Collaboratively Maintaining Semantic Consistency of Heterogeneous Concepts towards a Common Concept Set

Jingzhi Guo, Iok Ham Lam, Chun Chan and Guangyi Xiao
Department of Computer and Information Science, University of Macau
Av. Padre Tomás, Pereira, S.J., Taipa, Macau
{jzguo, ma46521, ma46507, ya97409}@umac.mo

ABSTRACT
In e-business, creating a common concept set for business integration, interoperation and interaction has to consider the heterogeneity reality of different interpretations from multiple concept providers. Maintaining semantic consistency between multiple concept providers is a difficult problem. To solve this problem, this paper first reviewed the existing technologies of collaborative editing systems and consistency maintenance in the areas of both CSCW and e-business. Based on the discussion of existing technologies, it then proposes a novel CHCES approach, which divides a collaborative editing system into two layers in topology and introduces four strategies to edit common concepts between the two layers. A set of operations is designed, which demonstrates the solution.

Author Keywords
Semantic consistency, collaborative editing, concept

ACM Classification Keywords
H.5 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *Collaborative computing; Web-based interaction*; K.4.4 [**Computing Milieux**]: Computers and Society – *Electronic Commerce.*

General Terms
Algorithms, Design, Human Factors

1. INTRODUCTION
In e-marketplace, business concepts such as the documents of inquiry, offer, acceptance, contract, invoice, draft, and bill of lading must be accurately exchanged between business partners without misinterpretation. Misinterpretation will lead to legal consequences. For example, receiving a message of "orange refrigerator, price at 200" from a US company, the computer agent of a Japanese company could misinterpret it as "easy-to-carry small sized cooler bags often used for camping and keeping fruits like oranges in low temperature, and its price is 200 Japanese Yens per piece", as compared with the original meaning of "household refrigerators normally used in kitchens to keep foods in low temperature, its color is orange, and its price is USD200 per piece". When this misinterpretation happens in e-marketplace systems, business orders could be wrongly placed and executed, and then legal disputes could occur between two parties.

This problem occurs when the heterogeneous e-business concepts are produced from unknown buyers and sellers who situate in their own "semantic communities" [19] but want to do e-business together. Doing e-business together is a way of collaboratively working in a common information space (CIS) [20], where properties of interdependence, distribution, autonomy and emergence have to be satisfied [12]. This type of collaboration further asks for the semantic consistency maintenance for accurate concept exchange, such as the accurate meaning understanding of any incoming terms, documents and operations that may be used for further e-business communication.

Consistency maintenance has long been researched in collaborative editing systems of Computer Supported Cooperative Work (CSCW) area and is divided into syntactic consistency maintenance and semantic consistency maintenance. Traditionally in CSCW, the former often refers to achieving consistency of viewed copies, causal relations and operation effects between same applications in collaboration (e.g. REDUCE [22] and CoMaya [1]). The latter, in general, means to ensure the application semantics when applications in collaboration are not identical, for example, applications of different versions [8]. This type of consistency aims to provide a set of consistent notions or artifacts, so that another collaborative toolkit (or collaborative engine) as an infrastructure for applications can be consistently developed to support the collaborative work between application users. Today, as the rapid development of Internet-based e-business, sellers and buyers require working together for doing business in order to maximize their sales revenue, even if they never meet with each other before. This practice presents two very challenging research requirements: how to make contextually different sellers and buyers to work together, and how to ensure that both sellers and buyers share the consistent meaning understanding of information that is in exchange, so that they can make unambiguous and consistent deals. Obviously, the first requirement asks for a cross-context collaborative toolkit where both sellers and buyers can consistently edit their communication messages for business. This surely requires studying both syntactic and semantic consistency maintenance. The second requirement, somehow, steps out of the existing CSCW research, which demands to maintain semantic consistency between heterogeneous concepts in information exchange when sellers and buyers edit their business messages. This paper is motivated by the second requirement to complement the existing CSCW research.

In literature, existing researches on maintaining semantic consistency of concepts between heterogeneous concept systems can be summarized into three categories of approaches [13].

EICS'10, June 19-23, 2010, Berlin, Germany.

(A1) $\{C_i, C_j \mid C_i, C_j \subseteq S\}$: all heterogeneous concept sets (C) for exchange shall conform to a standard concept set (S), for example, UNSPSC.org, ebXML.org, or x12.org.
(A2) $\{C_i, C_j \mid C_i \cup C_j = M\}$: all heterogeneous concept sets (C) for exchange shall be bridged ($\cup$) by a secondary mediating concept set (M), for example, ontology design [6, 15, 18] or mediator design [25, 27].
(A3) $\{C_i, C_j \mid C_i \cap C_j = \Gamma\}$: all heterogeneous concept sets (C) for exchange shall be mutually agreed ($\cap$) on meanings at a collaborative concept set (Γ) [13]. For example, "fridge" of C1 and "refrigerator" of C2 can be exchanged if and only if they are mutually agreed in meaning by collaboration.

It is obvious that Approach A1 is not workable because no one can guarantee that C_i, C_j of unknown parties are all converted to S. Approach A2 guarantees a unified access to C_i, C_j but cannot guarantee $C_i =_{sem} C_j$ (C_i, C_j are semantically equal). Approach A3 assures $C_i =_{sem} C_j$ through a collaboration mechanism Γ where parties make agreements on semantic equivalence for any heterogeneous concepts.

This paper aims to solve the problem using Approach A3 by proposing a novel and general collaboration mechanism, called CHCES and pronounced as [ʃes], where semantic consistency of heterogeneous concepts is maintained by collaborative editing. The CHCES design has the following requirements: (1) *distribution*: individual concept editing systems shall be distributed on multiple Internet locations; (2) *personalization*: editing user interfaces and editing concept sets of individual editing systems shall be maintained differently with each other; (3) *semantic consistency*: a same concept in different personalized (or heterogeneous) forms shall be explicitly and mutually agreed on its meaning in CHCES environment.

The rest of the paper will be arranged as follows: Section 2 discusses the related and prior work of CHCES system. Section 3 describes CHCES system framework. In Section 4, semantic consistency maintenance of heterogeneous concepts is depicted, followed by a conclusion.

2. PRIOR AND RELATED WORK

2.1. Related Work

In CSCW literature, collaborative editing systems are groupware that allows multiple users to view and edit the same concept, text, graphic, image, and multimedia document at the same time from multiple sites connected by communication networks. Traditionally, such systems focus the research on the satisfying the characteristics of real-time or non-real-time, distributed or centrally-managed, and constrained or unconstrained. *Real-time* is that the response to local user actions is quick and the latency for reflecting remote user actions is low. *Distributed* is that collaborating users may reside on different machines connected by different communication networks with non-deterministic latency. *Unconstrained* is that multiple users are allowed to concurrently and freely edit any part of the document at any time, in order to facilitate free and natural information flow among multiple users [22]. By contrast, non-real-time, centrally-managed and constrained have the relatively opposite meanings of real-time, distributed and constrained. To satisfy some of these characteristics for particular applications, different approaches are developed and can be categorized in Table 1.

Table 1: Types of Collaborative Editing Systems

Type	Characteristics	Research examples
1	Non-real-time, centrally managed, constrained	CVS [11]
2	Real-time, centrally-managed, constrained	Lantz [16], Begole et al [2] , C/Webtop [4]
3	Real-time and non-real-time, centrally-managed, constrained	ActivityExplorer [26], Miramar [14]
4	Real-time, centrally-managed, unconstrained	MOODS [5]
5	Real-time, distributed, constrained	Flexible JAMM [3]
6	Real-time, distributed, unconstrained	GROVE [10], REDUCE [22], GRACE [23], CoWord [24], CoMaya [1]
7	Real-time and non-real-time, distributed, unconstrained	Shen and Sun [21] to extend Type [4].

For the listed types of collaborative editing systems, a core research issue is to maintain consistency between operation results by collaborative users from their situated applications (either homogeneous or heterogeneous). The difficulty of consistency maintenance comes from the two facts of both syntactic inconsistency of spatial, structural, and temporal [8], and semantic inconsistency including non-shared application semantics (e.g. meanings of terms, commands, and their compositions) when applications used by users are coded in different artifacts. Syntactic consistency problems can often be resolved by various solutions of locking, serialization, operational transformation [22] and multi-versioning [23]. While there are many solutions to syntactic consistency, the research on semantic consistency for collaborative editing is not well explored. The early understanding of why semantic consistency is needed is the lack of knowledge of application semantics when designing collaborative toolkits above the applications to support users to work together on various either single or multiple applications. For example, in the Prospero of Dourish [7], semantic consistency is the data store consistency from the perspective of the application domain linking to the designed collaborative toolkits. The management of semantic consistency of data store is independent of syntactic consistency of the collaborative toolkits. Similarly, Edwards' Timewarp [8] considers semantic consistency as a collection of shared artifacts (particularly a set of shared actions) that enables to build an infrastructure as a collaborative toolkit below the applications, permitting divergent views from application users. These two examples thus design shared semantics of terms or action (i.e. operation) naming for sharing between collaborative toolkits, which can further resolve syntactic inconsistency. This is a correct thinking. Nevertheless, it is still an intention of sharing application semantics and is also not well elaborated. Modern understanding of semantic consistency maintenance is far beyond the application domain. For example, ontology design attempts to share concepts within an industry domain across particular application contexts, where shared ontology as metadata can leverage many different applications (*see* Approach A2 in Introduction). Standardization is another similar approach, which supports many applications (*see* Approach A1 in Introduction) if and only if all involved applications adopt the same standards of shared terms. These two approaches are now, somehow, noticed by some researchers in CSCW area (e.g. Intermezzo [9]). However, the ontology modeling for cross-context collaborative editing does not imply unproblematic for semantic consistency maintenance. When editors are situated in heterogeneous contexts, their understanding on terms presents synonym and homonym problems (e.g. given two actions "quote" and "offer", they may either mean the same or different). Thus, the terms themselves for supporting to design collaborative editing systems need the collabora-

tive design again. What's more, in e-commerce application design, the users even require the consistent meaning understanding of terms in communication but not merely for application semantics. For example, when user A sends a refrigerator order to user B, they need to make sure whether "refrigerator" meaning is consistent to their common understanding. If they share different understanding, legal disputes may arise. This has motivated the latest design of collaborative conceptualization approach (*see* Approach A3 in Introduction [13]), which is the theoretical foundation of this paper for CHCES design.

2.2. Prior Work

CHCES is a collaborative editing system for achieving semantic consistency. Although it is related to collaborative editing systems and their consistency maintenance technology, it is thoroughly based on the prior research of Collaborative Concept Exchange (CONEX) [13], where heterogeneous product concepts can be semantically exchanged accurately if and only if the following mapped path exists for concept equivalence:

Source concept 1 ↔ map(Source concept 1, Local concept 1) ↔ Local concept 1 ↔ map(Local concept 1, Common concept 1) ↔ Common concept 1 ↔ map(Common concept 1, Common concept 2) ↔ Local concept 2 ↔ map(Local concept 2, Source concept 2) ↔ Source concept 2.

It is obvious that Source concept 1 is semantically equivalent to Source concept 2 if all maps exist and the mapped concepts are mutually agreed in meaning equivalence.

In [13], concept equivalence is ensured in two types of collaborators: common-common collaborator to build the mappings between two common concepts, and local-common collaborator to build mappings between one local concept and one common concept. These collaborators are made in an XPM concept representation specification [13] on a semantic consistency maintenance model, where three properties of structure mappability, concept equivalence and context commonality must be satisfied. CONEX approach has the following advantages:

- The concept equivalence relationship is built on the basis of heterogeneous concept forms. It thus meets the requirements of personalization and allows legacy concepts to be mapped. For example, the heterogeneous forms of "refrigerator", "fridge", "电冰箱" and "réfrigérateur" is semantically equivalent if and only if they are mutually agreed in meaning equivalence for all concept designers.
- The separation of concept (i.e. concept meaning) from its structure (i.e. concept form or concept syntax) by introducing the unique identifier for each concept, which creates a relationship "concept annotation (AN) → concept identifier (IID)" such that an *iid* can represent all heterogeneous forms of concepts if and only if their *iids* are the same or their *iids* are semantically mapped by collaborative agreements, for example, "refrigerator→1101", "fridge→d358", "电冰箱→1101" and "réfrigérateur→1101" and *map*(1101, d358). This separation simplifies the way of semantic consistency maintenance between heterogeneous concepts.

In designing common-common collaborator, [13] introduces a node locking procedure to ensure that in any time there is only one single pair between a unique IID and an equivalent concept in any forms. It also discusses a translation procedure that allows different natural languages for a same concept to be automatically translated and semantically verified. The work of [13] has met the requirements of distribution, personalization and semantic consistency. However, it is only restrictedly applicable to the field of electronic product catalogues. A dedicated discussion for its generalization is needed for editing any concepts. In addition, node locking procedure affects concurrent operations on a same concept while machine translation procedure has introduced unpredictable semantic consistency problem in translation for concept annotation that defines a common concept.

3. CHCES FRAMEWORK

The meaning consistency for information exchange, but not only for consistent application semantics, requires a fully novel solution. To achieve this goal, this Section describes a CHCES framework where common concepts across contexts are collaboratively edited and their semantic consistency is maintained through the system design without node locking and machine translation.

3.1. CHCES Topology

Definition 1: CHCES is topologized as a tuple (G, E, ⇔, ↔), where G is a *global editing system* maintained by an e-marketplace provider (EMP), E are *common editing systems* maintained by multiple common concept providers (CCP), "⇔" is a WAN connection such that G⇔E, and "↔" is a LAN connection such that $\{g_i \leftrightarrow g_j \mid g_i, g_j \in G\}$. □

CHCES topology can be conceptualized in Figure 1.

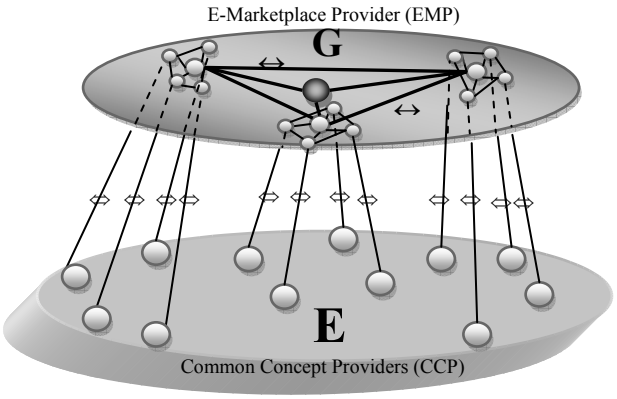

Figure 1: CHCES Topology

By Definition 1, a new business model is suggested as follows: an *e-marketplace provider* (EMP) provides the common concept editing service to *common concept providers* (CCP) and each CCP provides common concept mapping service to firms, allowing them to map local concepts onto common concepts.

3.2. E-System

Definition 2: A common editing system E is a tuple (MEX, OPT, CED, TCDB, PCDB, EDM), where MEX (*Message Exchanger*) is to receive and send XPM-based real-time operation messages or buffered operation messages. OPT (*Operation Transformer*) is to read remote (incoming) XPM operation message from MEX and execute the operations on a TCDB (*Temporary Concept Database*). It also reads the local (outgoing) operations on CED and writes them as XPM messages for MEX to propagate to G. The TCDB is a temporary database that records all concepts currently in editing. CED (Common Editor) is a user interface for common concept editing, which displays all temporary concepts of TCDB and indexing concepts of PCDB (*Permanent Concept Database*) and perceives user's editing and arbitration operations on these concepts. PCDB is a permanent database of all finalized common

concepts provided to all connected firms for use. EDM (*Editor Manager*) is to store and edit user information of the editor and responsible for the logon and logoff of G system. □

An E-System can be illustrated in Figure 2, in which OPT and CED are very important in dynamic editing.

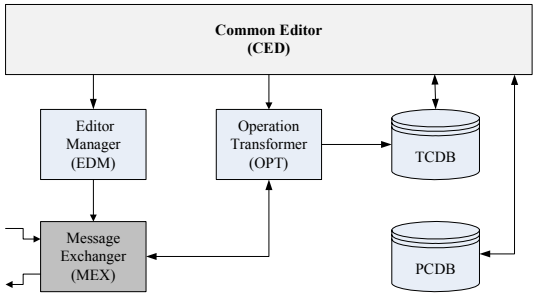

Figure 2: E-System Model

Definition 3: Operation Transformer (OPT) is a tuple (XOP, XOS, OEE), where XOP (XPM Operation Parser) parses and validates both incoming and outgoing editing operation message on XOS (XPM Operation Schema). OEE (Operation Event Executor) interprets remote operations from incoming messages and executes them on TCDB, or reversely transforms local operations into outgoing XPM-based operation messages for MEX. □

Definition 4: Common Editor (CED) is a tuple (TDP, PDP, DOE, AOE), where TDP is TCDB Display, PDP is PCDB Display, DOE is designer's operation executor, AOE is arbitrator's operation executor. For both DOE and AOE, operations are executed both locally in TCDB for DOE or in PCDB for AOE and remotely in G system where the operations are received from MEX of E systems in XPM operation messages via OPT. □

3.3. G-System

Definition 5: A global editing system G is a tuple (MEX, GOT, ECC, UIID, GCS, RCS, PHB), where MEX is message exchanger same as in E-System. GOT is the global operation transformer between MEX and GCS/RCS. ECC is existential concept checker to find out whether a concept intending to add is already in GCS/RCS. UIID is the unique concept identifier generator for adding modifying a concept. GCS is a global concept set for English. RCS is set of regional concept sets in natural languages. PHB is a set of personal history buffer of each connected E-System. It is used to store all the consecutive operations propagated to all E-System. □

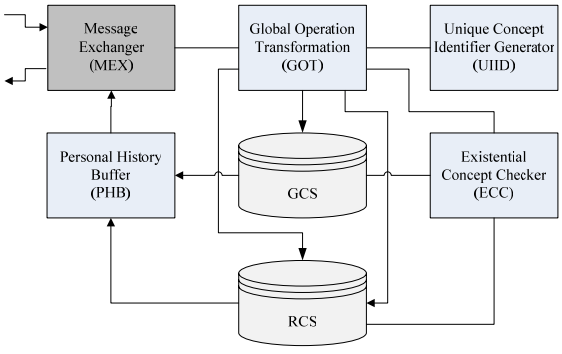

Figure 3: G-System Model

A G-System can be illustrated in Figure 3, where a PHB is a list of historical operations in G-system, which are run by any common editors of E-system and executed on G-system. It consists of all operations starting from the offline time of any common editors of E-System. The PHB will be automatically emptied when the longest offline common editors of E-System go online.

4. SEMANTIC CONSISTENCY MAINTENANCE

We define $E = \{E_1, E_2, \ldots, E_m\}$, where $E_1 = \{e_{11}, e_{12}, \ldots, e_{1m}\}$, $E_2 = \{e_{21}, e_{22}, \ldots, e_{2m}\}$, …, $E_m = \{e_{m1}, e_{m2}, \ldots, e_{mm}\}$ such that each E_i is in same natural language. Subsequently, for TCDB and PCDB in E, we have:

(1) $TCDB=T=\{T_1, T_2, \ldots, T_m\}$, where $T_1 = \{t_{11}, t_{12}, \ldots, t_{1m}\}$, $T_2 = \{t_{21}, t_{22}, \ldots, t_{2m}\}$, …, $T_m = \{t_{m1}, t_{m2}, \ldots, t_{mm}\}$.
(2) $PCDB=P=\{P_1, P_2, \ldots, P_m\}$, where $P_1=\{p_{11}, p_{12}, \ldots, p_{1m}\}$, $P_2=\{p_{21}, p_{22}, \ldots, p_{2m}\}$, …, $P_m=\{p_{m1}, p_{m2}, \ldots, p_{mm}\}$.

Our task is to maintain semantic consistency between any two p_{ij} and p_{pq} and between any two t_{ij} and t_{pq}, such that any meaning changes in p_{ij} and t_{ij} will also exactly be changed in p_{pq} and t_{pq}, respectively. To maintain this semantic consistency, we adopt four strategies:

(1) In any time, only a unique common concept identifier (IID) is assigned to any semantically same concepts. By requiring a central generation of IID, node locking mechanism is avoided.
(2) Bilingual common concept design is required and English is suggested as global concept (GC) language and non-English as common concept (CC) languages. By this strategy, machine translation is avoided.
(3) PHB is used for all E-systems and automatically executed when E-System goes online. By this strategy, asynchronous collaboration is supported.
(4) Arbitrator decides the final acceptable concepts. By this strategy, multiple copies of concept modification in a defined period are eliminated.

Steps of maintaining semantic consistency are as follows:

Step 1: Define syntax of TCDB, PCDB, GCS and RCS

(1) TCDB = T = <(IIDG, FCG, ANG, FCX, …), (IIDC, FCC, ANC, CTX), …>;
(2) PCDB = P = <(IIDG, FCG, ANG, FCX, …), (IIDC, FCC, ANC, CTX, …>;
(3) GCS = <IIDG, FCG, ANG, FCX, status, …>;
(4) RCS = <IIDC, FCC, ANC, CTX, status, …> ;

where IIDG is in form of ot-ct_1 (ot for *originalTimestamp* to create new concept, ct_1 for *currentTimestamp1* to modify ANG). FCG is English word. ANG is English annotation defining FCG. FCX is the parent concept of IIDG. IIDC is ot-ct_1-ct_2 (ct_2 for modify non-English ANC). FCC is non-English word. ANC is non-English annotation defining FCC. CTX is the parent concept of IIDC. The "status" denotes whether a concept is in added, deleted, edited and arbitrated.

For example, in a CCP's common editor of E system, an added term "orange" before arbitration will be stored in TCDB as <(ot111-ct111, "orange", "orange color", ot101-ct101, added), (ot111-ct111, "橙", "桔黃色", ot101-ct101, added)>. After arbitration, the new term will be deleted in TCDB and moved to PCDB as <(ot111-ct111, "orange", "orange color", ot101-ct101, arbitrated), (ot111-ct111, "橙", "桔 黃 色", ot101-ct101, arbitrated)>. Similarly, in G system, the part of global English standard concept (ot111-ct111, "orange", "orange color", ot101-ct101, arbitrated) will be placed in GCS while the common regional con-

cept (ot111-ct111, "橙", "桔黃色", ot101-ct101, arbitrated) will be placed in RCS.

Step 2: Maintain semantic consistency between heterogeneous concept sets

1. Add a new English global concept *gc* and add a corresponding non-English common concept *cc*.

```
(1)  If adding gc to TCDB Then
(2)    {If (FCG(gc) ∉ GCS Then   // GCS OEE check
(3)      {x = UIID(IIDG);  // create a new IIDG
(4)       IIDG(gc) = x; FCG(x) = gc; // assign to gc
(5)       ANG(x) = " ";  // human annotation.
(6)       // find the contexts of FCX and CTX for gc if any
(7)       // Write (iidg, fcg, ang, fcx, ctx) to TCDB;
(8)       // Propagate (iidg, fcg, ang, fcx, ctx) to GCS;
(9)       // Write Add(iidg, fcg, ang, fcx, ctx) to PHB;
(10)      If (FCC(cc) ∉ RCS And x ∈ GCS Then   // RCS OEE check
(11)        {y = UIID(IIDC(IIDG(x)));  // create a new IIDC of x
(12)         IIDC(cc) = y; FCC(y) = cc;
(13)         ANC(cc) = " "; // waiting for human input.
(14)         // Write (iidc, fcc, anc) to TCDB;
(15)         // Propagate (iidc, fcc) to RCS;
(16)         // Write Add(iidc, fcc) to PHB;}
(17)      Else // Go to modify cc;}
(18)   Else // Go to modify gc;}
```

This guarantees that each newly created concept is semantically unique for both real-time and asynchronous editing.

For example, when adding an "orange" global concept paired with regional common concept "橙", we first check whether FCG = "orange" is in GCS. If it is not, we generate IIDG = ot111-ct111 as a global concept identifier and let ot111-ct111 represents FCG = "orange" with the meaning ANG = "orange color". Likewise, we have IIDC = ot111-ct111, FCC = "橙", and ANC = "橙色" with the status = "added". When these have been done in common editor, it is written to TCDB of local E system and propagated to RCS and PHB of G system, where this ADD operation stored in PHB is immediately executed by other common editors of E-system and emptied. If there are some common editors in E system are offline, this ADD operation is not emptied until all finish the execution of this ADD operation.

2. Modify an English global concept *gc* and modify a non-English common concept *cc*.

```
(1)  If (FCG(gc) ∈ GCS Then
(2)    {x = UIID(IIDG(gc))   // Get IIDG of gc
(3)     y = UIID(x) // new gc IIDG in same ot but diff. ct1
(4)     IIDG(gc) = y; FCG(y) = gc; ANG(y) = " ";
(5)     // Do as 1:(7)-(9) for TCDB, GCS, PHB;}
(6)  If (FCC(cc) ∈ RCS Then
(7)    {x = UIID(IIDC(cc))   // Get IIDC of cc
(8)     y=UIID(x) // new cc iidc in same ot, ct1 but diff. ct2
(9)     IIDC(cc) = y; FCC(y) = cc; ANC(y) = " ";
(10)    // Do as 1:(14)-(16) for TCDB, RCS, PHB;}
```

This guarantees that all concept modification only happens on concept annotations respectively for ANG and ANC.

For example, if FCG = "orange" has already been in GCS, there are two possibilities: one is that the editor wants to modify ANG of "orange color", for example, to "color with the hue of that portion of the visible spectrum lying between red and yellow", or the editor wants to add a new meaning of "orange", for example, ANG = "a kind of fruit having a sweetish and acidic juice". For the latter case, a new concept identifier as a homonym of ot111-ct111 must be created, such as IIDG = ot211-ct211, following ADD algorithm again.

3. Delete an English global concept *gc* or a non-English common concept *cc*.

```
(1)  Select ANG(gc) from TCDB;
(2)  Read x = IIDG(ANG(gc));
(3)  Delete gc(x, FCG(x), ANG(x)) in TCDB;
(4)  // Find the children gcc of gc under FCX(gcc) = gc;
(5)  FCX(gcc) = FCX (gc); // formal context of gc becomes that of gcc
(6)  // Propagate Delete(); Multiple deletions on same gc are neglected;
(7)  Select (ANC(cc) from TCDB;
(8)  Read x = IIDC(ANC(cc));
(9)  Delete cc(x, FCC(x), ANC(x)) in PCDB;
(10) // Propagate Delete();Multiple deletions on RCS are neglected;
```

This guarantees no side effect will have when a concept is deleted. The formal context (FCX) of a deleted concept becomes the children's formal context. The normal context (CTX) is only upward and has no effect on any other concepts when a concept is deleted concept.

4. Make arbitration on any concepts in TCDB.

```
(1)  Select t ∈ TCDB;
(2)  Move (t, PCDB);
(3)  // Propagate Move(t, PCDB) to GCS and RCS
(4)  // Write Move(t, PCDB) to PHB;
```

This guarantees that only a best modification copy from multiple semantically-same yet annotation-heterogeneous concepts is selected to PCDB in every arbitration time.

For example, for a given ADD("orange") operation of both ANG = "orange color" and ANG = "color with the hue of that portion of the visible spectrum lying between red and yellow", arbitrator has the right to select only ANG = "orange color" as final annotation.

5. CONCLUSION

Achieving a consistent meaningful understanding in communication between collaborators is extremely important in collaborative applications, e-business transactions, and many other software systems. To fulfill such goal, this paper has described a general collaboration mechanism, called CHCES, to maintain semantic consistency between multiple concepts by collaborative editing in different natural languages. This approach is designed in two layers of E-System and G-System, in which semantic operations from different common editors of E-System is controlled by remote G-System. Particularly, the semantic consistency is maintained by four strategies of unique concept identifier (solving the problem of redundant concept creation and modification), bilingual editing (solving machine translation problem), personal history buffer (solving asynchronous offline problem), and arbitration (solving multiple copies of concept creation and modification).

This paper has contributed a new understanding of how to use collaborative editing technology to maintain consistency in common concept creation and modification from multiple concept domains. This contribution is novel to Computer Supported Cooperative Work (CSCW) area and is a natural extension of the discussion from syntactic consistency maintenance to semantic consistency maintenance, which, traditionally, is only limited to providing solutions of shared application semantics. As collaboration is more and more needed to support a common information space where ubiquitous computing applications are presented and mutual understanding of the exchanged information are required, the research topic of this paper appears urgent and worth being paid higher attention.

With the rapid development of Internet, we believe that the work of this paper is not only a continuation of the existing collaborative editing research but also very useful for emergent areas of e-

business, Semantic Web and Web 3.0. In future, its implementation work will be presented for a better illustration of this new technology.

ACKNOWLEDGEMENT

This paper is partially supported by University of Macau Research Grant No. RG055/08-09S/GJZ/FST.

REFERENCES

1. Agustina, Liu, F., Xia, S., Shen, H. and C. Sun (2008) Co-Maya: Incorporating Advanced Collaboration Capabilities into 3D Digital Media Design Tools. In: Proc. of ACM CSCW'08 (Nov. 8–12, 2008, San Diego, California, USA), 5-8.
2. Begole, J., Struble, C.A., and C.A. Shaffer (1997) Leveraging Java applets: Towards collaboration transparency in Java. IEEE Internet Compuing 1(2):57–64.
3. Begole, J., Rosson, M. B. and C. A. Shaffer (1999) Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems. Transactions on Computer-Human Interaction 6(2):95-132.
4. Bergenti, F., Poggi, A. and M. Somacher (2002) A collaborative platform for fixed and mobile networks. Communications of the ACM 45(11):39-44.
5. Bellini, P., Nesi, P. and M.B.Spinu (2002) Cooperative Visual Manipulation of Music Notation. ACM Transactions on Computer-Human Interaction 9(3):194–237.
6. Brickley, D,, Guha, R. V. and B. McBride (2004) RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004, www.w3.org/TR/rdf-schema.
7. Dourish, P. (1996) Consistency guarantees: Exploiting application semantics for consistency management in a collaboration toolkit. In Proc. of the ACM CSCW'96, 268–277.
8. Edwards, W. K. (1997) Flexible conflict detection and management in collaborative applications. In Proc. of ACM UIST'97, 139–148.
9. Edwards, W. K. (2005) Putting computing in context: An infrastructure to support extensible context-enhanced collaborative applications. Transactions on Computer-Human Interaction 12(4):446–474.
10. Ellis, C. A. and S. J. Gibbs (1989) Concurrency control in groupware systems. In Proc. of the ACM SIGMOD Conference on Management of Data, 399–407.
11. Grune, D. (1986) Concurrent version system, a method for independent cooperation. Report IR-114, Vrije University, Amsterdam.
12. Guo, J. (2007) A Term in Search of the Infrastructure of Electronic Markets. IFIP Volume 255: 831-840.
13. Guo, J. (2008) Collaborative Concept Exchange, VDM Verlag, Germany.
14. Hancock, s. M. Miller, J. D., Greenberg, S. and S. Carpendale (2006) Exploring visual feedback of change conflict in a distributed 3D environment. In: ACM Proc. of the working conference on Advanced visual interfaces, 209-216.
15. Klyne, G., Carroll, J. and B. McBride (eds) (2004) Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004. [online] http://www.w3.org/TR/rdf-concepts/.
16. Lantz, K. (1986) An experiment in integrated multimedia conferencing. In Proc. of ACM CSCW'86, 267-275.
17. Li, D. and R. Li (2004) Preserving operation effects relation in group editors. In: Proc. of ACM CSCW'04, 457-466.
18. McGuinness, D. and F. Harmelen (2004) OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004. [online] http://www.w3.org/TR/owl-features/.
19. Robinson, M. and L. Bannon, Questioning Representations, in: Proc. ECSCW'91 (Amsterdam, September 1991), 219-233.
20. Schmidt, K. and L. Bannon (1992) Taking CSCW Seriously: Supporting Articulation Work. Computer Supported Cooperative Work 1(1): 7-40.
21. Shen, H. and C. Sun (2002) Flexible Notification for Collaborative Systems. In: Proc. of CSCW'02, 77-86.
22. Sun, C., Jia, X., Zhang, Y., Yang, Y. and D. Chen (1998) Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. ACM Transactions on Computer-Human Interaction 5(1): 63-108.
23. Sun, C.and D. Chen (2002) Consistency maintenance in real-time collaborative graphics editing systems. ACM Transactions on Computer-Human Interaction 9(1):1-44.
24. Sun, C., Xia, S., Sun, D., Chen, D., Shen, H. and W. Cai (2006) Transparent adaptation of single-user applications for multi-user real-time collaboration. ACM Transactions on Computer-Human Interaction 13(4): 531-582.
25. Tzitzikas, Y., Spyratos, N. and P. Constantopoulos (2005) Mediators over taxonomy-based information sources. VLDB Journal 14:112–136.
26. Vogel, J, Geyer, W., Cheng, L-T and M. Muller (2004) Consistency Control for Synchronous and Asynchronous Collaboration Based on Shared Objects and Activities. Computer Supported Cooperative Work 13(5-6):573-602.
27. Wiederhold G (1992) Mediators in the architecture of future information systems. IEEE Computer 25: 38–49.