

Deconstruction and Reconstruction of Heterogeneous Electronic Product Catalogues for Semantic Interoperation

Jingzhi Guo, Chengzheng Sun and David Chen

School of Computing and Information Technology, Griffith University, Australia

{J.Guo, C.Sun, D.Chen}@griffith.edu.au

Abstract

A very important issue in constructing global electronic markets is to enable semantic interoperation between fragmented electronic markets by constructing an interoperable electronic product catalogue (EPC) for product data exchange. Nevertheless, the heterogeneous EPCs are highly complex, which causes the severe semantic inconsistency that prevents from semantic interoperation between them. To reduce complexity and to provide a solution to semantic consistency maintenance, this paper has developed a theory of deconstruction and reconstruction. The theory deconstructs heterogeneous EPCs into ordered concepts, structures and semantic relations by a proposed articulation approach. It reconstructs the deconstructed units into a novel CONEX framework for component EPC integration. The CONEX maintains semantic consistency amongst heterogeneous component EPCs in structure, concept and context.

1. Introduction

A very important issue in constructing global electronic markets is to enable semantic interoperation between fragmented electronic markets [9] by constructing an interoperable electronic product catalogue (IEPC) for the semantically consistent product data exchange [7]. An IEPC system consists of multiple interdependent *component EPCs* that have the characteristics of autonomy [15] and emergence [7]. Autonomy and emergence respectively lead to the spatial heterogeneity and temporal heterogeneity, which is often presented as semantic inconsistency, resulted from the differences of schemas [12], semantics [11] and contexts [8]. The effect of semantic inconsistency makes the business interoperation difficult.

This paper aims to investigate complex semantic consistency issue in EPCs, which is fundamental to constructing global electronic markets. The high degree of complexity results in the difficulty of designing a workable IEPC system that enables semantic interoperation for accurate product exchange. Complexity between heterogeneous EPCs can be demonstrated in the following.

Firm1: fridge (id: 222, clr: blue, prc: 300)

Firm2: réfrigérateur (art: x111, couleur : bleu, prix : 300)

It is uncertain to tell whether the representations are semantically same or different because a correct answer must depend on each firm's understanding in their situated context. If two firms have never cooperated before, they may have the following understandings:

(1) If Firm1 knows French, it may understand that Firm2 has a "fridge" specification where its "couleur" = "clr" and "bleu" = "blue". It is not sure whether value "300" of its "prix = prc" is the same as its own "300", because Firm1 implicitly refers "300" to US\$300 and cannot conclude what currency of Firm2 refers to. Firm1 also does not know what "art" means.

(2) If Firm2 knows English, it may infer Firm1 has a "réfrigérateur" specification if it understands "fridge" as "refrigerator". However, this inference may be wrong if "fridge" in Firm1 does not refer to "refrigerator". Firm2 cannot infer or understand the details of "fridge" specification of Firm1 because "id", "clr" and "prc" are only understandable in Firm1's own context.

(3) If no party understands the language of the opposite side, there is no understanding between Firm1 and Firm2.

The example exhibits that the issue of heterogeneous product data is extremely complex. This problem arises from where the concept interpretation of a piece of product data is context dependent. A product concept producer is impossible to imagine all contexts of product concept consumers and a product concept consumer is difficult to correctly infer the contexts of product concept producers [8]. The paper refers above related issues to *semantic consistency problem*.

To resolve this problem, the paper proposes a *deconstruction and reconstruction theory*. Deconstruction is defined as a methodology of articulating complex heterogeneous EPCs into concepts, structures and semantic relations that are easy to be manipulated for EPC reconstruction. Reconstruction is defined as a methodology of reconstructing deconstructed units for maintaining semantic consistency in integrating component EPCs. The theory is used to guide IEPC designers to design semantically consistent IEPC systems. To achieve the above goal, this paper discusses the theory in the following way: section 2 deconstructs complex heterogeneous component EPCs by articulation. Section 3 reconstructs the deconstructed units in a novel CONEX framework. The final section concludes the paper.

2. The Deconstruction of Component EPCs

This section deconstructs complex heterogeneous component EPCs by articulation, which is borrowed from semiotics ([6]:32; [3]), into atomic units in terms of concepts, structures and semantic relations.

2.1. Articulation of a Component EPC

2.1.1. Representing a component EPC. A component EPC as a whole, created by a business entity, is simply a *computational representation*, which is equivalently a *sign* ([10]:5) and could be applied for the representation of a component EPC, a product, an attribute or even a value. *Signs*, introduced from the semiotic theories [14], takes the form of words, images, sounds, odours, flavours, acts or objects, but such things have no intrinsic meaning and become signs only when people invest them with meanings. In this sense and at this moment, a sign is vague, interweaving complex semantics and syntax in different contexts when we interpret the representation of a catalogue, a product, an attribute or a value as a sign.

In order to articulate a complex component EPC representation, we employ the *dyadic sign model* ([14]:67), which has defined a sign as being composed of a “signifier” and a “signified”, where the *signifier* is the form that the sign takes, and the *signified* is the concept that the sign represents. Applying this model, we refer to the syntactic construct of a component EPC representation as the signifier called *structure* (**ST**), and refer to the semantic construct of the EPC representation as the signified to express the meaning, called *concept* (**C**). By these definitions, any representation (*rep*) in an EPC could be notated and defined as:

$$\text{rep} := (\text{ST}, \text{C})$$

where there exist a *causal relationship* “ \rightarrow ” such that $\text{rep}:\text{C}\rightarrow\text{ST}$. The causal relationship reflects that a structure of a representation is not a proxy for the representation but a vehicle for the meaning delivery of the representation. By this definition, a component EPC can then be modelled as a collection of pairs of structures and concepts, notated as:

$$\text{EPC} := \sum \text{rep}_i = \sum (\text{ST}_i, \text{C}_i), i \in (1 \dots n).$$

2.1.2. Articulating a component EPC into a hierarchy. We build relationship between a structure (signifier) and a concept (signified) by adopting “denotation” and “connotation” ([1]:89-94; [5]:54-57). Following Barthes’ *orders of signification* ([2]:114-115), we call the EPC concept as a whole as denotation (signified) and EPC structure as a whole as connotation (signifier). Then the EPC concept (signified) bonded to the EPC structure (signifier) is a sign (representation), called the *first system* ([1]:89), which again to become a signifier (structure) of a second level system and to be signified (concept). The second level system could be recursively developed as a signifier (structure) of a sign (representation) to be signified (con-

cept) in conformity with Barthes’ statement that the denotation leads to a chain of connotations. The complexly chained relationship suggests two types of signifieds, *denotative concept* and *connotative concept* [3], and their corresponding structures are both recursive.

2.1.3. Capturing recursive concepts. Definition of “rep” is expanded to include denotation and connotation to capture the concept meanings that recursively exist:

$$\text{rep} := \langle \text{DS}, \text{CS} \rangle, \text{C}$$

where $(\text{DS}, \text{CS}) = \text{ST}$, *DS* is “denotative concept structure” and *CS* is “connotative concept structure” that might be null. *C* causally determines (DS, CS) , expressed as “ $\text{C}\rightarrow(\text{DS}, \text{CS})$ ”. The relation between *DS* and *CS* is a *connotation relation*, expressed as “ $\text{DS} \uparrow \text{CS}$ ”, where *CS* connotes *DS* and hierarchically follows their parent *DS* to construct a representation classification system. Based on this, a 4-tuple structure $\langle \text{identifier}, \text{annotation}, \text{link}, \text{connotation structure} \rangle$ is defined to capture the denotative concept:

$$\text{DS} := \langle \text{IID}, \text{AN}, \text{LK}, \text{CS} \rangle$$

where the *DS* comprises four meaningful elements, which has *denotation causality* relation, expressed as $(\text{AN}\rightarrow\text{IID})@\text{LK}$. The @ refers to “located at”. The *CS* is the second level *connotative concept structure* that connotes the first level denotative concept structure *DS*.

$$\text{CS} := \langle \text{DS}(\text{C}_1) \dots \text{DS}(\text{C}_n) \rangle$$

where $\text{DS}(\text{C}_i)$ is a set of conveyed recursive concepts that specify the characteristics of the higher level parent denotative concept. Since the *IID* is the unique identifier of a concept *C* and causally determined by *AN* in each representation, *CS* can then be rewritten as:

$$\text{CS} := \langle \text{DS}(\text{IID}_1) \dots \text{DS}(\text{IID}_n) \rangle.$$

2.1.4. Reification of Meta representation. There are two types of representations: meta-representation and reification representation. The former is a type of genre and the latter is a reification of the former.

Reification Relation (“#”): Given a meta-representation *mRep* and a reified representation *rRep*, then “*mRep* # *rRep*”, called “*mRep* is reified by *rRep*”, if for $m\text{CS} \in m\text{Rep}$ and $r\text{CS} \in r\text{Rep}$, $m\text{CS} = \emptyset$ and $r\text{CS} = \emptyset$.

This definition specifies that in a recursively developed EPC meta-representation, only can a leaf denotative concept structure be reified as a reification representation that has no connotation.

2.2. Articulation of Component EPC Relations

Heterogeneous component EPCs can be articulated against the relations of *syntagm* ([14]:122-123), *paradigm* [3] and *pragmatism* ([13]:180-223) or pragmatics [4].

Syntagmatic relationship specifies the semantic inconsistencies in relation to two different syntactic structures (**ST**) for a concept in two component EPCs. The basic syntagmatic issues are *term combination* and can be

generalised as *structure consistency issue* that leads to product concept classification heterogeneity. The essence is that the recursive concept connotation structures are differently constructed and/or the denotative concept structures are differently constructed, which leads to incoherent semantic effects.

Paradigmatic relationship specifies the semantic inconsistencies in relation to two different denotations for a concept in two component EPCs. The basic paradigmatic issues are *synonymous* and *homonymous concept expressions*, which can be generalized as *concept consistency issue* that makes difficult in *term substitution*.

Pragmatic relationship specifies the inconsistent “relationships between signs and their users” [4]. Specifically, it describes how a component EPC designer or user *interprets* and *acts on* the inbound representations from other EPCs according to his/her stored knowledge in his/her semantic reference systems at the moment of interoperation. The basic pragmatic problem for EPCs is the *interpretation difference of concepts* and could be generalized as the issue of *context consistency issue*. This issue arises from inconsistent context referencing systems that shares no common context.

3. The Reconstruction of Component EPCs

This section reconstructs semantically consistent IEPC systems through a novel CONEX framework for integrating component EPCs based on the deconstructed units.

3.1. CONEX Framework

A CONEX framework is six-tuple $F = \langle C, L, S, \varphi, \Theta, \theta \rangle$ where

- C , the *common context*, is a labelled multi-set $\{comCat_1 := C_1, \dots, comCat_n := C_n\}$. The label $comCat_i$ is the name of a common context (i.e. a common component EPC), and C_i consists of a set of *common concepts* [8] such that $C_i := \sum ce_{ij} (j \in 1 \dots m)$, where each ce_{ij} comprises two parts of an *exterior meta-concept structure* ε_{ij} and an *interior meta-concept structure* δ_{ij} , which is not reified.

The δ_{ij} conveys complex semantics of a concept that is only accessible to internal concept designers of component EPCs. The δ_{ij} is simplified as ε_{ij} in terms of *IID* of a concept to hide the complexity of concept semantics in δ_{ij} . By this means, the meaning of a concept is uniquely expressed and encapsulated in an atomic *IID*, which is qualified for concept exchange between other *IIDs*.

- L , the *local context*, is a labelled multi-set $\{locCat_1 := L_1, \dots, locCat_n := L_n\}$. The label $locCat_i$ is the name of a local context (i.e. a local component EPC), and L_i consists of a set of *local concepts* [8] such that $L_i := \sum le_{ij} (j \in 1 \dots m)$, where each le_{ij} comprises two parts of an exterior meta-concept structure ε_{ij} and an

interior meta-concept structure δ_{ij} , which is not reified.

For any L_i , it is a sub context of at least one common context C_i such that $L_i \subseteq C_i$.

- S , the *source context*, is a labelled multi-set $\{srcCat_1 := S_1, \dots, srcCat_n := S_n\}$. The label $srcCat_i$ is the name of a source context (i.e. a source component EPC), and S_i consists of a set of source concepts such that $S_i := \sum se_{ij} (j \in 1 \dots m)$, where each se_{ij} is reified if it is constrained by a leaf *classifier* factor.

A *classifier* is defined as a pattern of concept classification. A *classifier factor* refers to the position of a concept in a classification based on classifier pattern. For example, C has the classifier of $comClassifier[concept(concept*)]$.

- φ , the *common-to-common context mapping*, defines a bijective replication function φ from C to C . If $\varphi(comCat_i) = comCat_j$, we say $comCat_i$ is replicated onto $comCat_j$, where for each pair of common concepts, their *iids* $\subseteq IID$ are made equal and a translation function $AN_2(iid_2) = \tau(AN_1(iid_1), LANG_1, LANG_2)$ translates their annotations *ANs* from one to another.

For example, for concept $(IID, AN) = (1.52.14.15.1, \text{domestic refrigerators})$, it will be automatically translated into $(1.52.14.15.1, \text{réfrigérateurs domestiques})$ because they have the same common *IID* in mapping process. It should be noted that a verification to ensure translation correctness is needed for common concept designers.

- Θ , the *local-common context map*, is a labelled multi-set $S = \{lcCtx_1 := S_1, \dots, lcCtx_n := S_n\}$. The label is the name of the context map, and S_i consists of a set of maps $map(locLid_i, comLid_j)$ where for $\forall locLid_i \in locCat$ and $\forall comLid_j \in CX \subseteq comCat$, $locCat$ is a heterogeneous sub-context of $comCat$.

The function of Θ is to exchange concepts between C and L . For example, for $1.52.14.15.1 \leftarrow (1.52.14.15.1, \text{domestic refrigerator})$ and $LF111 \leftarrow (LF111, \text{fridge})$, a $map(1.52.14.15.1, LF111)$ will correctly convey concept between “domestic refrigerator” and “fridge”.

- θ , the *source-local context mapping*, defines two mapping functions θ_1 from S to L or θ_2 from L to S . If θ_1 , then there is an implicit-to-explicit mapping function $\omega(locC, comC, comC_1, \dots, comC_n)$ to transform implicit concept to explicit concept. If θ_2 , then there is an explicit to implicit mapping function $\omega(comC, comC_1, \dots, comC_n, locC)$ to transform explicit concepts to implicit concept. For transformation θ_2 , there is a set of value factor functions $valFactor$ defined on

S for public use of all S_i to modify the incoming values against source definitions.

For example, given a hierarchical local concept:

```
(LF111-2, price)
  (LF111-2.1, currency)
  (LF111-2.2, value)
  (LF111-2.3, unit)
```

and a source concept “prc” for price (with implicit context of currency “Euro” and price unit “piece”), then the map will be $map[(LF111-2, LF111-2.1, LF111-2.2, LF111-2.3), prc]$. If there is an incoming document from C to L in the form of:

```
LF111-2, AUD : LF111-2.1, 600 : LF111-2.2,
piece : LF111-2.3,
```

then it is a explicit-to-implicit transformation θ_2 and will involve the calculation of value factors to modify the incoming value. The transformation will be:

```
[(AUD $\rightarrow\emptyset$ ) : LF111.2.1, (piece $\rightarrow\emptyset$ ) : LF111-2.3,
(600 $\rightarrow$ 600*currency(Euro, AUD)*unit(piece, piece)) :
LF111-2.2]  $\rightarrow$  reification : prc.
```

The reification of source concept “prc” will be: 400 : prc.

3.2. Encapsulation of Contextual Semantics

An important feature of CONEX is the insulation of context related semantics from concept exchange, which means that a product concept causally developed in a denotative concept structure is encapsulated and only exposes its concept identifier to the public for concept exchange. The encapsulation feature that hides complexity is given by the *concept externalization mechanism* that includes interior concept structure and exterior concept structure, shown as in Figure 1:

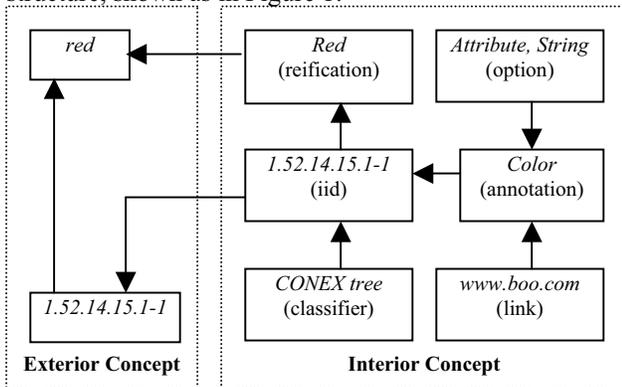


Figure 1: Concept externalization mechanism

By externalizing a complex interior concept into a simple exterior concept, mapping mechanism on CONEX framework becomes simple. The simplicity of concept has also derived another important feature, i.e. *exactness of exchanged concepts* between different contexts. This is because exterior concepts are unique and unambiguous.

4. Conclusion

This paper has investigated the semantic inconsistency problem in the semantic interoperability of product data. It has cope with the complexity in maintaining semantic consistency of heterogeneous EPCs and their relationships by deconstructing and reconstructing of IEPC systems. It has contributed a theory of deconstruction and reconstruction. The contribution includes an articulation approach that deconstructs the heterogeneous component EPCs into units in terms of *concepts* and *structures* and semantic relations. It also includes a novel IEPC reconstruction model in terms of CONEX framework, which provides semantic interoperability between heterogeneous component EPCs through a set of maps.

5. References

- [1] Barthes, R., *Elements of Semiology*, Hill and Wang, 1968.
- [2] Barthes, R., *Mythologies*, Hill and Wang, 1972.
- [3] Chandler, D., *Semiotics for Beginners*, <http://www.aber.ac.uk/media/documents/S4B/semiotic.html>.
- [4] Dijk, T., *Text and Context: Explorations in the Semantics and Pragmatics of Discourse*, Longman, 1977.
- [5] Eco, U., *A Theory of Semiotics*, Indiana Uni. Press, 1975.
- [6] Guiraud, P., *Semiology*, English version, translated by George Gross, Routledge & Kegan Paul Ltd., 1975.
- [7] Guo, J. and C. Sun, “Concept Exchange: Constructing Interoperable Electronic Product Catalogues in an Emergent Environment”, in *CEC 2003: Proceedings of IEEE International Conference on E-Commerce*, IEEE Computer Society, 2003, pp. 165-172.
- [8] Guo, J. and C. Sun, “Context Representation, Transformation and Comparison for Ad hoc Product Data Exchange”, in *DocEng’03: Proceedings of the 2003 ACM Symposium on Document Engineering*, ACM Press, 2003, pp. 121-130.
- [9] Guo, J. and C. Sun, “Global Electronic Markets and Traditional Electronic Markets”, *Electronic Markets*, Vol. 14, No. 1, Routledge, 2004.
- [10] Innis, R. (ed), *Semiotics: An Introductory Anthology*, Hutchinson, 1985.
- [11] Kashyap, V. and A. Sheth, “Semantic and Schematic Similarities between Database Objects: A Context-Based Approach”, *The VLDB Journal* 5, 1996, pp. 276-304.
- [12] Kim, W. and J. Seo, “Classifying Schematic and Data Heterogeneity in Multidatabase Systems”, *IEEE Computer*, Vol. 24, No. 12, 1991, pp. 12-18.
- [13] Peirce, C., *Charles S. Peirce Selected Writings - Values in a Universe of Chance*, Philip P. Wiener, ed, Dover Publications, Inc., New York, 1966.
- [14] Saussure, F., *Course in General Linguistics*, McGraw-Hill Book Company, 1966.
- [15] Veijalainen, J. and R. Popescu-Zeletin, “MAP: An Open Multidatabase System Architecture”, in: *Proceedings of the 3rd Workshop on ACM SIGOPS European Workshop: Autonomy or Interdependence in Distributed Systems*, Cambridge, UK, September 1988.