FAULT-TOLERANT EMERGENT SEMANTICS IN P2P NETWORKS

Abdul-Rahman Mawlood-Yunis[1], Michael Weiss[2] and Nicola Santoro[1]
[1] School of Computer Science, Carleton University
[2] Department of System and Computer Engineering, Carleton University
1125 Colonel By Drive, Ottawa, Ontario, K1S5B6, Canada
{armyunis,santoro}@scs.carleton.ca, weiss@sce.carleton.ca

ABSTRACT

To survive in the twenty-first century, enterprises need to collaborate. Collaboration at the enterprise-level presupposes the interoperability of the underlying information systems. Access to heterogeneous information sources must be provided transparently while maintaining their autonomy. Further, the availability of nearly unlimited information calls for efficient and precise information retrieval, which can be achieved by making the semantics embedded in information sources explicit. Solving the semantic interoperability problem becomes imperative to the success of information search and retrieval applications and enterprises that rely on them.

Inspired by self-organizing systems found in biology, physics and computing, the approach of *emergent semantics* has been proposed as a solution to the semantic interoperability problem. Emergent semantics refers to the bottom-up construction of interoperable systems, in which semantically related peers are discovered and linked together during the normal operation of the system. Individual information source providers will provide mappings (so-called semantic bridges) between their own local and semantically related foreign information sources. Emergent Semantics in a peer-to-peer (P2P) network is the lowest common knowledge, semantically relevant concepts, among all the peers of the network.

Local mappings between peers with different knowledge representations, and their correctness are prerequisite for the creation of emergent semantics. Yet, often approaches to emergent semantics fail to distinguish between permanent and transient mapping faults. This may result in erroneously labeling peers as having incompatible knowledge representations. In turn, this can further prevent such peers from interacting with other semantically related peers[1]. This is because, in emergent semantics, peers use past interactions to determine which peers they will interact with in future collaborations.

This chapter will explore the issue of semantic mapping faults. This issue has not received enough attention in the literature. Specifically, it will focus on the effect of non-permanent semantic mapping faults on both inclusiveness of semantic emergence and robustness of applications and systems that use semantic mappings. A *fault-tolerant* emergent semantics algorithm with the ability to resist transient semantic mapping faults is also provided. The contributions of this chapter are: (i) an analysis of the impact of the semantic mapping faults on the inclusiveness of semantic knowledge sharing in P2P

[1] Emergent Semantic depends on the adaptive query routing algorithm. Checking for semantically related peers are embedded in the method. There is little chance for *unrelated peers to interact*.

systems, (ii) a preliminary solution to the problems created by semantic mapping faults in P2P semantic knowledge sharing systems, and (iii) a qualitative analysis of the causal links between fault causes and fault types.

The rest of this chapter is organized as follows. Section 2 provides broad discussion and literature review about semantic interoperability problem among heterogeneous information source. Section 3 defines what we mean by a semantic mapping fault and the types of faults. Section 4 lists sources of semantic mapping faults. Section 5 classifies temporal semantic mapping faults. Section 6 describes the emergent semantics approach. Section 7 presents an algorithm to eliminate the harmful effects of transient mapping faults on emergent semantics (fault-tolerant emergent semantics). Section 8 concludes the chapter and Section 9 identifies directions for future work.

KEYWORDS
Ontologies, Emergent Semantics, Semantic Interoperability, Data sharing, Distributed Systems, Semantic Matching, Query Processing, Fault-tolerance

BACKGROUND

In today's globally connected and digitalized world, the ability to exchange information, provide services and carry out business worldwide has become an essential requirement for many government agencies and departments, interest groups, businesses, etc. The need for transparent exchange of information and doing business on the global scale is faced with the *semantic heterogeneous information representation* problem among autonomous and distributed information source providers.

Existing information sources are *scattered* around the world. They are stored in repositories located in different government departments, research labs, universities, interest groups, enterprises, etc. The stored information is represented heterogeneously along different aspects. For example, data or information can be in XML files[2], relational tables, HTML files, RDF[3] documents etc. Further, when the same type of representation format is used for storing information, the information modeling, the structure and semantics of concepts used in the modeling may vary among different information source providers.

An example of *semantic differences* would be using *different* vocabularies to refer to the same physical or conceptual object by different information representations: one's “zip code” is somebody else's “area code”; or using the *same* vocabulary to refer to *different* conceptual or physical real life objects in different representations: a “terminal” for one is a computer monitor, but a “station” for somebody else.

In the distributed environment, information source providers are autonomous. In other words, information source providers have control on their local information sources. They could make changes, update, remove or restrict the access to their information

[2] http://www.w3.org/XML/
[3] http://www.w3.org/RDF/

sources. Consequently, in order for various businesses and service applications and systems to be able to cooperate and exchange information in the environment described above, they need to overcome the barrier of heterogeneity between semantic information representations.

In the sections below, we will delineate how a common ontology and emergent semantics help resolve the issue of semantic heterogeneity, and review existing literature on the different approaches for solving the problem.

***Ontology-enabled* Semantic Reconciliation**

Common ontologies and shared semantics (Gruber, 1993) have been used for semantic reconciliation, recognizing similarities and enabling information exchange to overcome the representational differences. Knowledge engineers and domain experts use concepts from common ontologies to model the area of interest (e.g. medicine, education, tourism) where concept meanings are shared and agreed upon by members of the domain, i.e. individuals commit to the meanings assigned to vocabularies used to describe the domain.

To enable information exchange among multiple independent ontologies for the same domain or among ontologies from overlapping domains, an upper ontology is utilized as mediator. Concepts from independent ontologies are mapped to the common ontologies and from common ontologies to the other independent ontologies. This procedure continues back and forth and for as much as needed.

Several global ontologies have been constructed including OpenCyc, SUO/SUMO, UNPSC etc (Gomez-Perez, 2004). Despite some usefulness of this approach and the existing number of common upper ontologies, the prominent problems with this type of work are the maintenance and scale up difficulties as ontology domain concepts change or evolve over time. It is hard to have an ontology which is comprehensive and highly agreed upon. Thus, to date, there is no privileged or standard common ontology in use for any domain.

More recently, contextualization, or use of local ontologies, has been suggested by some authors (Bonifacio, 2002 ; Bouquet, 2003; Ghidini) as a strategy for modeling information sources. Following this paradigm, individual information source providers (be they Web site owners, operators of peers in a semantic P2P network, or database designers) will annotate their information sources with semantics in their own ontologies. These semantics will be provider-specific, and reflect the information provider's knowledge of the application domain, experience, or culture. This implies a shift from large and centralized to smaller and possibly simpler distributed ontologies.

However, contextualization also imposes new restrictions. Allowing users to create their own local data representations and semantics raises heterogeneous representation problem, e.g. problem of semantic incompatibility among the interacting information sources. To resolve the heterogeneity problem (i.e. enable independent and autonomous information sources to communicate with one another) we need to provide semantic mappings, i.e. translations between semantically related peers.

Local Translation and Emergent Semantics

Emergent behavior is a well-known phenomenon in biology, physics and (distributed) computing. For example, several optimization and network routing techniques have been inspired by the way the behavior of an ant colony as a whole emerges from local interactions between individual ants.[4] Similarly, local cooperation between robots in multi-robot systems for search and rescue operations has been modeled after the formation of flocks of birds (Bahceci, 2003).

Inspired by emergent behavior, the approach of *emergent semantics* has been proposed as a solution to the semantic interoperability problem among autonomous, heterogeneous information sources with local ontologies. Emergent semantics refers to the bottom-up construction of interoperable systems, in which semantically related peers are discovered and linked together during normal operation of the system --- as part of regular search and query forwarding operations. Under this approach, individual information source providers provide semantic mappings (so-called *semantic bridges*) between their own local and semantically related foreign information sources (Aberer, 2003, 2004; Larry, 2006; Staab, 2002). This implies a shift from large and centralized approach to a decentralized approach with smaller ontologies. Bottom-up construction of emergent semantic enables consensus reaching on the semantics of concepts used in distributed local ontologies. This in turn paves the way for the knowledge sharing among independent and autonomous peers. Emergent Semantics in a P2P network is the *lowest common knowledge* among all peers' contextual ontologies in the network.

The decentralized approach, not only puts the scalability problem behind, but also if used with simpler ontologies -- ontologies with less expressive power and less restricted language -- mainly taxonomy, causes dramatic change in the scale of semantic Web applications and semantic information exchange in P2P applications. This is because simplicity encourages users to annotate their information sources with semantics (Rousset, 2004), to understand and make use of others' ontologies.

The decentralized semantic reconciliation approach is especially attractive for semantic search and query forwarding in peer-to-peer (P2P) network (Staab, 2006). This is not only because the information peers bring to the network is heterogeneous and their meanings need to be reconciled in order to improve the search and query results, but also because P2P network is dynamic and the decentralized approach performs dynamic semantic mapping.

Using dynamic semantic mapping, concepts that constitute the query are the only ones which need to be translated and it is done on the fly, i.e. during system operation. This

[4] http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html

approach suits the P2P dynamic network well and is much preferred over the pre-defined mappings of all concepts among semantically connected peers.

Local mappings between peers with different knowledge representations, and their correctness are a prerequisite for the creation of emergent semantics. Yet, often approaches to emergent semantics fail to distinguish between permanent and *transient* mapping faults. This may result in the erroneous labeling of peers as having incompatible knowledge representations. In turn, this can further prevent such peers from teaming up with other semantically related peers in the future. This is because, in emergent semantics, peers use past interactions to determine which peers they will interact with in future collaborations.

The importance of tolerating non-permanent faults (also known as noise) has long been recognized in hardware and software reliability studies. Non-permanent faults include transient, but also intermittent faults (which are recurring transient faults; for definitions of these terms see Section 3). Methods for controlling the effects of non-permanent faults form an important part of disciplines such as fault-tolerance (Bondavalli, 1997, 2000; Pizza, 1998) and evolutionary game theory (see e.g. Axelrod, 97; Wu, 1995 for a discussion of noise in the iterated prisoner's dilemma).

We argue that Web information systems must also tolerate non-permanent faults. This is particularly true for mission-critical applications such as security and business-to-business applications. Discarding a viable source of information, or preventing a valuable business partner from participating in business transactions just because of transient faults will negatively impact the level of accuracy of the collected information in the security case, and could jeopardize potential financial gains in business-to-business applications.

Existing Approaches

We observed from the literature review that approaches to solve semantic interoperability problem are somewhat different from each other. The existing works could be roughly classified into to four different inter-related classes: Local Mapping and Query Translation, Collaboratively Building Ontologies and Consensus Reaching, Pattern Extraction or Structure Similarity and Tagging and Social Networks. The names are related to the way each approach tries to reconcile the semantic differences among different information source representations. Below is a short a description of each approach.

Local Mapping and Query Translation

The underlying working environment for this approach is mostly a P2P network and a common theme among systems belongs to this approach is the use of local mapping to achieve some form of knowledge sharing and cooperation. In other words, peers have their own local representations and local mappings, i.e. translations between local information presentations are provided to enable information exchange among communicating peers. Examples of systems that use this approach include Chatty Web (Aberer, 2003), OBSERVER (Mena, 2000), Piazza (Halevy, 2003), H-Match (Castano, 2003), KEx (Bonifacio, 2002), Bibsiter (Haase, 2004) and SomeWhere (Rousset, 2006). For a short survey about these systems, the reader is encouraged to see (Mawlood, 2007).

Collaborative Building of Ontologies and Consensus Reaching

An engineering methodology for building ontology collaboratively and reaching consensus on concept definition and domain conceptualization has been suggested by Tempich (2004, 2005). The procedure starts by building general core ontology then, individual users extend the core ontology and adapt it to their local needs. After using the core ontology, users are asked to send feedback to a centralized authority regarding what should and should not be part of the core ontology. The centralized authority will look after user's suggestions and updates the core ontology accordingly. Authors of the methodology assert that after several iterations, a stable and shared common ontology will emerge.

Pattern Extraction or Structure Similarity

Distributed Emergence System (DistES) (Fergus, 2003) and Constructing consensus ontologies for semantic web (Stephens, 2001) are examples of the systems which use structure similarity among distributed ontologies to solve the interoperability problem. DistES protocol is based on the evolutionary algorithm for discovering and merging knowledge in P2P environment. Each peer owns local ontology represented in hierarchical structure. Peers extend their knowledge by querying other peers, selecting best result among the query answers and merging the selected result with local ontology. The process of selecting foreign concepts and forging concept relations for integration with local data is based on their frequency of occurrence in the query answers. Concepts and concept relation with high occurrence, i.e., appeared in multiple query answers; will be selected for merging with local data, those with fewer occurrences are ignored. Thus, the end result information source structure, i.e. emerging ontology, manifests the general consensus among peers who participated in the interaction. Similarly, Stephens (2001) uses the occurrence rate of concepts and concept relations among multiple, small and related ontologies used for web annotation to construct a merged ontology on the fly. The newly constructed ontology is then presented to the user for further refining.

Tagging and Social Networks

The launch of the social book marking Web site "del.icio.us" [5], the photo sharing service "Flickr"[6] and others opened-up a new way of categorizing Web information sources, i.e. building ontologies collaboratively by large numbers of Web users.

[5] http://del.icio.us/

A network of English words made of *numerous tags* used by independent users for labeling the *same online document* forms the basis for ontology creation by this strategy. Similarly, using *same tag* by numerous independent users to refer to some *resource* is the basis for creation of online communities around using common resources, i.e. share common interest (Mika, 2005). Currently, serious discussion and interest have been devoted to social networking and collaborative building ontologies in academia. Several works following this strategy have been surveyed in Staab (2005).

We will extend or build upon these existing techniques by eliminating/ reducing the effect of the temporal mapping faults which confronts the semantic information flow. In our system we will try to overcome two fundamental problems of the existing systems: the lack of fault tolerance and the inability to distinguish permanent from non-permanent semantic mapping faults. The ability to resist semantic mapping fault helps in building a robust system. It also prevents peer's unwarranted removal from future participation on further collaboration events. This implies an intelligent use of peer's past collaboration to determine future decision on further collaboration in a best possible way.

SEMANTIC MAPPING FAULTS

In this section we define what we mean by a *semantic mapping fault*, and identify different types of faults based on notions from fault tolerance literature.

Faults

A *fault* is an incorrect semantic mapping, or the failure to map between concepts from different ontologies. We say that a fault occurs when (i) a concept in one ontology is mapped to a semantically unrelated concept in a different ontology, or (ii) a concept in one ontology cannot be mapped to an existing semantically related concept in a different ontology.

Formally we can express this definition as follows. Assume we have two ontologies O1 = $\{C, P, R\}$ and O2 = $\{C', P', R'\}$ where C and C' are sets of concepts, P and P' are sets of concept properties and R and R' are sets of relations between concepts. Given two semantically equivalent concepts or their instances[7] $c \in C$ and $c' \in C'$ such that $c \equiv c'$ we say that a fault occurs if either one of the following is true:

- c is mapped to a semantically unrelated concept $x \in C'$ such that $x \neq c$.
- c cannot be mapped to a semantically related concept $c' \in C$, i.e. the mapping process incorrectly leads to nil.

---

[6] http://www.flickr.com

[7] For information on instance data and schemas, we refer interested readers to (http://jena.sourceforge.net/ontology/common-problems.html#aBox-tBox; http://www.w3.org/TR/owl-guide/).

The fault-tolerance literature classifies faults based on their duration. Accordingly, we distinguish between permanent, transient and intermittent faults.

Permanent Faults
A *permanent fault* is a fault that continues to exist, unless some outside action takes place to remove its underlying cause.

For example, any attempt to map between two concepts from two unrelated ontologies, i.e., two ontologies from different domains, will result in a permanent fault. This situation will continue indefinitely, unless, e.g., a change is made in the mapping semantics linking the ontologies.

Transient Faults
A *transient fault* is a type of fault that appears once, and remains in place for a short period of the time.

A transient fault may corrupt the data of a system, but the system will remain operational. It is a statistical fault, and it is hard to predict when exactly it will happen. For example, the change of a company's stock symbol can result in a transient semantic mapping fault, if either the propagation of the change notification to related peers or applications is delayed, or the related peers or applications are unable to incorporate the change immediately.

Intermittent Faults
An *intermittent fault* is a fault that occurs periodically. It appears for a short period of time, disappears, and then reappears repeatedly.

For example, in a situation where ontology modification is not a full substitution of one ontology by another, it is possible for semantically related peers to continue operating. In the described scenario, there can be intermittent faults. Faults will occur, because there are situations, when related peers are unable to interpret the meaning of concepts in the modified ontologies.

Although transient and intermittent faults manifest very similarly, they are quite different. While the first one is generated from temporal condition, the second one is the result of unstable system. Also, the intermittent fault could be fixed by removing the unstable component from the system, but transient faults cannot be eliminated.

The diagram in Figure 1 is one way that the three types of errors can be visualized. A semantic mapping can either be correct (no fault) or incorrect (faulty). In the case of a permanent fault, once the status of the mapping changes from no fault to faulty, it remains faulty. For a transient fault, the mapping will be faulty for some time interval. In the case of an intermittent fault, the status of the mapping repeatedly changes between no fault and faulty.

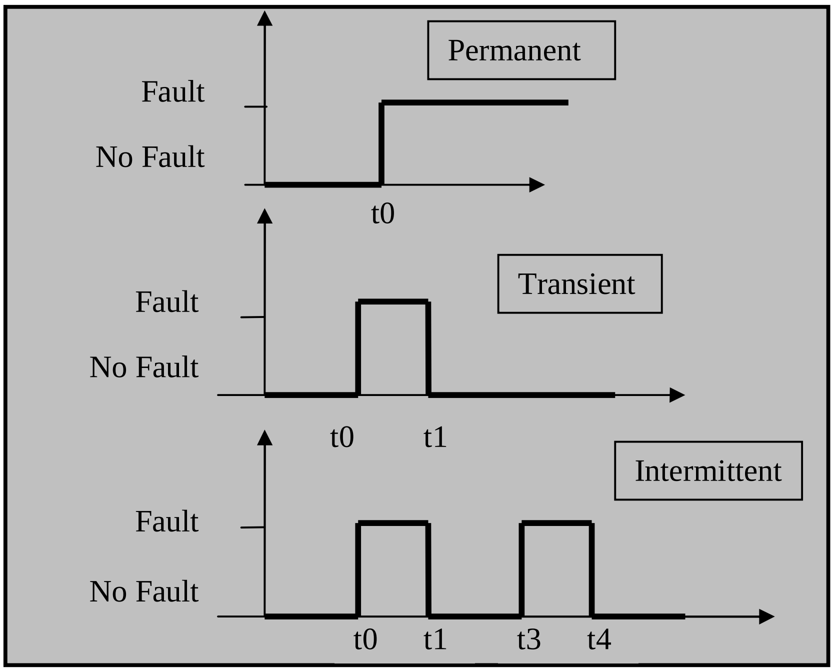

Figure 1 Fault types

4. CAUSES OF NON-PERMANENT SEMANTIC MAPPING FAULTS

In this section, we discuss situations that can cause non-permanent (transient or intermittent) semantic mapping faults. Our intention is not to be comprehensive, but to illustrate the need for handling (either by tolerating or guarding against) temporary mapping faults. The causes of faults discussed below include ontology evolution, query context and static mapping, temporal nature of data, unavailability of data sources, and misbehavior of peers.

Ontology Evolution

Ontologies evolve as existing components/elements are replaced with new components, or components are modified. The evolution of software and its consequences on system functionality has received much attention in the software engineering and database design communities (Roddick, 1995). Observations about software evolution can also be applied to the evolution of ontologies. Noy (2004a) has studied the effect of ontology evolution on Web applications and concludes that it strongly impacts system operability and the interpretation of data [8].

[8] (Noy, 2004b) argues that the issue of versioning and evolving are same in the context of ontology mapping. What we see as important is that both versioning and evolving introduce modifications to the existing ontology.

There are several scenarios in which different types of semantic mapping faults could occur as a result of ontology evolution. For instance, they could occur as a result of either one or all of the following factors:

- *Adding new concepts* to an existing ontology, e.g. adding a newly discovered class or type of drug to existing relevant ontologies.
- *Deleting concepts* from an existing ontology. Outdated concepts or concepts that are no longer used or useful may be deleted from the ontology.
- *Changes in concept meaning*. A changed meaning can result in the removal or addition of a concept relation or property.

An example of a change in ontology by adding new properties is attaching a concept for hydrogen as a new type of fuel to the concept car. Removing a relation that links the concept floppy drive to the concept PC is another example of a change in ontology, e.g. for a PC maker who no longer supports floppy drives in its product configurations

Query Contexts and Static Mapping

Static mapping is a mapping without consideration of the *context* in which a concept is used, i.e. the relations and properties of a concept. It is a *term-by-term* association. For example, if a concept x is mapped statically to another concept y; mapping will always produce the same results no matter the context of x.
Static mapping may generate faulty answers to queries when used in different contexts. This can explained by way of the following example.

Take two concepts (shown in Figs. 2 and 3) from different ontologies that represent information about *Students* at a *University* and *Members* of a *Research Institute*. *Assume* the following relations between the two ontologies:

- Some *Members* of a *Research Institute* are *Students* of a *University,* and the *Employer* concept represents this relation.
- The relationship of the *Research Institute* to the *Institute* and the relationship of *Educational Institute* to *Institute* from the two ontologies were as depicted in Figure 4.

One can see that the *University* concept from the first ontology and the *Research Institute* concept from the second ontology become semantically equivalent, i.e.

$$\text{University} \equiv \text{Research Institute.}$$

This is possible, because the *Institute* concept from both ontologies can be declared in the mapping table as equivalent concepts.

Consider the effect of the static mapping from *Institute* to *Institute* on the following two queries:

Q_1: List the *Names* of all *Members* of *Institutes*

When this query is posed on both ontologies, it asserts that University $\equiv$Institute. However, consider the second query:

Q_2 List the *Names* of all Members of *Educational Institutes*

The relation University $\equiv$ Institute no longer holds, and its assumption will result in a fault.
That is, while the semantic correspondance between the concepts resulted in a correct answer to the first query, it resulted in fault for the second query. This scenario is a good example of an *intermittent* fault. Every time the static mapping between *University* and Research *Institute* is used, a fault will occur, but not if it is not used. The work of (Ouksel, 1999) further elaborates on the effect of context and static mapping on faults.

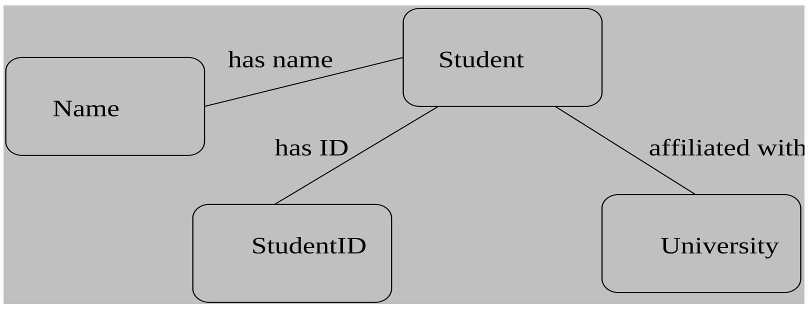

Figure 2: Representation of the concept *Student* at a *University*

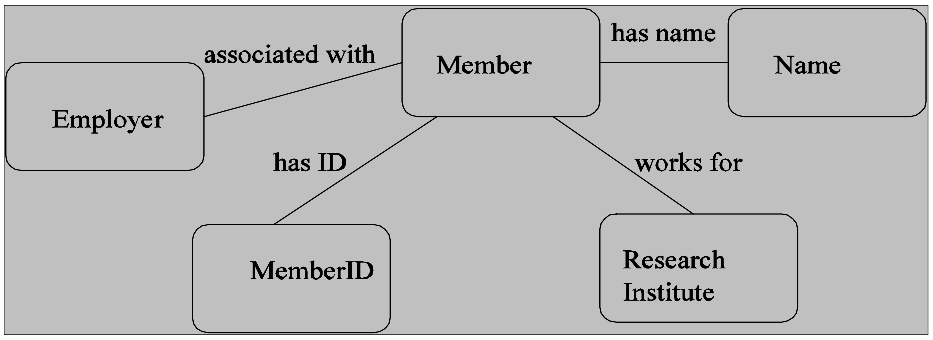

Figure 3: Representation of the concept *Member* of a Research *Institute*

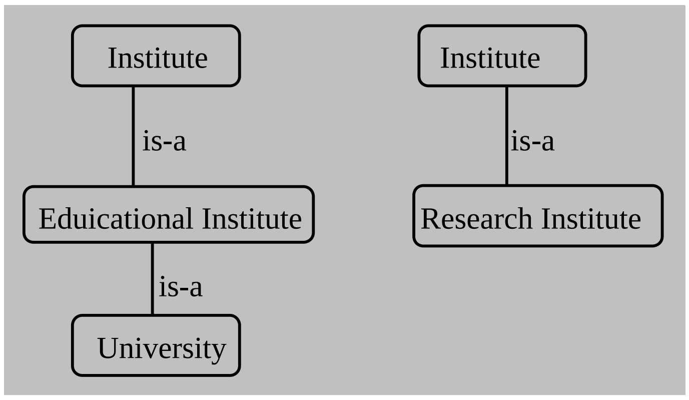

Figure 4: Two Different Institute Concept Representations

Temporal Nature of Data

While in pervious examples we talked about *concept mapping faults*, concept instances could also lead to transient or intermittent faults. Even though different researchers have different views on whether instances should be part of ontology or not (e.g. Lacy, 2005; McGuiness, 2003)[9], an important source of faults during query answer evaluation, not directly in mapping, involves changes over time in the concept instances. This is true whether instances are part of ontology or not

The issue of temporal data is of high importance in situations where data are changing continuously such as stock prices or weather temperature. Both *price* in the *stock* ontology and *temperature* in the *weather* ontology are properties of concepts whose instances change over time. A query answer evaluator that compares temperature values or stock prices represented in two different ontologies may produce different results at different points in time. Not accounting for time dependency can lead to faulty query answer assessments.

Assume that there is a network of peers that provide weather information for different cities, each with weather ontology similar to that shown in Figure 5. Also assume that we want to find the coldest city in the network. One way to achieve this is by running a query similar to the following over all related cities and subsequently compare the results:

Q: Find the *Location* with the lowest *temperature*
If query propagation is delayed for some reason, or queries were posed at different times to each peer, the result will not reflect the correct weather temperature. This fault is not the result of differences in semantic representation (all peers use the same ontology), but rather due to the temporal nature of the temperature concept. This fault could be temporary or permanent, based on whether temporal concepts are accounted for or not in the ontology. Something similar could be said about a query to find the cheapest stock

[9] see these following links as well: http://www.w3.org/TR/owl-guide/, http://jena.sourceforge.net/ontology/common-problems.html#aBox-tBox

price. Other examples related to temporal changes of ontology concepts are presented in (Zhu, 2004).

The temporal issue is not limited to the concept instances. Similar issues also apply to temporal ontologies. However, while the issues relevant to the temporal schemas have extensively been studied in Database, the temporal issue is still an open area of research for ontologies (Gutierrez, 2005, Hurtado, 2006).

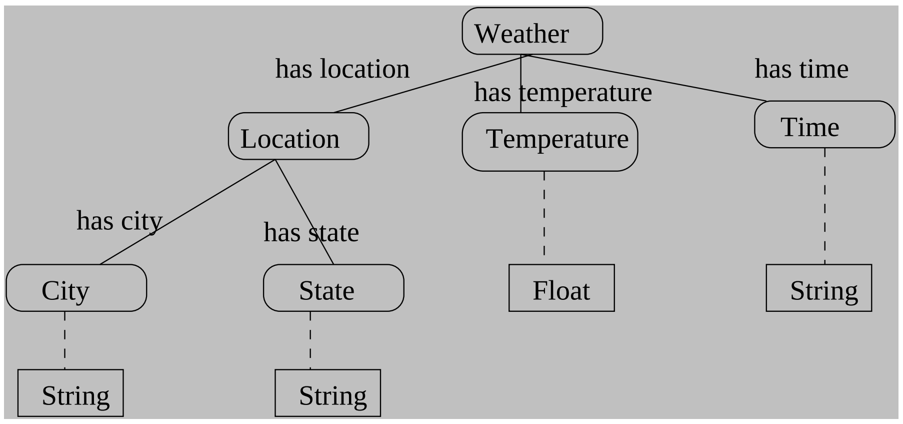

Figure 5: Partial Weather Ontology [10]

Unavailability of Data Sources

It has been pointed out by (Gal, 2001) that the design of the conceptual schema for information services possesses special properties. These include (1) a rapid change of data sources and metadata, and (2) instability, since there is no control over the information sources. The *availability* of information sources is solely dependent upon information source providers. A possible scenario is the temporary unavailability of information when such information is needed. This possibility is particularly acute during query execution.

Misbehavior of Peers

Correctness of semantic mapping depends on the honest conduct of peers. A peer could be dishonest or biased in his interaction with other peers during the mapping process for reasons such as selfishness and greed. There are various ways through which a peer could influence the mapping process. These ways include (1) not forwarding a query to other peers during transitive mapping process, or, (2) not forwarding answers to the other peers during mapping, and, (3) altering or delaying queries (results) before forwarding them to other peers. In all of these situations the mapping will be incorrect.

[10] Note that this weather ontology is a partial ontology with instances

Working in a hostile or uncooperative environment gives rise to situations where peers are permanently hostile or uncooperative. This may lead to permanent faults. However, in the case of unintentional misinterpretation or incorrect implementation of mappings, faults are produced from “noise-like” actions, and it will be correct to assume that they are non-permanent.

There is some similarity between the information source unavailability described in the subsection 4.4 and peers misbehavior, but they are not quite the same. While the former is caused by information unavailability, the latter results in the information unavailability. Thus, we decided to present them separately.

In the above scenarios we need to differentiate between permanent and temporary mapping faults. The knowledge about different types of faults along the temporal dimension will help us determine when peers should be excluded from further interaction. This helps in better consensus formation, which in turn contributes to solving the semantic interoperability problem.

5. CLASSIFICATION OF TEMPORAL MAPPING FAULTS

In this section, we will re-examine the different fault causes that have been listed in previous section to find out under what circumstances each individual fault cause could result in transient, intermittent or permanent faults, and classify them along two dimensions:

- *Type*. As described in Section 3, we distinguish three types of faults: permanent, transient and intermittent faults.
- *Cause*. Fault causes were identified in Section 4. Below, we describe when they occur and identify their associated fault types.

Since we assume that local mappings between ontologies already exist, our classification will focus on what faults may occur during *mapping execution*, rather than on faults that may occur because of errors in the *mapping logic*, e.g. substituting a concept by its hypernyms or hyponyms. Hence, mapping faults caused by meaning and representations of concepts are not included in this classification. For this type   of fault we refer reader to (Naiman, 1995; Ram, 2004; Glushko, 2005). Also, In order to simplify the analysis we sometimes refer to both intermittent and transient type errors as non-permanent faults.

Permanent Mapping Faults

The following situations could result in permanent mapping faults:

- Mapping *temporal* concepts without a representation of time constraints in the ontology leads to permanent faults. This is because temporal ontology concepts are continuously changing with time. Even if the mapping process produces (by accident) some correct mappings without consideration for time constraints, eventually the system will fail completely.

- The *degree* of ontology *modification* (versioning and evolution), and whether or not the modified concepts will be *used* in the mapping process, will determine the mapping result. A high degree of modification and the frequent use of the modified concepts may prevent semantically related applications or peers from working with the modified ontology.
- If the system is *unavailable*, the mapping process cannot be performed. Unavailability may be the result of a network or peer failure.
- Working in a hostile or *uncooperative* environment can create conditions where peers are permanently hostile or uncooperative. [11]

We would like to point out that *query context and static mapping* will less likely lead to permanent faults. If this were not the case, it would indicate that the existing mapping is incomplete. Hence, a better concept mapping would be required.

Non-permanent Mapping Faults

Except from those situations identified in the first case, all other situations will result in non-permanent faults. These situations include:

- A change in *query context* which can give rise to intermittent faults. This is because every time an existing correspondence between two concepts, i.e. an existing static mapping, is used when mapping for contexts other than the contexts for which the relation was defined for, an error may occur.
- A *denial of service* request due to *temporary* server crashes or the disappearance and reappearance of peers which will result in a non-permanent fault.
- *time constraints* represented in temporal ontology concepts: e.g., if a *delay* is experienced during transitive query rewritten, i.e. $q_{\_start} + q_{\_delivary} > d_{\_time}$, where $q_{\_start}$ refers to the time when the query is submitted to other peers, $q_{\_delivary}$ refers to the length of time a query takes to propagate from a peer A to a peer B, and $d_{\_time}$ refers a point in time where the information on the remote site is correct, query result evaluator could falsely concludes that the query result is unavailable or nil. Depending on the frequency of query propagation delays this will lead to intermittent or transient faults.
- Circumstances where *ontology evolution* is not a complete substitution of the previous ontology, it is possible for related peers or applications to continue operating. In this scenario there can be intermittent faults. Faults will occur, because there are situations, where semantically related peers are unable to interpret the meanings of concepts in a modified ontology.

[11] If multiple peers cooperate and misbehave intentionally, this will create a different type of fault known as *Byzantine* fault, which is not considered in this chapter.

Moreover, the ontology modification procedure also has an impact on the fault type. Modification procedure could result in either (1) the unavailability of the information source for a short period of time, if the ontology is locked for updating or (2) a *race* condition between the information source provider and information user, if the ontology user is informed about the change after the modification. That is, the modification problem becomes an instance of the unavailability or temporal problems described above. From this observation we may conclude that every ontology modification can lead to a non-permanent fault.

- Unintentional *misinterpretation* or incorrect implementation of mappings gives rise to an incorrect mapping. Since the faults are produced from "noise-like "actions, it will be correct to assume that they are non-permanent.

The observations about ontology modification, unavailability and temporal ontology concepts can be generalized as follows:

- The effect of an ontology modification is not as severe as the effect of unavailability. This is because we assume that modifications to ontologies are less frequent than an information source becoming unavailable.
- The probability of transient faults may be higher than that for intermittent faults. Again, this is for the same reason.

It is important to note that, in this section, we have looked at causes of faults one cause at a time. For example, we studied the effect of query context, temporal aspects, and ontology modification separately. It will be interesting to explore whether a fault can be the result of multiple causes, and whether we need to distinguish between different fault causes, when a fault occurs. However, the approach that we will take in the next section to detect and remedy faults does not require knowledge of the underlying cause. Table 1 summarizes this classification.

	Transient Fault	**Intermittent Fault**	**Permanent Fault**
Temporal Semantic Conflict	One-time message delay	Frequent message delays	Unsupported time constraint
Versioning and Evolution	During changes	During changes	Unsupported change management
Query Context and Static Mapping	Unsupported Query Context	Unsupported Query Context	Disqualify
Unavailability of Data Sources	Unavailability > Timeout	Frequent unavailability	Unavailability = infinity
Peers Misbehavior	Misbehavior for short period of time	Repeated misbehavior	Permanent misbehavior

Table 1: Classification of Temporal Faults

6. CRITICAL REVIEW OF EMERGENT SEMANTICS

This section starts by describing the steps used by current methods to emergent semantic as documented in the literature (Aberer, 2003, 2004; Larry, 2006; Staab, 2002). These steps (shown in Figure 6) are:

1. Peers join a network after finding the first peer with a compatible knowledge representation. That is, peers establish mappings to the semantically related peers[12]. Subsequently, peers submit queries to their neighboring peers using concepts in their own personalized local ontologies.
2. Upon receiving a query, each peer starts processing the query locally, if the concepts used to formulate the query are compatible with concepts in its local ontology, and sends back query results to the querying peer. Otherwise, it forwards the query to other peers for which they have a direct mapping, after invoking the translation facilitator. Query forwarding will continue, until either (1) the query reaches the query initiator[13], (2) the query exceeds a specified number of query forwards ("hops"), or (3) the time to live for the query message expires.
3. The querying peer (query initiator) collects all answers returned, and *evaluates* them. If the answers were satisfactory, the query initiator will inform the neighbors involved about the result. Thus, the entire translation paths will be informed of the result of a successful query[14].
4. By comparing (mapping) list of query concepts to the list of concepts from the query result, the querying peer could conclude if the semantic relation along the translation path has been preserved. The semantic preservation is used to increase (decrease) peers confidence in their immediately connected peers. For mapping query concepts to the concepts of query answer, semantic affinity between concepts should be defined and used by mapping process. For example, the semantic affinity of $\{\subset, \supset, \equiv, \perp\}$ between two concepts could be defined as {0.5, 0.5, 1, 0} respectively. Similarly, peers could use satisfaction (dissatisfaction) of the query answers to reward (punish) their directly connected peers. The latter case is more appropriate for situation where emergent semantics is not the issue of the concern. Combing both described methods also possible. Query semantic would be considered preserved if (i) key concepts in the query did not drop during semantic mapping chain, and (ii) the average value of the semantic preservation were greater than or equal to some threshold. The value of the threshold could be set by system administrator, where higher value means that higher semantic affinity between concepts of query and query answer is required. Query answers could be considered satisfactory if they meet query constraints.

[12] Peers join the network by crafting their own mappings. Thus, it is reasonable to assume that they start with correct mappings. It is when ontologies change or evolve, mapping faults become a serious issue. Similar assumption is made by (Robert, 2005).

[13] The query must stop here, otherwise an infinite forwarding loop would be possible.

[14] A successful query result implies a successful series of mappings.

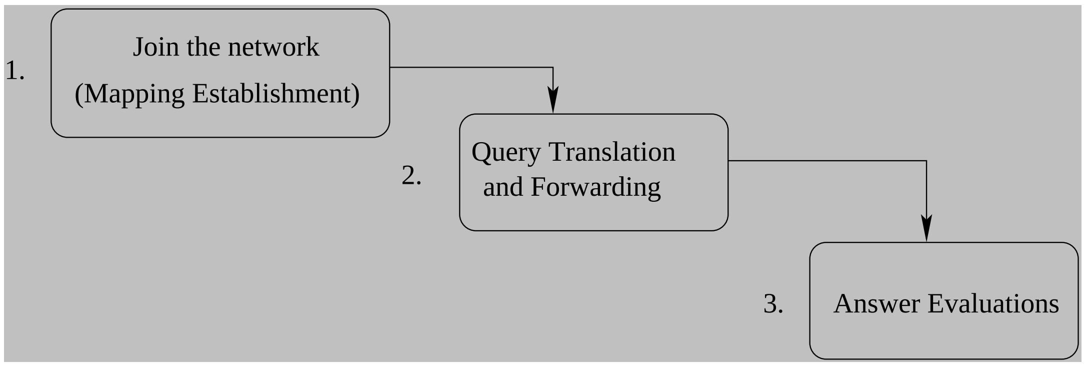

Figure 6 Main Steps of the Current Emergent Semantic Process

The described steps could be conceived as a process of constructing a directed graph, where anytime a local peer P encounters another peer P` that provides a correct answer to its query, i.e. a peer with a comparable semantic representation, the existing semantic mapping between these peers will be further reinforced. That is, semantically related peers are discovered and linked to each other during the normal operation of the system --- search and query forwarding. Figure 7 depicts such a graph. In the Figure, the highlighted peer is the query initiator, labels on the links represent a mapping from source to target and semantically related peers are connected by a link. The graph will be used by peers for future collaboration, e.g. when initiating or forwarding a similar query.

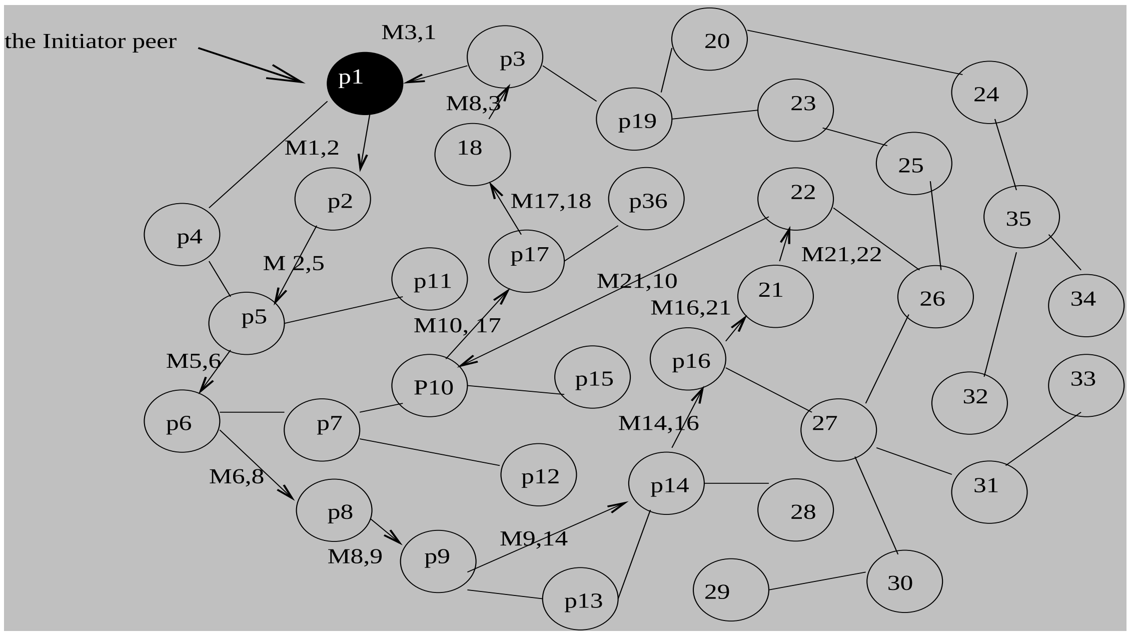

Figure 7 Semantically Related Peers Without Temporal Fault Handling

The graph is a snapshot of a network where peers are connected by semantic mappings. The whole graph is not stored at any peer in the network, it is a distributed graph. Each peer in the network has knowledge through the confidence value of its out-going mapping links to semantically relevant peers. However, peers could store the whole graph locally, if they decided to do so, by integrating foreign concepts into their local knowledge repositories. Example of integrating foreign concepts into local knowledge representation is described in (Castano, 2003).

A fundamental prerequisite for the creation of the described semantic graph is the existence of *local mappings* between peers with different ontologies and the *correctness* of those local mappings. Thus, when peers are unable to answer queries or provide correct answers to them, how this failure is handled can become a source of problems. We need to make a subtle distinction between permanent and non-permanent semantic mapping faults or risk the erroneous labeling of peers as having incompatible knowledge representations.

To give an idea of the consequences of erroneously labeling peers as incompatible we consider the effect on the number of outgoing mapping links each peer has to other peers in the network. We will consider two cases:

Case 1 In this case, one of the peers on the mapping path used to answer the query has only *one* outgoing link. By mapping path we mean the chain of translations used to produce the query result. Figure 8 represents this case, where peer P_1 is the query initiator, peer P_k the peer with one outgoing link M_k, and all links from peer P_{k+1} from different paths participating in query answers returned to the initiator peer P_1. Small circles on the edges of the graph indicate that multiple peers participated in forming the results.

It should be clear from Figure 8 that unless the system can distinguish between transient and permanent mapping faults, if the mapping M_k between Peer P_k and peer P_{k+1} is not successful, even only for a short period of time, peer P_1 will conclude that the outgoing mapping link M_1 is not entirely reliable, that is, its confidence in the outgoing mapping link M_1 will be reduced.

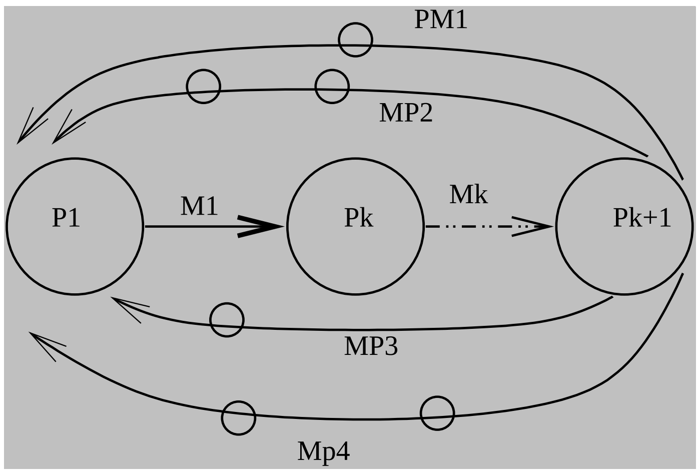

Figure 8 A Peer on the Mapping Path with One Outgoing Link

This is because even a temporary failure of a mapping link, $M_{k,}$ results in incorrectness of all paths MP_1, MP_2, MP_3 and MP_4, following that mapping link. That is, mapping faults are dependent. Hence, all the results originated from peer P_k will be considered incorrect for a particular query.   Based on (1) the current state of the link M, i.e., its prior value, and (2) the rate of fault occurrence, peer P_k and all other peers on the mapping paths going through peer P_k could be excluded from participation in emerging shared semantics.

Case 2 In this case, we are considering a situation when peers have k outgoing mapping links and $k > 1$. Figure 9 represents this case. It shows that P_k has three outgoing mapping links $\{M_{k1}, M_{k2}, M_{k3}\}$. Hence, the decision on the reliability or trustworthiness of the outgoing link M_1 does not depend entirely on the outgoing link M_{k1}, as it was the situation in case 1. Nonetheless, not distinguishing between transient and permanent mapping faults, i.e., treating the mapping link M_k as permanently faulty, will have an impact on the perception about the correctness of the outgoing mapping link M_1.

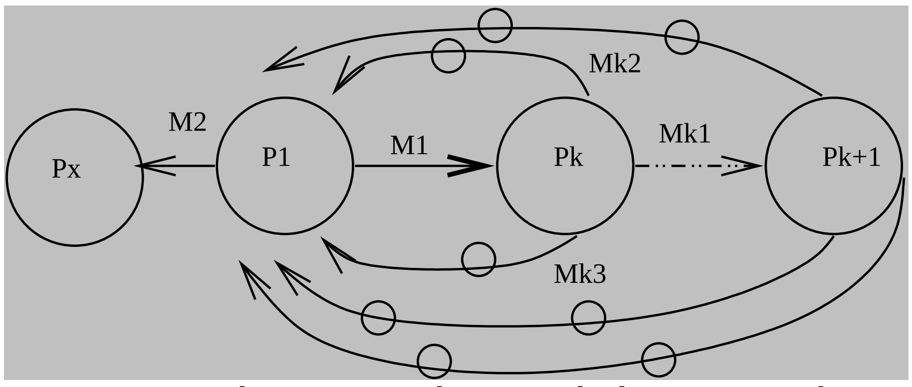

Figure 9 Peers on the Mapping Path Have Multiple Outgoing Links

The wrong perception about any outgoing mapping link, when peers have k outgoing links, could impact the way subsequent queries will be routed. Consider the situation shown in Figure 9. If the original trust in the outgoing links M_1 and M_2 were X and Y respectively and X-Y = d, then, if a transient fault on the mapping link M_{k1} downgrades the trust value of M_1 by a value Z, where Z > d, the peer P_k will favor M_2 over M_1 the next time it needs to forward a query. This, could, in turn, isolate other peers from participating in future collaborations, and lower the precision and recall of query results because of a lower number of peers participating in answering the query.

These cases lead us to believe that a complete semantic emergence between independent and heterogeneous ontologies is not possible without tolerating temporary semantic mapping faults. In the next section, we will propose a solution to non-permanent semantic mapping faults for emergent semantics.

7. Fault-tolerant Emergent Semantics

Not all existing fault recovery techniques, e.g. checkpoints, rollback, and error log analysis are appropriate for P2P networks. Some of the existing solutions are difficult to implement and others are not appropriate for the context of semantic mapping.

We propose a solution to detect and correct non-permanent semantic mapping faults. Our solution is simple in concept and easy to apply. It is based on the *time redundancy* technique, which is well-known in the fault tolerance literature (Anghel, 2000; Avizienis 1995; Dupont 2002; Paradhan , 1996).

In this context, we refer to the replication of a query and checking for the query answer consistency as time redundancy.

Time Redundancy

Software fault-tolerance can be accomplished through various methods including information, component and time redundancy. The choice of time redundancy is more applicable than component redundancy (N-version programming) and information redundancy in the semantic mapping context. This is because peers are autonomous, and no peer has control over another peer's fault handling mechanism. More importantly, in the context of semantic mapping, we are talking about peers sending queries to other peers and not getting correct responses. Hence, the most appropriate way to determine whether a fault is permanent in such a setting is to resend the query.

The time-redundancy technique can be used to add fault-tolerance capabilities to semantic mapping-based systems in at least two ways: (1) querying the peer service provider more than once at different times and comparing the obtained results, and (2) preparing a test query for which the querying peer knows the answer. In both of these cases, the query initiator can directly verify whether the related peers executed correctly (Papalilo, 2005).

Proposed Algorithm

The procedure of our proposed algorithm for tolerating non-permanent semantic mapping faults comprises of two main parts: fault detection and fault recovery. The algorithms steps are:

1) To detect faults, peers will be tested with a repeated query as follows:

   a) Submit K sequential queries in place of one query every time query submission or query forwarding is performed. Queries are separated from each other by a time δ. For instance, if K is set to 2 then the origin query and its clone will be separated by δ time. That is, the second query will be posed at $t_0 + \delta$, where t_0 is the time for initial query and δ is the delay time between the two sequential queries. The system designer determines the maximum transient-pulse duration δ that the system must tolerate. It is a system variable, and characterizes the length of the time which system is guarded against the negative impact of the transient fault. That is, faults occurring during the transient-pulse period will have no impact on the system operational.
   b) Query answers from replicated queries are compared for consistency. The inconsistency among answers for the same query is a deciding criterion for the transient fault occurrences. The consistency checking leads to the following two cases:

      (i) If query answers were consistent and incorrect then querying peer concludes that the queried peer is incapable of providing an answer to the query. Hence, it is permanently faulty relative to the posed query.
      (ii) If query answers were inconsistent, then a transient fault must have occurred, and an action should take place to eliminate its negative impact.

   The consistency relation is a system-defined relation. An example of consistency relations between two answers A_{s1} and A_{s2} is $\{\subset, \supset, \equiv\}$ where answers $A_{s1} \subset A_{s2}$ means that A_{s1} is less general than A_{s2}, $A_{s1} \supset A_{s2}$ means that A_{s1} is more general than A_{s2} and $A_{s1} \equiv A_{s2}$ means that A_{s1} and A_{s2} are identical.

2) A transient fault recovery action comprises two steps:

   a) Query answer cancellation. If a transient fault is detected, the infected query impact on the semantic relation between peers should be ignored. This is achieved by sending a cancel signal to the peer that originally initiated the query. This signal indicates that answers resulting from the query should be ignored. The cancel signal has one parameter, a *query-id*. The query-id identifies the query for the semantic mapping under investigation. As each peer returns the cancel signal to the peer it received the query from, the signal reaches the query initiator and the result of a query with the query-id in the cancel signal will be ignored. The result of query will be ineffective on grounds of the trust peers have in their outgoing links.

b) Query re-submission. In order for queries to recover from the impact of the transient faults, query re-submission needs to take place. This happens after waiting for Δ_2 length of the time from the last time a transient fault is detected and query re-submission could take place. The query re-submission can be repeated up to K times. The Δ_2 value and the *number of query retry* are system parameters. These values will be set by system administration in such a way that a system will maximize the recall for the least additional queries. These values could be determined experimentally. Further, Yinglong et at. (2007), suggests that ontology update notification in distributed systems should be enforced and performed within a time window. In the latter case the Δ_2 value will be set equal to time constraint.

We can make the following observations about this algorithm:

- Query consistency checking could be accomplished in two different ways: Central Checking and Distributed Checking.

    (i) Distributed Checking: when peers receive identical queries separated from each other by Δ time, they can use this knowledge for testing their immediate neighbors on the mapping path. This will be done by comparing query answer results. An inconsistency among query answers would be an indication of transient faults.

    (ii) Centralized Checking: the querying peer will receive all answers for repeated identical queries. That is, the querying peer will receive a number of answer sets of different size and checks for consistency by comparing the answer sets. An inconsistency among answer sets would be used for transient fault detection.

There are tradeoffs for each approach. In the distributed fault detection approach, peers are required to perform some extra functionality, and send an extra message once a fault has been detected. In the centralized approach, a new policy for query re-submission and out-going mapping link reward (punishment) could be adapted. For example, queries could re-submit only when the all answers of the first query were wrong resulting in a reduced number of messages exchanged. However, the heavy computation that has to be done on querying peer and computation capabilities of existing peers will not be used.

- The impact of submitting multiple queries in place of one query on the level of the confidence peers have in their out-going mapping links has to eliminate. This could be done by assigning unique IDs to queries. Multiple queries with the same ID will be treated as one query. A querying peer will reward (punish) an outgoing mapping link only once for receiving multiple answers from multiple queries with same query ID.

- Re-submitting queries in working environment with frequent transient faults could end-up repeating queries forever. To cope with this situation, our proposed solution will tolerate up to x transient faults, where x is a system variable whose value is determined as a ratio of queries to transient faults. Peers need to do some extra computation and book keeping works in order to determine the value of x.

- All peers run the same algorithm, and they update their confidence value in out-going mapping links based on the results they obtain from their own queries. The updating policy could be changed by requesting querying peers to propagate their acceptance (rejection) to query answers along the mapping path(s). This will help peers along the mapping path to update their confidence value in their out-going mapping links, based on the use by other peers.

Figure 10 shows a diagram of our proposed solution, where K = 2 and the delay between queries is δ. The two arrows, **As1** and **As2** represent the query answers. In a case when there is no answer to a query, the arrow represents the time-out signal from the system clock. Having each peer check and capture transient mapping faults, we will build a robust system where the chances for expelling peers for non-permanent semantic mapping faults are minimized.

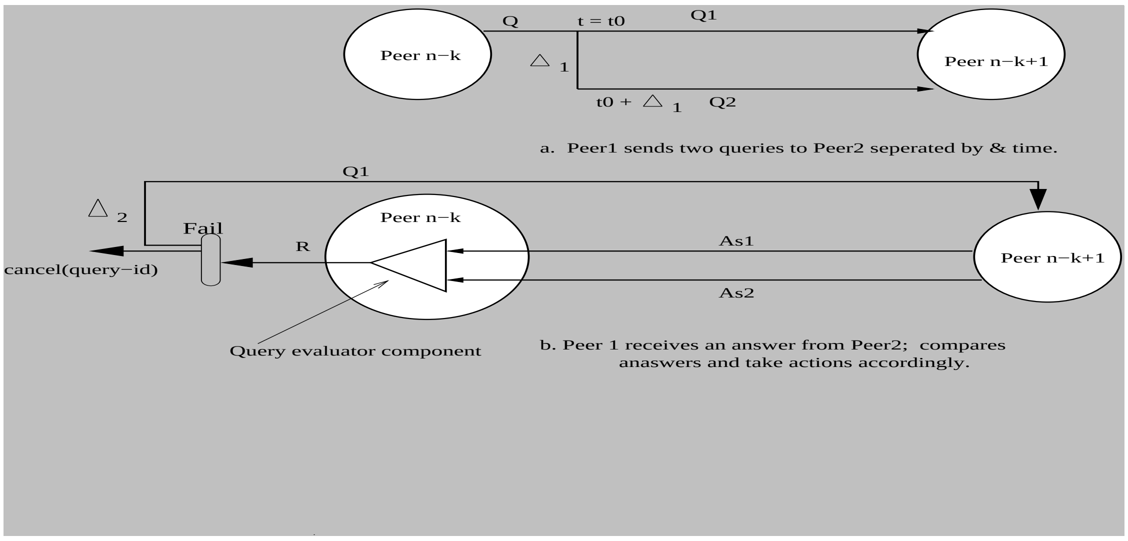
Peer n−k
Q
t = t0
Q1
Peer n−k+1
△ 1
t0 + △ 1
Q2
a. Peer1 sends two queries to Peer2 seperated by & time.
Q1
△ 2
Fail
cancel(query−id)
R
Peer n−k
As1
As2
Peer n−k+1
Query evaluator component
b. Peer 1 receives an answer from Peer2; compares anaswers and take actions accordingly.

Figure 10 Query Replication and Answer Consistency Checking Steps

The algorithm is valuable, not only because it is simple in concept and easy to apply, but also because it is capable of detecting a range of faults without the need for knowing the causes of the faults. However, the algorithm will increase in the number of messages exchanged, could increase in computation time and will not detect all faults (e.g. faults caused by static mapping and context will not be detected).

Alternative Approaches

A key element for fault-tolerance is *redundancy*. That is, additional resources to provide capability for detecting/tolerating faults which will not be used if the fault-tolerant were not required (Laura, 2001). In the following we discuss briefly some alternatives to the described time redundancy technique.

Inspired by hardware redundancy, *Software Redundancy* techniques have emerged to tolerate software faults. It encompasses additional software programs, components, modules or codes in the system to support fault tolerance. Software redundancy could be used for tolerating software faults originating from requirement specification, design and/or coding errors. Different versions of the same software with the same functionality would be developed by different teams and possibly in different programming languages. Each software version would run on a separate machine, with same input. The program outputs are then compared for consistency.

Performing multiple computations in a dynamic environment such as P2P semantic knowledge sharing systems is difficult and subject to termination, thus depriving peers from opportunities to produce responses. A reasonable alternative would be the duplication of critical variables and/or blocks of code and comparing the output of these code blocks and variables at different stages of the execution of the same program.

Information or data redundancy, some times grouped with software redundancy, utilizes diverse data, i.e., variations in the input format or structure, to assist in fault tolerance. A data re-expression algorithm could be used to generate multiple formats of data with same content but different representations. The generated data, multiple identical queries in the context of P2P semantic knowledge sharing systems, is then used as input into different versions of the same programs or, software component, for detecting and tolerating faults.

One more way to enable P2P semantic knowledge sharing systems to be fault-tolerant is by using the *majority voting technique*, a well-known technique used for determining consensus outcome from the results delivered by multiple computation sources. Consider the P2P semantic knowledge sharing system presented in Figure 11, where nodes represent peers, links represent mapping among peers and directed paths represented query answers to the query *Q* initiated by peer *A*.

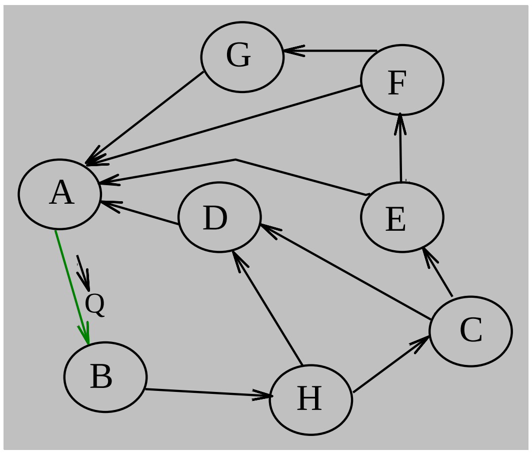

Figure 11 Multiple Answers Use Voting for Consensus

The selection (de-selection) of a query result among multiple results returned from different translation paths by *query initiator* could by done using simple majority voting technique. The voting technique here would serve three purposes: (i) reduces number of answers to one answer, (ii) increase the confidence in the query result since it is been asserted by the majority of mapping paths, and (iii) increases the trust in the decision that will be made about correctness of the out-going mapping links.

In summary, there are various techniques for improving system fault-tolerance capability. Some of these techniques are well-known for handling faults in certain situations. As discussed above, the time redundancy, software block redundancy and voting techniques are more suitable than others for adding fault-tolerance to the P2P semantic knowledge sharing systems. Developing software, time and information redundancy based algorithms for tolerating faults in P2P semantic knowledge sharing systems, and carrying out empirical studies in order to determine the best possible usage of redundancy techniques for tolerating faults is a significant contribution to the science.

CONCLUSION

In this chapter, we identified one of the shortcomings of existing research on emergent semantics: current approaches fail to distinguish between permanent and transient semantic mapping faults. Instead, they treat all faults as permanent.

We identified a list of different situations that could cause non-permanent semantic mapping faults and classified them along temporal dimension.

We demonstrated that it is difficult to reach a complete state of emergent semantics among independent, heterogeneous and distributed local ontologies, unless the system can eliminate or reduce the impact of temporary semantic mapping faults. The reason is that treating all faults as permanent may result in the erroneous labeling of peers as having incompatible knowledge representations, which reduces the number of peers that can participate in emerging shared semantics.

We proposed to solve the lack of fault-tolerance problem using standard techniques from fault-tolerant discipline. The proposed solution improves the opportunities for emerging more agreeable semantics, as a higher number of peers will be available to participate in emerging shared semantics. Teaming up more semantically related peers with one another also could enable query answers with higher precision (recall).

9. FUTURE RESEACH DIRECTION

In addition to the demand for augmenting fault-tolerance capability, we believe that having more intelligent peers than those used in current emergent semantics system is another viable and open research area. Peers participating in the emergent semantic could be equipped with *learning* capability. They could make use of the discovered semantic affinity between their own concepts and concept from other peers' ontologies for better query routing answering.

Storing network *locations* of foreign concepts that are identified, during query processing and answering, as complementary concepts or properties to the peers' own concepts, constitute the learning capability peers lack today. Another important learning issue would be aligning or integrating semantically related foreign concepts into local ontologies. Peers should preserve their state. State maintenance refers to the endowment of peers with the ability to preserve the knowledge they acquire from the interaction with other peers in the network between sessions, i.e. between peers disconnection and reconnection to the network.

Investigating emergent semantics system performance would be another important issue to be considered in future research. This includes further study on query answer caching, overcoming point-to-point semantic mapping limitation without the need for query broadcasting.

Studying the effect of non-permanent semantic mapping faults in areas such as consensus formation, semantic Web services and semantic negotiation is an important matter worth investigating.

We consider the above-mentioned issues to be viable and open research subjects which need further study and attention from research communities.

Acknowledgement

I would like to thank Dr. Abdulghany Mohamed from Sprott School of Business, Carleton University for his valuable time, thoughts and discussions we had throughout writing this chapter.

QUESTION FOR DISCUSSION

Basic:

1. Define the semantic heterogeneity and semantic interoperability problems.
2. Define the concept of semantic reconciliation.
3. What are semantic mapping faults?
4. What is the difference between a semantic mapping fault and a *temporal* semantic mapping fault?
5. What are advantages/disadvantages of common upper ontologies?
6. What are advantages/disadvantages of local ontologies?
7. Why do some researchers prefer simple, less expressive ontologies over complex ontologies for open and dynamic environments?

Intermediate:

1. Name two consequences of failing to tolerate temporal semantic mapping faults.
2. How could peers extend their local ontologies using the proposed fault-tolerant emergent semantics?
3. Can you think of causes of temporal semantic mapping faults other than those described in the chapter?

Advanced:

1. Can you describe a technique, other than time redundancy, to solve temporal semantic mapping faults?
2. Using the provided algorithm, is it possible to identify precisely the cause of temporal faults any time a fault occurs? How important is that?
3. Should we design a set of criteria for selecting query answers, and what are they?
4. How would you measure the effect of non-permanent semantic mapping faults on emergent semantics?
5. Create a simulation model for transient semantic mapping faults?

SUGGEGESTED ADDITIONAL READING

Antoniou G. and Harmelen F. V. (2004). A Semantic Web Primer. Cambridge, Massachusetts, The MIT Press, USA.

Berners-Lee T. (2000) Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. Collins Publishing, New York, USA.

Bonifacio M., Bouquet P., et al. (2004). Peer-Mediated Distributed Knowledge Management. Abecker A., Dignum, V & Elst L. v. (Ed.), Agent-Mediated Knowledge Management: International sysmposium Amkm, Springer, 31-47.

Cardoso J. & Sheth A. P. (Eds.) (2006). Semantic Web Services, Processes and Applications, Springer publication, USA.

Elmagarmid A. Rusinkiewicz M. & Sheth A. (1999). Management of Heterogeneous Autonomous Database Systems, Morgan Kaufmann publication, San Francisco, California.

Klein M., Kiryakov A. et al (2002). Finding and Characterizing Changes in Ontologies. In *Conceptual Modeling – ER, 21st International Conference on Conceptual Modeling* Proceedings, 79-89.

Lacy L. W. (2005). OWL: Representing Information Using the Web Ontology Language, Victoria, BC, Canada.

Mawlood-Yunis A-R., Weiss W. & Santoro N. (2006). Issues for Robust Consensus Building in P2P Networks. In *Intl. Workshop on Ontology Content and Evaluation in Enterprise (OnToContent),* LNCS 4278, 1020-1028.

Mena E., Kashyap V. et al (2000). Imprecise Answers in Distributed Environments: Estimation of Information Loss for Multi-Ontology Based Query Processing. *Journal of Cooperative Information Systems*, 9(4), 403-25.

Stuckenschmidt H. & Harmelen F. V. (2005), Information Sharing on the Semantic Web, Springer publication, Germany.

Spaccapietra Stefano (Ed.). *Journal on Data Semantics* Vol. I, II, III, IV, V, VI, VII. Springer Publishing.

REFERENCES

Aberer K., Cudre-Mauroux P. & M. Hauswirth (2003). Start Making Sense: The Chatty Web Approach for Global Semantic Agreements. *Journal of Web Semantics*, 1(1), 89-114.

Aberer K., Catarci T. et al. (2004). Emergent Semantics Systems. *In Lecture Notes in Computer Science Vol.3226,* p 14-43.

Anghel L., Alexandrescu D. & Nicolaidis M.(2000). Evaluation of a Soft Error Tolerance Technique Based on Time and/or Space Redundancy. In Proceedings of *13th Symposium on Integrated Circuits and Systems Design*, IEEE Computer Society, 237 – 242.

Avizienis A. (1995). The Methodology of N-Version Programming. In M. R. Lyu (ed.), Software Fault Tolerance, Wiley, 23-46.

Axelrod R. (1997). The Complexity of Cooperation. Princeton University Press.

Bahceci E., Soysal O. & Sahin E. (2003). A Review: Pattern Formation and Adaptation in Multi-Robot Systems. Report CMU-RI-TR-03-43, Robotics Institute, Carnegie Mellon University.

Bondavalli A., Giandomenico F. D & Grandoni F. (2000). Threshold-based Mechanisms to Discriminate Transient from Intermittent Faults. IEEE Trans. on Computers, 49(3), 230-45.

Bondavalli A., Chiaradonna S. et al (1997). Discriminating Fault Rate and Persistency to Improve Fault Treatment. In Digest of Papers. Twenty-Seventh Annual Intl. Symposium on Fault-Tolerant Computing, 354-62.

Bonifacio M., Bouquet P., et al. (2002). KEx: a Peer-to-Peer Solution for Distributed Knowledge Management. In Karagiannis D. & Reimer U.(Eds.) Practical Aspects of Knowledge Management, 4th International Conference, LCNS, Springer, 490-500.

Bouquet P., Giunchiglia F. et al (2003). C-OWL: Contextualizing Ontologies. In *2nd Intl. Semantic Web Conf.*, Springer. LNCS vol. 2870, 164-179.

Castano S., Ferrara A. & Montanelli S. (2003). H-Match: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. In the 1st VLDB Int. Workshop on Semantic Web and Databases (SWDB), 231-250.

Dupont E., Nicolaidis M. & Rohr P. (2002). Embedded Robustness IPs for Transient-Error-Free ICs. IEEE Design & Test, 19(3), 56 - 70.

Fergus P., Mingkhwan A. et al.(2003). Distributed Emergent Semantics in P2P Networks. In Second IASTED Intl. Conf. on Information and Knowledge Sharing,75-82.

McGuinees D. L. (2003) Ontologies come of Age . Book chapter in Spinning the Semantic Web: Bringing the World Wide Web to its Full Potential, Fensel D. Hendler J. Lieberman H. et al. (eds). The MIT Press.

Gal A. (2001) Semantic Interoperability in Information Services: Experiencing with CoopWARE. SIGMOD Record, 28(1),68-75.

Ghidini C. & Giunchiglia F. (2001). Local Models Semantics, or Contextual Reasoning = Locality + Compatibility. *Artificial Intelligence Archive*, 127(2), 221-259.

Glushko R. J. & McGrath T. (2005). Document Engineering, Cambridge, Massachusetts, The MIT Press.

Gomez-Perez A., Fernandez-Lopez M. and Corcho O. (2003). Ontological Engineering. Springer publishing, London.

Gruber T. R. (1993). A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition Archive*, 5(2), 199-220.

Guarino N. (1998). Formal ontology and information systems. Proceedings of Formal Ontology in Information Systems, pp. 3-15, 1998.

Gutierrez, C., Hurtado and C. Vaisman, A. (2005) Temporal RDF. *The Semantic Web: Research and Applications. Second European Semantic Web Conference, ESWC 2005. Proceedings (Lecture Notes in Computer Science Vol. 3532),* p 93-107

Haase P., Broekstra J. et al. (2004). Bibster -- A Semantics-based Bibliographic Peer-to-Peer System. In Third Intl. Semantic Web Conf. (ISWC)}, 122-136.

Halevy A., Ives Z. et al. (2003). Piazza: Mediation and Integration Infrastructure for Semantic Web Data. In the Intl. Worldwide Web Conf. WWW-03.

Hurtado, C.and Vaisman, A. (2006) Reasoning with temporal constraints in RDF. *Principles and Practice of Semantic Web Reasoning. 4th International Workshop, PPSWR 2006. Revised Selected Papers (Lecture Notes in Computer Science Vol. 4187),* p 164-78

Larry K. Hanjo J. & Wooju K (2006). Emergent Semantics in Knowledge Sifter: An Evolutionary Search Agent Based on Semantic Web Services. Journal on Data Semantics VI, 187-209.

Laura L. P. (2001). Software Fault Tolerance Techniques and Implementation. Norwood, Mass. Publication.

Mawlood-Yunis A-R., Weiss M. and Santoro N. (2007). Fault Classification in P2P Semantic Mapping. *In Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa)} at Intl. Conf. on Artificial Intelligence (IJCAI).*

Mena E., Illarramendi A. et al (2000). OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interpretation Across Pre-existing Ontologies. Distributed and Parallel Databases, 8(2), 223-71, 2000.

Mika P. (2005). Ontologies are Us: A Unified Model of Social Networks and Semantics. In *4th Intl. Semantic Web Conference,* 522-36.

Naiman C. F. & Ouskel A.M (1995) A Classification of Semantic Conflicts in Heterogeneous Database Systems. *Journal of Organizational Computing*, 5(2),167-193.

Noy N. F. & Musen M. A. (2004a). Ontology Versioning in an Ontology Management Framework. IEEE Intelligent Systems, 19(4), 6-13.

Noy N. F. & Klein M. (2004b). Ontology Evolution: Not the Same as Schema Evolution. *Knowledge and Information Systems,* 6(4), 428-440.

Ouksel A.M (1999). Ontologies are Not the Panacea in Data Integration: A Flexible Coordinator to Mediate Context Construction. Distributed and Parallel Databases, 15(1), 7-35.

Paradhan D. K. (1996). *Fault-Tolerant Computing System Design*. Prentice-Hall PTR publication.

Papalilo E., Friese et al (2005). Trust Shaping: Adapting Trust Establishment and Management to Application Requirements in a Service-Oriented Grid Environment. In *Proc. 4th Intl. Conf. on Grid and Cooperative Computing (GCC),* LNCS 3795, pp. 47-58,

Pizza M., Strigini L. et al. (1998) Optimal Discrimination between Transient and Permanent Faults. In *Third IEEE Intl. High-Assurance Systems Engineering Symposium,* 214-23,.

Ram S. & Park J. (2004) Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflicts. *IEEE Transactions on Knowledge and Data Engineering,* 16(2), 189-202.

Robert. Mc et al. (2005)Mapping maintenance for data integration systems. In Proceedings of the 31st international conference on VLDB, Pages: 1018-1029.

Roddick J. F. (1995). A Survey of Schema Versioning Issues for Database Systems. *Information and Software Technology,* 37(7), 383-393.

Rousset M. C. (2004). Small Can Be Beautiful in the Semantic Web.
In *Third Intl. Semantic Web Conf.*, p 6-16.

Rousset, P. Chatalic et al. (2006) SomeWhere: A Scalable P2P Infrastructure for Querying Distributed Ontologies. *5th Intl. Conf. on Ontologies Databases and Applications of Semantics,* 698-703.

Staab S. (2002) Emergent Semantics. *IEEE Intelligent Systems*, 17(1), 78-86.

Staab S.(2005) Social Networks Applied. *IEEE Intelligent Systems*, 20(1),80-93.

Staab S. & Stuckenschmidt S. (2006). *Semantic Web and Peer-to-Peer,* Springer-Verlag, Berlin Heidelberg, Germany.

Stephens L.M. and Huhns M.N.(2001) Consensus Ontologies. Reconciling the Semantics of Web Pages and Agents. In *IEEE Internet Computing,* 5(5), 92-95.

Tempich C., Staab S. Wranik A.(2004). REMINDIN': Semantic Query Routing in Peer-to-Peer Networks Based on Social Metaphors. *In 13th Intl. Conf. on the World Wide Web,* 640-649.

Tempich C., Pinto H.S. et al. (2005). An Argumentation Ontology for Distributed, Loosely-controlled and Evolving Engineering Processes of Ontologies (DILIGENT).*In Second European Semantic Web Conf. (ESWC),* LNCS 3532, 241-56.

Wu J. and Axelrod R.(1995). How to Cope with Noise in the Iterated Prisoner's Dilemma. *Journal of Conflict Resolution,* 39(1),183-189.

Yinglong M. , et al. (2007) A Timing Analysis Model for Ontology Evolutions Based on Distributed Environments. In (Zhi-Hua Zhou, Hang Li, Qiang Yang (Eds.): Advances in Knowledge Discovery and Data Mining, 11th Pacific-Asia Conference, PAKDD, LNCS 4426, pages 183-192.

Zhu H., Madnick S. E. & M. D. Siegel (2004). Effective Data Integration in the Presence of Temporal Semantic Conflicts. In *11th Intl. Symposium on Temporal Representation and Reasoning*, 109-114.