}$ of each A, (3) *parent* is a vector $(A_i^1, \ldots, A_i^{k-1})$ and *ancestor* is a set of vectors $(A_i^1, \ldots, A_i^{k-x})$ with $1 \le x < k$ and $k \ne 1$, (4) *child* is a set of vectors $(A_i^1, \ldots, A_i^{k+1})$ and descendants is a set of vectors $(A_i^1, \ldots, A_i^{k+x})$ with $x \ge 1$, (5) *root* is a vector $(A_i^1, \ldots, A_i^k)$ with $k = 1$ and $i = 1$, and (6) *siblings* are a set of vectors $(A_i^1, \ldots, A_{x \in i}^k)$. For simplicity, we notate a vector as <k, i>.

We have mapped vectors onto concepts in a catalogue, and identify product concepts with *product concept identifiers*, notated as PID<k, i>, and identify attribute concepts with *attribute identifier*, notated as AID<m, n>. A complete concept is called *concept internal identifier* IID = (PID<k, i>, AID<m, n>), where PID<1, 1> is the root concept of a catalogue while AID<1, 1> is the root concept of a product.

2.2 Concept Exchange Framework

Based on the meta concept defined, we have proposed a concept exchange framework to exchange product data [13]. In this framework, concepts are differentiated as local concepts and common concepts. *Local concepts* are ad hoc generated in local product catalogues of various "semantic communities" [14][27] such as different SMEs, which are not semantically interoperable because they are ad hoc formatted and represented. *Common concepts* are generated based on standards in interoperable product catalogues of product catalogue providers and are used as semantic mediators for local product catalogues. Local concepts of different ad hoc local product catalogues interoperate each other through globalizing them into common concepts. New local concepts are generated by localizing common concepts, thus they can interoperate each other. The product data exchanges between different catalogue providers are achieved by maintaining a replicated document structure of common concepts.

2.3 Concept-Centric Product Data Model

Following the concept exchange framework [13], we have proposed a concept-centric product representation to decompose a product catalogue as a set of concepts: catalogues, products, attributes, and value types [12]. A product is electronically represented in a 3-tier triple (product concept, product annotation, product structure (attribute concept, attribute annotation, attribute structure (value concept, value annotation, value structure))). This product representation model has been revised in [15] and is restated as a model of *n-level 4-tuple* (product concept, product annotation, product link, structure (attribute concept, attribute annotation, attribute link, structure (attribute concept, attribute annotation, attribute link, structure (…)))). This revision has simplified the catalogue representation to ease the algorithm design and include spatial information of a concept.

2.4 Context Representation Model

The fact that product catalogues are Web-distributed in different semantic communities indicates that semantically same local concepts in different local catalogues may be heterogeneous in several ways: concept annotation, concept structure and different data sources. This situation is particularly severe for SMEs where product catalogues are ad hoc formatted and represented. This observation leads us to consider the semantic contexts of concepts in different semantic communities. In our research of [14], a context representation model is proposed to present the publicly understandable semantic contexts of product data. The semantic contexts are captured in two steps: transforming irregular local product definitions stored in various data sources to canonical local product representations, and transforming canonical local product representations into common product representations.

2.5 Transformation of Ad Hoc Product Data

Nevertheless, the context representation model [14] is only a high-level proposal. It does not describe the crucial transforma-

tion process of how to transform irregular local product definitions into canonical local product representations. It does not present the details about the public understandable context representation and comparison between canonical local product representations and public common product representations.

Our latest research [15] has proposed a concept-centric transformation approach that captures ad hoc product data in various local data sources and transforms them into canonical local product representations. The transformation makes concepts canonical and comparable within enterprise-wide systems. It has outputted a set of canonical local product representations that can further interact with public understandable common product representations.

However, the issue of how to represent publicly understandable contexts and how to compare these contexts for product data exchange is not discussed.

This paper is, therefore, to solve this problem by dedicating concept-centric context representation, transformation and comparison for ad hoc product data exchange.

3. CONTEXT FOUNDATION

How can an attribute, a product or a catalogue in one Web-based SME be understood in another? In another word, how can two heterogeneous contexts be recognized mutually? This section will propose a novel generic concept-centric context representation approach to lay the foundation of catalogue engineering.

3.1 Generic Concept Representation

A catalogue, a product, a product attribute or an attribute value is a concept. The semantics of a concept generally can be notated by three parts: a concept meaning, a concept structure that is again characterized by a set of concepts, and a set of particular values specifying those concepts that have no characteristics any more (leaf concept of the concept chain). This notation of a concept is consistent with Barthes's "system of signification" [1] and "order of signification" [2], where a denotation leads to a chain of connotations. For example, "refrigerator" is a meaning and "dimension, color" is a structure where "dimension, color" are again concepts. "Red" can be the particular value of the "color" concept because "color" cannot have any lower level concepts. In contrast, "dimension" may still have lower level concepts such as "width", "length" and "height".

More formally, we introduce three terms to describe concept semantics: denotation, connotation and particular.

- *Denotation* is a piece of natural language annotation that describes "the definitional, 'literal', 'obvious' or 'commonsense' meaning of a sign" [4] for a concept. It is generally specific to a certain "semantic community" [14][27] that shares the same perspectives to a concept, such as "refrigerator" for English speakers. Mapping onto a product representation [12][15], it can be defined in a functional relationship as "denotation: (identifier, annotation, link)→concept".
- *Connotation* is an internal structure of a concept that "is used to refer the social-cultural and 'personal' associations … of the interpreter's class, age, gender, ethnicity and so on" [4]. It expresses how a concept can be hierarchically decomposed into a set of lower level concepts to constrain and explicate the denotation. For example, "dimension, color" constrains the refrigerator by enumerating and explicating the features of a refrigerator. Connotation can be defined as a containment relationship "connotation: (concept, …, concept)→concept".
- *Particular* is a specific concept that refers to a concrete or conceptual entity instance or property value. For example, data "29"×33"×66", "silver" and "228 kw/h per year" are particular concepts of "dimension, color, energy consumption" for a certain specific refrigerator. A particular is an instantiation relationship "particular: instance → concept".

To be consistent with general used terms, we call those concepts without carrying particular values as *meta concepts* and the particulars as *instance concepts*. Meta concepts can be utilized to express publicly common meanings of a concept domain. An instance concept can refer to an instance of a meta concept at leaf.

3.2 Heterogeneous Concept Representation

Observing the business world, we can find that people exchange business data through a common language such as English. However, people may also make deals in their common national languages or community languages. Within a company, people even use dialects to format their business information. Mapping this observation onto business organizations and concept expressions, the concept heterogeneity problem can be depicted in Fig.1.

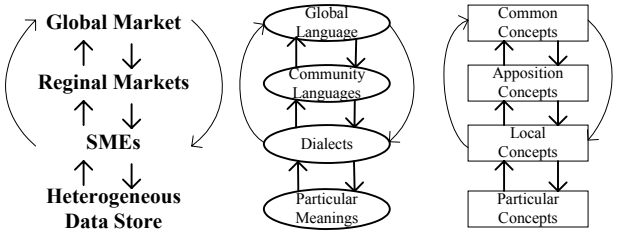

Fig. 1: Levels of heterogeneous concepts

Fig. 1 indicates that the meaning of a concept, if heterogeneously stored in different data stores in an SME, should be integrated for communication. What's more, if it desires to exactly convey a concept to another unknown SME for business interoperation, it should ascend the chain from enterprise to regional markets and/or global marketplace and then descend. This is because each organization or market is an independent semantic community where *heterogeneous concepts* are generated.

For convenience, we call concepts that are represented in global language are *common concepts*. Respectively, *apposition concepts* are from community languages, *local concepts* are from SMEs' dialects and *particular concepts* are from the particular meanings of heterogeneous data stores.

3.3 Generic Context Representation

3.3.1 Context Definition and Representation

A *context* is a concept definition relating to two perspectives to the definition. Since a concept is defined by a denotation, a chain of connotation and a set of particulars, a concept definition can be expressed as a vector tree that contains many sub-tree concepts (see Definition 1). Any perspectives of that full concept tree are specific contexts with respect to the full concept tree. For example, in Fig. 2, the sub tree of (C1, C2, C6, C7) is a context of the full concept tree (C1…C13) and (C1, C3, C8, C9, C10) is another context with respect to the concept definition of C1. Given a full concept tree, a perspective of the full concept tree, which takes only a sub tree, is a *partial context*. The perspective that takes the full concept tree is a *full context*. The relationship between partial context and full context is an inclusion relationship (⊆):

Partial Context ⊆ Full Context

Homogeneous context is a concept definition relating to two perspectives that happen in a same semantic community where the elements that construct perspectives are mutually understandable. Therefore, for homogeneous contexts, they are either partial or full. When comparing two partial contexts, they may be equivalent, intersected or disjoint.

Heterogeneous context is a concept definition relating to two perspectives that happen in two different semantic communities. Specifically, the elements that construct perspectives are different. For example, though (C1, C2, C6) and (Ac1, Ac2, Ac6) may refer to the same concept, the ways they form their perspectives are different. A heterogeneous context with respect to two perspectives is useless unless one of the perspectives is *transformed* to be consistent with another perspective.

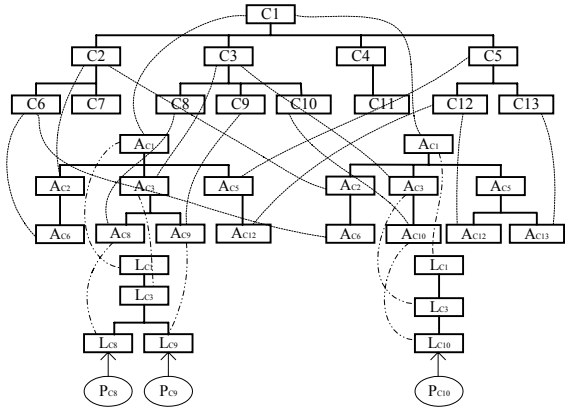

Fig. 2: Context analysis and representation

3.3.2 Context Transformation on Data Exchange Chain

Abstracted from Fig. 1, we have found that product concepts must experience a series of transformations moving on a chain of "particular concepts⇒local concepts⇒(apposition concepts)⇒ common concepts⇒(apposition concepts)⇒local concepts⇒ particular concepts" to enable the exchange of product concepts between two SMEs. We call this chain as *data exchange chain*. Mapping this chain onto the Fig. 2, two relationships could be found: *partial context relationship* and *heterogeneous context relationship*. For example in Fig. 2, if {P} are particular concepts, {L} are local concepts, {A} are apposition concepts and {C} are common concepts, then {P}⊆{L}⊆{A}⊆ {C}. Therefore, the relationship between them is partial. However, these concepts are also heterogeneous because they belong to different semantic communities. Therefore, they need to be transformed before context comparison.

Formally, these two contextual relationships between heterogeneous product concepts can be expressed by several specific contextual relationships:

Instance Context: particular concept⇒
partial local concept⊆local concept
Local Context: local concept⇒
partial apposition concept⊆apposition concept
Common Context: apposition concept⇒
partial common concept⊆common concept

where "⇒" is one-to-one transformation relationship and "⊆" is inclusion relationship.

These specific contextual relationships are the foundation of designing concept-centric electronic product catalogues to meet the requirements of exactness, flexibility and evovability [13].

4. XML REPRESENTATION STRATEGY FOR GLOBAL PRODUCT CATALOGUE

Applying the generic context representation, this section will develop an XML representation strategy for representing a global product catalogue system.

4.1 A Global Product Catalogue System

In [13], we have proposed a strategy for constructing interoperable electronic product catalogues for a global product catalogue system. In [15], we have simplified meta concept types of a product catalogue to "product" and "attribute". A global product catalogue system can thus be designed as a set of hierarchical meta product concepts and many sets of attribute concepts, where each catalogue concept is a product concept tree, a product concept is an attribute concept tree, and a leaf attribute concept may connects to many instance concepts.

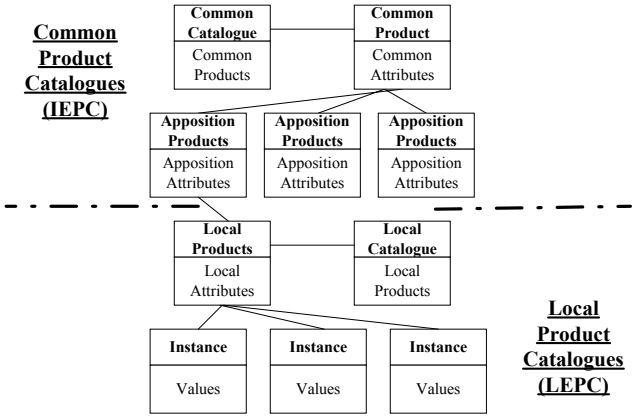

Fig. 3: A global product catalogue system: IEPCs and LEPCs

Following the product data flow on the data exchange chain and the nature of each semantic community along the chain, we divide the global product catalogue system shown in Fig.3 into following parts:

- *Heterogeneous data stores* may be XML files, relational databases or ad hoc web pages where dynamic instance values are generated from instance concepts.
- *Local catalogues* include *local products* where each has a set of *local attributes*. All of them are local concepts.
- *Apposition products* where each consists of a set *apposition attributes* are apposition concepts that different communities may understand.
- *Common catalogues* comprise *common products* where each consists of a set of attributes. They are all common concepts.

Common concepts are publicly understandable in a global scope. Apposition concepts are understandable in a certain community, while local concepts are private and only understandable in an enterprise. Instance concepts are special and specific, which are differentiated in different data stores.

Colloquially, we jointly call common catalogues, common products and apposition products as common product catalogues (IEPCs) to characterize the public nature, and collectively call local catalogues, local products and instances as local product catalogues (LEPCs) to characterize the private nature. It is obvi-

ous that the partial and heterogeneous contextual relationships exist in the underlying global catalogue system.

Context: LEPC ⇒ partial IEPC ⊆ IEPC

To have IEPCs and LEPCs interoperable, it is desirable to engineer the global product catalogue in a consistent and semantic understandable way.

4.2 Limitations of XML Markup

XML as a widely used markup language has a lot of standard Internet supports. The global catalogue system is Web-based and may be benefited from the standard XML language. Nevertheless, how to exploit XML's benefits should be discussed, especially how to overcome the limitations of XML markups for semantic descriptions.

- Self-describing markup approach for describing semantics is imaginable [5][26] for markup desingers. We cannot reply on markups to solve interoperability problems between different semantic communities such as SMEs. For example, a metamarkup <!ENTITY refrigerator "an appliance, a cabinet, or a room for storing food or other substances at a low temperature"> may not be understandable in different SMEs, especially when different enterprises are independently formatting their product catalogues.
- Inference to markup semantics is not guaranteed exact. "A DTD presents only a vocabulary and a *syntax* for that vocabulary – it does not provide a semantics for the vocabulary" [26]. In applying AI technologies to retrieve product information, inexactness and errors are found for terms [8].
- Semantic relationships between markups given by language designers are often not fully captured by software developers, and thus gradually software cannot be semantically interoperated with each other because of this cognitive problem.
- Heterogeneous document schemas are everywhere in independent SMEs. Different independent firms may use the same popular markups by accident but mean wholly differently. Synonymous and homonymous markups created in such circumstances are difficult to be structured in glossaries such as the proposed in ISO TC 37 [16], because standards are not referenced to write terms.

Reflected in global catalogue system design, these limitations cause semantic conflicts in ad hoc product data exchange [13][14].

4.3 A Strategy for XML Representation

The limitations of XML markup destine that we cannot directly use markup for semantic interoperation. However, we recognize that XML structure and its syntax can be utilized to represent IEPCs and LEPCs. In this section, we present a novel concept-centric representation strategy by mapping IEPCs and LEPCs onto XML language to solve semantic interoperation problem.

A global product catalogue system can be decomposed into two basic constructs: *concept* and *classifier*. Concepts are catalogue's elements and classifiers are the relationships between the concepts. A catalogue, a product, an attribute or a value is a concept, which can be denoted and connoted as discussed in Section 3. The structural relationship between these concepts is a parent-child relationship. Yen et al [32] in their research confirm this point: "from a structural point of view, the electronic catalog can be defined as a collection of classified information and presented in forms of a catalog tree". Therefore, it is plausible if we adopt a strategy to map the global product catalogue system onto an XML tree with each node as a concept. This strategy is natural and intuitive. To facilitate the mapping, we describe the strategy in several general rules in forms of XML DTDs:

Rule 1 (*connotation*): given a meta concept that are connoted by a set of hierarchically arranged meta concepts [12], then it can be mapped onto an XML document tree defined in an XML DTD:

<!ELEMENT concept (concept*)>

where the root concept element is a catalogue concept or a product concept which only allows to occur once. The connotation (a set of concepts) of the catalogue or product concept maps onto a set of child concept element nodes. A child concept element node is again connoted by a set of its direct child concept element nodes until to the leaf nodes.

This strategic rule naturally constructs a catalogue consisting of products or a product comprising attributes. For example:

```
<concept annotation= "refrigerator">
   <concept annotation= "price"/>
   <concept annotation= "dimension">
       <concept annotation= "width"/>
       <concept annotation= "length"/>
       <concept annotation = "height"/>
   </concept>
</concept>.
```

Rule 2 (*denotation*): given a meta concept that is denoted by a 4-tuple (concept, annotation, link, structure) [15], then its XML DTD is:

```
<!ATTLIST concept iid NMTOKEN #REQUIRED
                  annotation CDATA #REQUIRED
                  link CDATA #IMPLIED
                  structure CDATA #IMPLIED>
```

where *iid* is the NMTOKEN form of calculable IID(PID<k, i>, AID<m,n>). *Annotation* is the semantic description of the concept, which could be any natural languages or dialects dependent on the underlying semantic communities. *Link* is the Web URL address of the concept. *Structure* refers to an XML file that connotes the concept.

For example, a refrigerator in a French company can be:

```
<concept iid="1-52-14-15-1" annotation = "réfrigérateur" link="" structure="réfrigérateur.xml">.
```

Rule 3 (*classifier*): given a set of meta concepts that are arranged in an XML tree according to Rule 1, then each concept can be identified by an internal identifier IID<k, i> defined in Definition 1, making each IID equals to the concept element node position in the underlying XML.

Concept(IID) := NodePosition(concept)

Specifically, for a catalogue, the root PID<1, 1> corresponds XML document root position, and all product concepts PID<k, i> correspond other child concept element node positions. For a product, AID<1, 1> corresponds the XML document root node position and AID<m, n> correspond other XML concept element node positions. This rule naturally maps an XML tree to the vector concept tree that defines the identifiers of all concepts.

The benefits of this rule are: (1) the semantics of a concept can be replaced and dynamically traced through a calculable *PID* or *AID* by querying the XML concept element node position. The semantics described in the concept element position is exactly same as the semantics what IID identified but heterogeneously denoted elsewhere. Therefore, in any heterogeneous systems, semantics can be arbitrarily represented only if an IID in any systems are consistent. (2) IID is a vector that records all the information of current concept's ancestor information, which is genealike. For example, a refrigerator may be encoded as IID(1, 52, 14, 15, 1). Query the information of any ancestor is direct and effi-

cient. For example, in a given context, IID of "domestic kitchen appliances" will always have IID(1, 52, 14, 15) as the parent of IID(1, 52, 14, 15, 1). (3) The mutation of child concepts produces the natural evolution of a product or a catalogue, where the mutated semantics can be exactly captured in all governing heterogeneous systems. (4) Another possible side benefit is that its query speed for the semantics of a given term might be increased, because the concept node position is directly given for query (cf. [21]). The actual performance may depend on XML parsers in use.

Rule 4 (Instantiation): given an instance concept that exists in a certain data store, then this concept is instantiated by a leaf meta concept and is placed in an XML document stipulated by DTD:

<!ELEMENT concept (#PCDATA)>
<!ATTLIST concept locIID NMTOKEN #REQUIRED
link CDATA #REQUIRED
type NMTOKEN #REQUIRED>

where *locIID* is the identifier in a local catalogue, *link* is the current retrieved value location and *type* refers to the type of data.

This rule defines how the particular value of a leaf attribute is retrieved from heterogeneous data sources such as relational databases or other XML files. For example, an instantiation of a leaf meta concept may be:

```
<instance><concept locIID = "1-2-3p-2-2" link= "color" type= "code"><![CDATA[SELECT color FROM table]]></concept></instance>
```

to dynamically retrieve a value in run-time. This strategy maintains all legacy systems by plugging a data retrieval tool.

The concept-centric representation strategy guarantees that all LEPCs and IEPCs are semantically merged and classified by sharing IIDs that are dynamically generated from the concept element node positions of the underlying XML documents.

5. CONTEXT TRANSFORMATION

To implement the above XML representation strategy, we should understand how the catalogue components are related to each other, and what actions we should take to have different components work together for achieving consistent semantic flow along data exchange chain. In this section, we describe the catalogue component relationship in an *XML product map* (XPM) and transform the heterogeneous contexts between these components for semantic interoperation.

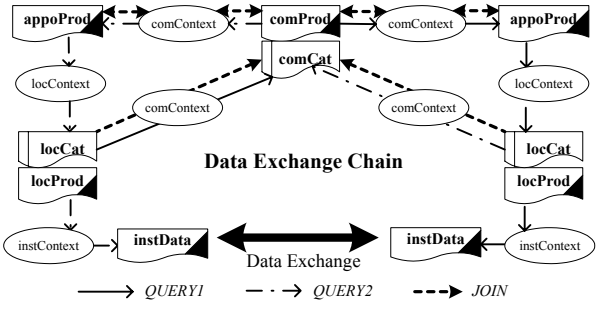

Fig. 4: XML product map

5.1 XML Product Map

The underlying global catalogue system depicted in Fig. 3 can be constructed by a series of XML documents that conform to the proposed XML representation strategy. Along with the data exchange chain, different XML files can be designed to correspond different semantic communities. The Fig. 4 illustrates this document flow by explicitly capturing the intentions and activities of semantic communities in an *XML product map* (XPM).

XPM defines all the relevant XML documents that are necessary for catalogue designers and users to either join the global catalogue systems or query semantic product data. Along with the data exchange chain, we describe two activities of catalogue designers and catalogue users: LEPC join in IEPC and LEPC users query ad hoc product data. In the following, we classify these XML documents and briefly discuss their general features.

XPM documents can be classified in two types according to the semantic data type and the ability of generating IID, shown as in Table 1. *Metadata documents* that provide the abstract semantics of ad hoc product data while *instance data documents* carry or generate the instantiated values referring to the corresponding abstract metadata. *IID or locIID generable documents* can generate unique IID/locIID for the newly insert concepts for the use of global catalogue system or local catalogue system while *IID/locIID un-generable documents* can only use the generated IID/locIID to refer their established concepts.

Table 1: XPM document type

Document Type	Semantic Data Type	IID Generability
Metadata	locCat, locProd, comCat, comProd, appoProd	
Instance Data	instData	
IID Generable/ locIID generable (*)		comCat, comProd, locCat*, locProd*
IID Un-Generable		instData, appoProd

In details, these documents have the following features:

The comCat.xml and comProd.xml. The comCat.xml consists of a set of product concepts while a comProd.xml consists of a set of attribute concepts. They are parts of IEPC that provides public services for LEPC to join and query. They generate common concept IID when a new concept is inserted.

The appoProd.xml. These documents are created by local catalogue designers when joining their LEPC into IEPC and are placed in IEPC for public access of the local products. An appoProd.xml does not generate any concept IID but consist of a set of apposition concepts semantically equal to common concepts.

The locCat.xml and locProd.xml. They are part of LEPC and the generated IID is only for local use and called **locIID**. These documents can be independently created to form an LEPC. Nevertheless, when an LEPC joins in IEPC, the global IID must be imported into these documents in connection with locIID.

The InstData.xml. This instance data document carries the dynamic values of instance concepts that are instantiated from leaf meta attribute concepts in locProd.xml.

As we have mentioned in Section 4.1, there are partial and heterogeneous contextual relationships in the global catalogue system. These contextual relationships are activated when JOIN or QUERY operations are executed on XPM (see Fig. 4). In Section 5.3 and 5.4, we will discuss the heterogeneous relationship in terms of context transformation. In Section 6, we will discuss the partial relationship in terms of context comparison.

5.2 Heterogeneous Context Transformation in JOIN Operations on XPM

JOIN operations on LEPC and IEPC are to insert new product or attribute concepts into IEPC. It involves context transformation relating to LEPC and IEPC. Often, JOIN operation can be divided into three sequential sub operations: browse, retrieve and insert.

BROWSE operations on comCat or comProd are to find which concept on comCat and comProd should be decided as the new concept insert point. The target is to find the desired common concepts in the common concept set that are semantically equivalent to the intended concepts for insertion. It is a process of building heterogeneous common context,

RETRIEVE operations on the comCat, comProd or appoProd are to extract the insert point concept information. The operation retrieves the information of the common concept Concept(iid, ∑sibling iids, ∑child iids) and sends them to the local designer's user interface for context transformation. It is a process of heterogeneous context representation.

INSERT operations on appoProd, locCat and/or locProd are to insert the new local concept into appoProd for public use and into locCat and locProd to relate the common concept and the new local concept. This is a process of heterogeneous context transformation, which makes heterogeneous concepts understandable each other.

There are two situations of insertion: (1) to add a new common concept, and (2) to insert an apposition concept conforming an existing common concept. For the first situation, an LEPC designer is prohibited to add any new common concepts to comCat or comPro. Adding common concepts is only allowed for IEPC designers not LEPC designers. The LEPC designers, however, can propagate the insert requirements to IEPC designers by launching a collaboration process [12].

Inserting a concept involves two processes: inserting into appoProd and inserting into locProd. These two processes form a mapping between local concept and common concept by mutual heterogeneous context transformation. From the viewpoint of appoProd, an apposition concept is inserted by transforming the common context of a local concept relating to a common concept. For example, an apposition concept is inserted as:

AppoProd.xml

<concept iid="" annotation="community language" link="" locLink=""/>

where local concept addressed by "locLink" and annotated by "community language" has been inserted through the transformation of a common concept identified by public "iid".

Viewing from locCat/locProd, the insert process is to transform heterogeneous common context into a homogeneous common context. The transformation is to relate common concept and local concept by mapping IID in IEPC onto locIID in LEPC. For example, a local concept is inserted as:

LocProd.xml

<concept locIID="" annotation="local dialects" link="" iid=""/>

where a common concept is transformed to a local concept by relating to a public "iid".

5.3 Heterogeneous Context Transformation in QUERY Operations on XPM

A QUERY operation on LEPC triggers a series of sub heterogeneous context transformation operations between adjacent heterogeneous document sets of LEPC and IEPC.

Given a set of XPM documents, a QUERY operation then requests remote data along the data exchange chain (see Fig. 4) in the following way: LocCat/locProd⇒comCat/comProd⇒ appoProd⇒locCat/locProd ⇒instData. To facilitate the data query, the adjacent heterogeneous documents should be able to interoperate each other for passing the operation semantics. Our strategy to meet this requirement is to build heterogeneous contexts between the adjacent heterogeneous documents and then transform them. Specifically, the strategy is implemented by building the query concept set and then being compared with the adjacent concept set to establish the concept identifier mapping. The detailed transformation mechanism is documented in the following steps:

Algorithm (*Heterogeneous Context Transformation*). Assuming that the heterogeneous concept sets as heterogeneous contexts for QUERY flowing on data exchange chain are ordered as:

(1) locConcept(locIID<m, n>, IID<k, i>, {local denotation})
(2) comConcept(IID<k, i>, {common denotation})
(3) appoConcept(<IID<k, i>, {apposition annotation})
(4) locConcept(locIID<m, n>, IID<k, i>, {local denotation})
(5) parConcept(locIID<m, n>, {values})

Then, the adjacent heterogeneous contexts are orderly transformed (⇒) by checking the inclusion (⊆) and equivalence (=) relationships between two sets of concept identifiers:

(1) locConcept⇒comConcept if IID(locConcept)⊆IID(comConcept)
(2) comConcept⇒appoConcept if IID(appoConcept)⊆IID(comConcept)
(3) appoConcept⇒locConcept if IID(appoConcept)=IID(locConcept)
(4) locConcept⇒parConcept if locIID(parConcept)⊆locIID(locConcept)

By this heterogeneous context transformation algorithm, heterogeneous semantic representations contained in denotations are transformed but unchanged for different semantic communities. This algorithm has not only solved the semantic conflicts between independent SMEs, but also given the solution to some problems of legacy systems, customizations and personalization.

6. CONTEXT COMPARISON

Heterogeneous context transformation has resolved the problem of heterogeneous semantic representations. Nevertheless, when multiple parties are involved, multiple contexts may need parallel comparison to facilitate what reactions the receiver should take. The process of handling multiple contexts at the destination is context comparison, which assumes all the compared contexts are homogeneous.

6.1 Homogeneous Context Comparison

Given two homogeneous contexts $context_1$ and $context_2$, the context comparison between them is then defined by mapping of a couple:

$$\mathbf{ctxComp(ctx_1, ctx_2) = Map<(locIID_1, V_1), (locIID_2, V_2)>}$$

where locIID and V are two components of the compared contexts.

A context is a possibly partial context compared with another context in the same semantic community. If the compared contexts are heterogeneous, they should be transformed first. A context can be simplified as ctx = (locIID, V) because concept identifier and instance data are enough to form a context structure. For this reason, context comparison is based on the components of locIID and V of product representations.

A *Map* is a *specificity relationship* between two contexts. Let $ctx_1 = <(locIID_1, V_1), (locIID_2, V_2), \ldots, (locIID_m, V_m)>$ and $ctx_2 = <(locIID'_1, V'_1), (locIID'_2, V'_2), \ldots, (locIID'_n, V'_n)>$, then:

$$\mathbf{ctx_1 \le ctx_2 \text{ if } ctx_1 \text{ is not more specific than } ctx_2 \text{ in terms of } ctx_2}$$

Since a context is a relative expression of a product concept, it can be compared both in *meta concept* level or *particular concept* level. When we compare contexts in meta concept level, we compares the two set of locIIDs to present:

meta concept specificity:

$$\mathbf{locIID_1 \le locIID_2 \text{ iff } locIID_1 \subseteq locIID_2}$$

When we compare contexts in particular concept level, we compares the two sets of values Vs to present:

particular concept specificity:

$$V_1 \leq V_2 \text{ iff } V_1 \leftarrow locIID_1 \subseteq V_2 \leftarrow locIID_2$$

The context comparison is to find out the value of the Map. How is one context more specific than the other context? Specificity relationship defines the extent of the similarity between two product concepts and the extent of interoperability between two product representations.

The following section will go deep to examine the specificity relationship between two contexts.

6.2 Semantic Similarity

Two contexts are compared to find out the semantic similarity between two contexts' underlying product concepts by building a Map between two sets of (locIID, V). The value of Map determines the semantic similarity between two product concepts.

In comparing contexts, we are interested in finding two types of semantic similarity exposed by meta concept specificity relationship and particular concept relationship: *meta concept similarity* and *particular semantic similarity*.

A detailed classification of these two relationships will be beneficial to business interoperation. To understand meta concept similarity is particularly useful for processing business inquiry and product search where no specific needs of product values are required. For example, an inquiry for refrigerator can be refrigerator(price (currency), color, dimension(width, length, height), capacity(gross, freezing)). No particular values are needed. However, particular semantic similarity is important when a business quotation is received for comparing the receiver's product data. Detailed semantic similarity relationships between two contexts can be classified in Table 2.

Table 2: Semantic similarity between two product concepts

Relationship	$ctxComp(ctx_1, ctx_2)$
Meta equivalence	$Map(locIID_1, locIID_2) = ALL$
Meta intersection	$Map(locIID_1, locIID_2) = SOME$
Meta disjoint	$Map(locIID_1, locIID_2) = NONE$
Value equivalence	$Map(V_1 \leftarrow locIID_1, V_2 \leftarrow locIID_2) = ALL$
Value intersection	$Map(V_1 \leftarrow locIID_1, V_2 \leftarrow locIID_2) = SOME$
Value disjoint	$Map(V_1 \leftarrow locIID_1, V_2 \leftarrow locIID_2) = NONE$

In Table 2, three types of semantic similarities are given:

(1) ALL expresses the equivalence relationship between two contexts either for meta concept level or for particular concept level. It is the strongest semantic similarity where two contexts' underlying product representations are fully interoperable in their respective concept levels.

(2) SOME expresses intersection relationship between two contexts either for meta or particular concept levels. It is weaker to semantically relate two contexts' underlying product representations. Whether it is useful for semantic interoperation depends on the users' rules to utilize the intersected semantics.

(3) NONE expresses disjoint relationship between two contexts either for meta or particular concept levels. It shows the weakest semantic relevance between two contexts' underlying product representations. Users are able to utilize this relationship to prevent product information overloading from Internet and minimize the systems traffic to improve systems performance.

A more detailed classification can be given. For example, under meta equivalence, there can be value equivalence, intersection and disjoint. These can be utilized to design different ad hoc data exchange scenarios.

Common Product Datalogue (IEPC)

comCat.xml (English)

```
<concept iid="1" annotation="UNSPSC" link="boo.com/en/unspsc.xml"  scope="en">
   ... ...
 <concept iid="1-52" annotation="domestic appliances and supplies and
      consumer electronic products" >
   ... ...
  <concept iid="1-52-14" annotation="domestic appliances" >
    ... ...
   <concept iid="1-52-14-15" annotation="domestic kitchen appliances" >
     <concept iid="1-52-14-15-1" annotation="refrigerator" structure
         ="boo.com/en/product/1-52-14-15-1.xml"/>
   </concept></concept></concept></concept>
```

comProd.xml (English)

```
<concept iid="1-52-14-15-1" annotation="refrigerator"
          link="boo.com/en/product/1-52-14-15-1.xml" scope="en">
 <concept iid="1-52-14-15-1p-1" annotation="price">
  <concept iid="1-52-14-15-1p-1-1" annotation="currency"/>
  <concept iid="1-52-14-15-1p-1-2" annotation="value"/>
 </concept>
 <concept iid="1-52-14-15-1p-2" annotation="gross capacity"/>
 <concept iid="1-52-14-15-1p-3" annotation="color"/>
</concept>
```

appoProd.xml 1(French)

```
<concept iid="1-52-14-15-1" link="boo.com/fr/appo/fr1-com/1-52-14-15-1.xml"
                            locLink="fr1.com/refrigerateur.xml"/>
<concept iid="1-52-14-15-1p-1"/>
<concept iid="1-52-14-15-1p-1-1"/>
<concept iid="1-52-14-15-1p-1-2"/>
<concept iid="1-52-14-15-1p-3"/>
```

appoProd.xml 2 (English)

```
<concept iid="1-52-14-15-1" link="boo.com/en/appo/en1-com/1-52-14-15-1.xml"
                            locLink="en1.com/fridge.xml"/>
<concept iid="1-52-14-15-1p-2"/>
<concept iid="1-52-14-15-1p-3"/>
```

Local Product Datalogue (LEPC)

locProd.xml 1 (French)

```
<concept iid="1-52-14-15-1" annotation="refrigerateur" link="fr1.com/refrigerateur.xml"
          locID="1-3-4-5-5"/>
<concept iid="1-52-14-15-1p-1" locID="1-3-4-5-5p-1" annotation="prix"/>
<concept iid="1-52-14-15-1p-1-1 locID="1-3-4-5-5p-1-1" annotation="devise"/>
<concept iid="1-52-14-15-1p-1-2" locID="1-3-4-5-5p-1-2" annotation="valeur"/>
<concept iid="1-52-14-15-1p-3" locID="1-3-4-5-5p-2" annotation="couleur"/>
```

locProd.xml 2 (English)

```
<concept iid="1-52-14-15-1" annotation="fridge" link="en1.com/fridge.xml"/>
<concept iid="1-52-14-15-1p-2" locID="1-9-7-4-3p-1" annotation="gCap"/>
<concept iid="1-52-14-15-1p-3" locID="1-9-7-4-3p-2" annotation="clr"/>
```

instData.xml 1

```
<concept locID="1-3-4-5-5p-1-1"
          link="fr1.com/data/refrigerateur.xml#1-3-4-5-5p-1-1">Fr</concept>
<concept locID="1-3-4-5-5p-1-2"
          link="fr1.com/data/refrigerateur.xml#1-3-4-5-5p-1-2">2500</concept>
<concept locID="1-3-4-5-5p-2"
          link="fr1.com/data/refrigerateur.xml#1-3-4-5-5p-1-2">white</concept>
```

instData.xml 2

```
<concept locID="1-9-7-4-3p-1" link="en1.com/data/fridge.xml#1-9-7-4-3p-1">
     <![CDATA[SELECT gCap FROM fridge WHERE quote="normal"]]</concept>
<concept locID="1-9-7-4-3p-2" link="en1.com/data/fridge.xml#1-9-7-4-3p-2">
     <![CDATA[SELECT clr FROM fridge]]</concept>
```

Fig. 5: An XPM Example

7. AD HOC PRODUCT DATA EXCHANGE ON XPM: AN EXAMPLE

The major purpose of representing, transforming and comparing contexts is to exchange ad hoc product data for business interoperation. In this section, we discuss four concept-centric context operations of context query, concept insert, quotation request and offer response by using the example data in Fig. 5 to illustrate the ad hoc data exchange cycle.

QUERY. An operation on an LEPC queries remote contexts from IEPC by a "pull" strategy. Suppose that an SME (en1.com) is querying a white refrigerator by sending a query operation.

(1) Query((1-9-7-4-3p-2, white, receiveURL, en1.com)) //prepare data
(2) Transform: 1-52-14-15-1p-3$\leftarrow$1-9-7-4-3p-2 //instContext transform
(3) Send(1-52-14-15-1p-3, white, receiveURL, en1.com) //to comCat
(4) Validate: 1-52-14-15-1p-3 $\subseteq$ IID // common context transform
(5) Match: Enumerate (1-52-14-15-1p-3 = IID(appoProd - en1.com)) // appoContext transform to find fr1.com for meta equivalence
(6) Compare(white, Values(<locLink(fr1.com)$\leftarrow$1-52-14-15-1p-3 >)) // context comparison to find value equivalence

(7) Return: receiveURL←locLink(fr1.com)

INSERT. An operation on an LEPC inserts a concept to appoProd and locProd to enrich a context representation. Suppose that en1.com wants to inquire the price of the white refrigerator from fr1.com. When it makes the inquiry, it finds that it lacks of price concept and then inserts price concept.

(1) Browse: Price←comProd("refrigerator")←comCat("en")
(2) Retrieve: IID("price"), IID("currency"), IID("value")
(3) Insert on locProd/appoProd: IID("price")←childOf("refrigerator") etc. // since no instData, no locLinks are inserted in appoProd.
(4) Create on locProd: locIID←IID and annotation←locIID // for (3))/(4), only metadata inserted.

REQUEST. An operation on an LEPC requests a response from the remote SME by a "push" strategy. Suppose that en1.com tests whether fr1.com can satisfy its product requirement by including the information of the inquirer for fr1.com's decision.

(1) Request(request, inquirerInfo, fr1.com, <(1-52-14-15-1p-1-1, USD, receiveURL), (1-52-14-15-1p-1-2, 300, receiveURL), (1-52-14-15-1p-3, white, receiveURL)>) // send request and prepare blank response sheet
(2) Transform(<(locIID←IID, annotation←locIID, value("en1.com"))>) // transform context, and copy inquierInfo and receiveURL to make inquiry sheet. The "USD" and "Fr" need currency conversion tool.
(3) Compare(<(value("en1.com"), value←locIID)>) // compare two contexts by considering rules for en1.com if any

RESPONSE. An operation on LEPC responds a request operation from a remote SME. Suppose that fr1.com responds the request from en1.com.

(1) Response(<(1-52-14-15-1p-1-1, USD, receiveURL), (1-52-14-15-1p-1-2, 350, receiveURL), 1-52-14-15-1p-3, white, receiveURL)>)

The operations of query, insert, request and response constitute the major operations on XPM documents. Other operations can be developed in future to enhance the functionalities.

8. DISCUSSION AND CONCLUSION

Concept-centric catalogue engineering approach to representation, transformation and comparison of contexts for ad hoc data exchange between SMEs is novel and promising.

In constructing electronic product catalogues (EPC) for exchanging product data, traditional approaches focus on product ontologies and vocabularies for product data interoperation. For example, smart catalogue [18] dynamically maps product terms from multiple ontologies through facilitators. This approach cannot prevent the interoperability problem between multiple ontologies. MEPCs [19] and Internet EPC [31], on the other hand, provide a global shared vocabulary by statically mapping product terms through one or more mediators. By this approach, all enterprises can interoperate with each other if they stick to and have no misunderstandings about the shared ontology. Both approaches need enterprises to have available ontologies of the target interoperating systems. However, ontology approach may be expensive and technically difficult for most SMEs. In addition, the ad hoc nature of SMEs indicates that SMEs often follow no rules to encode their product catalogues.

Recent approaches for constructing EPCs for product data exchange are more industry-oriented, which emphasizes on the product standards such as cXML (www.cxml.org), xCBL (www.xcbl.org), UNSPSC (www.unspsc.com) and ecl@ss (www.eclass-online.com). Nevertheless, the proliferation of product standards leads to the severe interoperability problems between multiple product standards [6][8][20][23][28][29]. Integration of product standards becomes an important issue for many researchers [3][7][8][24][25][30].

Concept-centric catalogue engineering approach for exchanging ad hoc product data by context representation, transformation and comparison has overcome the drawbacks of ontology and standard approaches, and has given several contributions to the research community. It enables ad hoc product data to be joined into common product catalogues in a dynamic, evolvable and collaborative way [12][13]. It allows local catalogue designers to maintain all the local information relevant only to its own semantic community [14]. It makes it possible to connect to different ad hoc product data stores for dynamic value generation [15]. What's more, it enables context comparison to find out the semantic similarities between several product representations. This approach has achieved product interoperability requirements of exactness, evolvability and flexibility proposed in [13].

There is a limitation for this approach. Based on our current research, this approach is more suitable for linking ad hoc product data in SMEs where they only have tens of or several hundred of product representations. More product representations may involve higher labor costs. Therefore, how to enlarge the application scope for this approach needs further research.

In summary, this paper has proposed a novel concept-centric catalogue engineering approach for representing, transforming and comparing contexts for ad hoc product data exchange. It explains how concepts are represented in both generic and heterogeneous forms and how concepts are organized to form the contexts of product representations. This paper has also proposed a novel XML product map (XPM) for representing contexts. Based on XPM, a global product catalogue system has been developed, which contains the common product catalogue (IEPC) and local product catalogues (LEPC). Contexts from LEPCs can be transformed between different semantic communities through the IEPC. Incoming contexts from other semantic communities thus can be compared. The processes of context representation, transformation and comparison are enabled by a collection of context operations such as query, insert, request and response. These operations constitute the foundation of ad hoc product data exchange between different semantic SMEs.

This paper is a research of concept-centric catalogue engineering approach. Several research issues are still urgent for exploration. First, a more accurate protocol for XPM document exchange is needed to optimize the parameters of various input/output operations between different semantic communities. Second, the replicated systems should be developed to allow collaboration between IEPCs for the new concept development. Third, how to integrate XPM with current product standards should be researched for enlarging the application scope. These issues are invitations to researchers for concept-centric catalogue engineering approach.

9. ACKNOWLEDGEMENT

We thank the anonymous reviewers for their insightful comments, which were invaluable for improving the presentation of this work.

10. REFERENCES

[1] Barthes, R. *Elements of Semiology*. Hill and Wang, English version 1967.

[2] Barthes, R. *Mythologies*. Hill and Wang, English version 1972.

[3] Bergamaschi, S., Guerra, F. and M. Vincini. A Data Integration Framework for e-Commerce Product Classification. In:

Horrocks and J. Hendler (Eds.): *ISWC 2002*, LNCS 2342, Springer-Verlag Berlin Heidelberg 2002, 379-393.

[4] Chandler, D. *Semiotics for Beginners*. http://www.aber.ac.uk /media/Documents/S4B/semiotic.html.

[5] Coombs, J.H., Renear, A. H. and S.J. DeRose. Markup Systems and the Future of Scholarly Text Processing. *Communications of the ACM* 30, 1987, 933-947.

[6] Dogac, A. and I. Cingil. A Survey and Comparison of Business-to-Business E-Commerce Frameworks. *ACM SIGEcom Exchanges*, Vol. 2.2, 2001, 16-27.

[7] Dogac, A., Tambag, Y., Pembecioglu, P., Pektas, S., Laleci, G., Kurt, G., Toprak, S. and Y. Kabak. An ebXML Infrastructure Implementation through UDDI Registries and RosettaNet PIPs'. In: *Proceedings of 2002 ACM SIGMOD International Conference on Management of Data*, Madison, Wisconsin, USA, 2002, 512-523.

[8] Fensel, D., Ding, Y., Omelayenko, B., Schulten, E., Botquin, G., Brown, M. and A. Flett. Product Data Integration in B2B E-Commerce. *IEEE Intelligent Systems* 16(4), July/August 2001, 54-59.

[9] Glushko, R. and T. McGrath. Document Engineering for e-Business. *ACM DocEng'02*, McLean, Virginia, USA, November 8-9, 2002, 42-48.

[10] Goh, C., Madnick, S. and Siegel, M. Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment. *ACM CIKM'94*, Gaitherburg, MD, 1994, 337-346.

[11] Guo, J. and C. Sun. Measurement Models for Survivability and Competitiveness of Very Large E-Marketplace. In: *Proceedings of Computational Science - ICCS 2003*, LNCS 2658, Springer-Verlag Berlin Heidelberg, 2003, 802-811.

[12] Guo, J. and C. Sun. Collaborative Product Representation for Emergent Electronic Marketplace. In: *Proceedings of 16th Bled Electronic Commerce Conference: eTransformation*, Bled, Slovenia, June 9-11, 2003, 847-859.

[13] Guo, J. and C. Sun. Concept Exchange: Constructing Interoperable Electronic Product Catalogues in an Emergent Environment'. In CEC'03: *Proceedings of the IEEE International Conference on E-Commerce*, IEEE Computer Society, 2003, 165-172.

[14] Guo, J. and C. Sun. Context Representation of Product Data. *ACM SIGEcom Exchanges*, Vol. 4.1, 2003, 20-28.

[15] Guo, J. and C. Sun. Transforming Ad Hoc Product Data into Canonical Local Product Representations. *Submitted*.

[16] ISO TC 37. http://www.iso.ch/iso/en/stdsdevelopment /tc /tclist /TechnicalCommitteeDetailPage.Technical CommitteeDetail? COMMID=1459.

[17] Kashyap, V. and A. Sheth. Semantic and Schematic Similarities between Database Objects: A Context-Based Approach. *The VLDB Journal* 5, 1996, 276-304.

[18] Keller, A. and M. Genesereth. Multivendor Catalogs: Smart Catalogs and Virtual Catalogs. *EDI Forum: Journal of Electronic Commerce* 9(3), 1996, 87-93.

[19] Linche, D. and B. Schmid. Mediating Electronic Product Catalogs. *Communications of the ACM* 41(7), 1998, 86-88.

[20] Ng. W. K., Yan, G. and E. Lim. Heterogeneous Product Description in Electronic Commerce. *ACM SIGEcom Exchanges*, Vol. 1.1, 2000, 7-13.

[21] Noga, M., Schott, S. and W. Löwe. Lazy XML Processing. *ACM DocEng'02*, McLean, Virginia, November 8-9, 2002, 88-94.

[22] Omelayenko, B. RDFT: A Mapping Meta-Ontology for Business Integration. In: *Proceedings of the Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at the 15th European Conf. on Artificial Intelligence*, Lyon France, 23 July 2002, 76-83.

[23] Omelayenko, B. and D. Fensel. An Analysis of B2B Catalogue Integration Problems: Content and Document Integration. In: *Proc. of Int'l Conf. on Enterprise Information Systems (ICEIS-2001)*, Setubal, Portugal, July 7-10, 2001.

[24] Omelayenko, B. and D. Fensel. A Two-Layered Integration Approach for Product Information in B2B E-Commerce. In: *Proc. of the 2nd Int'l Conf. on Electronic Commerce and Web Technologies*, LNCS 2115, Munich, Germany, September 4-6, 2001, 226-239.

[25] Omelayenko, B., Fensel, D. and C. Bussler. Mapping Technology for Enterprise Integration. In: *Proc. of the 15th Int'l FLAIRS Conf.*, Pensacola, FL., May 14-16, 2002, 419-424.

[26] Renear, A., Dubin, D., Sperberg-McQueen, C. M. and C. Huitfeldt. Towards a Semantics for XML Markup. *ACM DocEng'02*, McLean, Virginia USA, November 8-9, 2002, 119-126.

[27] Robinson, M. and L. Bannon. Questioning Representations. In Bannon, Robinson and Schmidt (eds) *Proceddings of the Second European Conference on CSCW*, 1991, Dordrecht: Kluwer: 219-233.

[28] Schulten, E., Akkermans, H., Guarino, N., Botquin, G., Lopes, N., Dörr, M. and N. Sadeh. Call for Participants: The E-Commerce Product Classification Challenge. *IEEE Intelligent Systems* 16(4), July/August 2001, 86-c3.

[29] Shim, S., Pendyala, V., Sundaram, M. and J. Gao. Business-to-Business E-Commerce Frameworks, *IEEE Computer* 33(10), 2000, 40-47.

[30] Somers, H. (Ed). *Terminology, LSP and Translation: Studies in Language Engineering in Honour of Juan C. Sager*. John Benjamins Publishing Company, Amsterdam/Philadelphia, 1996.

[31] Stanoevska-Slabeva, K. and B. Schmid. Internet Electronic Product Catalogs: an Approach beyond Simple Keywords and Multimedia. *Computer Networks* 32, 2000, 701-715.

[32] Yen, B. and R. Kong. Personalization of Information Access for Electronic Catalogs on the Web. *Electronic Commerce Research and Applications* 1, 2002, 20-40.