

Bridging heterogeneous, autonomous, dynamic knowledge at runtime

Declan O'Sullivan, Ruaidhrí Power
Knowledge and Data Engineering Group
Department of Computer Science
Trinity College Dublin

Abstract

The ubiquitous computing vision assumes that a wide range of application/services will interoperate seamlessly. Reaching global agreements on interoperability standards (which is the current assumption) will be impossible. Ubiquitous Computing applications/services will need to utilise knowledge encoded in a variety of formats, gathered and controlled by diverse entities, and which have a dynamic deployment lifecycle. Traditionally providing an integrated view upon such diversity has been achieved at design time. Model heterogeneity has been overcome through common models and expert mappings, coupled with an assumption of a static knowledge source deployment lifecycle. In addition, there had been acceptance that autonomy had to be ceded by knowledge owners in order to participate in an integrated view.

In this paper the authors contend that application interoperability solutions are needed which are metadata and ontology based, loosely coupled and runtime based. In particular two major interoperability use cases for applications in ubiquitous computing are examined: the sharing of context information and the matching of concepts based on semantics. In both cases a solution is needed that will bridge heterogeneous, autonomous, dynamic knowledge sources at runtime. For each case, the paper reports upon initial experiments and results. In conclusion, the experiments demonstrate that a variety of ontology, loosely coupled and runtime approaches show promise in bridging the kind of heterogeneous, autonomous, dynamic situations that will be prevalent in the ubiquitous computing era.

1. Introduction

The source of the most serious challenges to deploying the ubiquitous computing vision are not technological but structural (O'Sullivan and Lewis 2003). Embedding processors, sensors and actuators in everyday products implies an explosion in the

number and type of organisations that need to be involved in achieving the seamless interoperability implied by the ubiquitous computing vision. Many of the network interoperability problems can be addressed by the inter-networking approach of the Internet's network and transport protocols. However, the potential for debilitating heterogeneity in application level interoperability remains. Consider the complexity involved in reaching agreements on and enforcing conformance to interoperability standards when the players involved expand from the computer and software developers (Microsoft, IBM, HP and the like) to all the organizations which will embed application into their products, e.g. windows, alarm clocks, pens, automobiles, coffee machines etc to name a few implied by Weiser's original ubiquitous computing scenario (Weiser 1991). It is therefore clear that the ubiquitous computing vision implies a massive increase in scale of the application interoperability problem.

The work presented here is motivated by the observation that we cannot, therefore, rely on shared a priori knowledge via common interoperability standards to solve the application interoperability problems on the scale needed for ubiquitous computing. Instead, application software must somehow adapt at deployment time and runtime to integrate their functionality and dynamically interoperate with other application software. Two major interoperability use cases for applications in ubiquitous computing will be: the sharing of context information and the matching of concepts based on semantics. In both cases a solution is needed that will bridge heterogeneous, autonomous, dynamic knowledge sources at runtime. In the context sharing case, the knowledge sources in question represent a wide variety of context generation services (sensors, databases, application software, etc.) as well as context aggregation services. In the concept matching case, the heterogeneous knowledge sources in question are ontologies.

The following two sections in turn outline each use case, describe the rationale for the initial

experiment undertaken, and go onto detail the experiment and preliminary results. The last section presents conclusions and identifies future research directions.

2. Context Sharing

2.1. Background

The vision of ubiquitous computing is that computers will be integrated seamlessly into our daily lives. We already make much use of computers, however we can gain the most value from them when they are no longer things we interact with explicitly, rather are blended into the background and assist us when needed. In order to do this, ubiquitous computing environments must be able to collect a wide range of information and use this information to work with the user in order to achieve the user's goals. This information is termed context information, and its collection and management is termed context management.

One of the realities that must be faced in context management is that there will not be a globally standard model for representing context information. Many current approaches (Mitchell 2002) to context management advocate pre-defined models for context information, which applications interact with using middleware platforms for querying and manipulation. However, context data will come in many different forms, from many different sources. Any attempt to formally structure all potential context information would be difficult at best in a controlled situation, within one organisation for example, but almost impossible in an inter-organisational scenario.

Context information for ubiquitous computing environments have particular characteristics, which provide challenges in undertaking context management. Firstly, the information that can compose the context of these environments is very broad, and can come from a variety of heterogeneous sources. A user's name, age, address, native language, current location and learning style could compose part of his context. Similarly, the people sharing a room with him or working in his office could be considered to be part of this context information, as could the current temperature and lighting conditions. Any system for context management must therefore be able to cope with information from a large variety of heterogeneous sources that will provide this information. Because almost any information could be considered context information from the point of view of some entity in a ubiquitous computing environment, there is very little information that we can discard as being irrelevant. Perhaps the most important characteristic of context information is that we cannot be entirely certain what information will be relevant in advance

of constructing a system to manage this information. A useful solution to the problem of context management will therefore have a low impact on existing infrastructure, and cope well with heterogeneity. Such a system should also cope well with new forms of context information.

The second challenging characteristic of context information in a ubiquitous computing environments arises from the fact that the environment will consist of a highly dynamic collection of users and computing devices. These devices must seamlessly integrate with whatever computing environment they are presented with, so that their users can make most efficient use of them. In this environment a roaming user or device is the norm, rather than the exception. These environments frequently make use of temporary, ad hoc connections between devices to accomplish tasks. Therefore, context information systems must be able to dynamically discover and connect to information sources in order to extract data and manipulate it into relevant context knowledge. This frequently changing environment can lead to uncertainty: where gathered information can quickly become stale; services and devices can also suddenly become available or unavailable due to changes in connectivity.

Finally, these environments should present context information in terms that the user can relate to, in other words the information should be user-centric. Context information will almost certainly be managed by the entity to whom that context information relates (for example a business or an ordinary individual), rather than being managed globally. This is due to a few factors: the sheer volume of data that will compose context will make a global view of all context information impossible. Privacy and security concerns will also prompt people to manage their own context information. Perhaps most importantly, the set of information that will compose this context is so dynamic that it will never be standardised, so mechanisms will have to be developed to promote interoperability between context systems. These mechanisms will translate context into a form where it can be understood by each organisation's context system. This is particularly the case for roaming applications, where context information must be supplied and received for a roaming user or device to avail of services within another environment. A characteristic of a good solution will be that this information will be merged into each user's own view of the world, and redefined in terms that the user can understand.

2.2. Rationale for Topic Maps for Context Management (TM4CM)

A Context Information System (CIS) is a software system whose task is to perform the

gathering and management of context information from the ubiquitous computing environment and make this information available to consumers of context. A CIS therefore needs a way of representing the data that can be retrieved from these individual sources in a way that can be processed and reasoned about by a computer. This representation is termed metadata. Additional knowledge can be encoded by adding connections representing the relationships between the metadata objects within the CIS. This knowledge will give the CIS an overall view of context data being provided to the system, and it will then be possible to use this knowledge both internally within the CIS to make decisions and also to provide views upon this knowledge to interested context-aware clients.

The authors are examining Topic Maps (www.topicmap.com) as a means to cope with the characteristics of context information in ubiquitous computing environments, as outlined in Section 2.1, namely: heterogeneity, distribution and autonomy of context information sources, and dynamism/accuracy/timeliness of the context information.

Topic maps is one candidate for the task of encoding the metadata. This method attempts to connect pieces of data into a graph that represents the relationships between them. Topic Maps do not attempt to enforce a rigid structure on the information they describe but rather provide a lightweight way of navigating and accessing the information that exists in separately maintained data sources. Another potential application of topic maps is to facilitate translating the knowledge provided by one source into a form that can be used elsewhere. This concept is interesting in a ubiquitous computing scenario, where the form of the information cannot be completely defined a priori, particularly when availing of context information from sources external to one's organisation, for example when roaming.

Topic maps provide a common framework for managing interconnected sets of information objects. These information objects are the 'subjects' that are represented as 'topics' within the topic map. A subject can be pretty much anything that you can think or reason about: a chair, a webpage, a person, the sun, the concept of religion, or your favourite television character. A topic is created within the topic map to 'indicate' (or refer to) this subject.

Information is then added to these topics by creating associations between them, thereby encoding additional knowledge. For example, the fact that a person has a webpage could be described by an association 'hasWebpage' between the topics representing the person and their webpage.

Since topics and associations can represent almost anything we can think of, topic maps are a

means of representing a virtually unlimited number of relationship types between a virtually unlimited number of information types. When we are given these topic maps, we can ask interesting questions of them by following the associations between topics.

Another feature of topic maps are occurrences, which are information resources relevant to a topic. A topic representing a person could have occurrences such as a portrait, a CV or some descriptive text. These resources are linked to from the topic itself, meaning that anyone with a reference to the topic representing that person would also automatically have access to these occurrences which would give more information. Occurrence types specify which sort of information the occurrence links to, giving any processing application the option of filtering based on occurrence type.

Two topic maps can be merged if they both have topics that refer to the same subject. When they are merged, a new topic is created with the union of the characteristics (names, associations and occurrences) of the two originals. The new topic map that is created will still contain all the information that was in these originals. Topic map merging works best when the topic maps generally refer to the same subjects, producing a valuable new set of information.

2.3. Experiment

The objectives of this experiment were to address the following issues in the design of a context information system:

- To discover how topic maps could be used to support context management;
- To show how raw context information might be distributed across context sources through example scenarios;
- To demonstrate how the merging and querying of this information from different sources into a combined view would work.

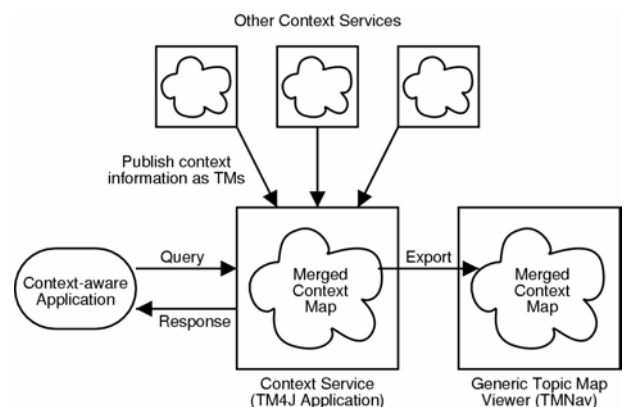


Figure 1. Experiment Architecture

The experimental architecture (as shown in Figure 1) represents a distributed set of context services. These services take input in the form of information from the environment. This may be information from a database, user-supplied information, data from a sensor or some other source. Each context service is a designated source for context information about a particular set of concepts, for which it has this raw information from the environment. This information is processed by the context service, and made available to the context clients in the form of a topic map.

In the current implementation all the context information is available within the topic map. However, it is possible for the topic map to simply reference external occurrences, thereby keeping the topic map more lightweight.

Context clients can take the topic map and merge it with their own, producing a combined topic map. They can then perform queries on the combined topic map, which contains the sum of context knowledge from each of the context services that provided topic maps. These queries would typically be embedded in a context-aware application, which would parse their result and take action based on that result.

Each context client may also act as a context service, and export context information to other clients. All of the topic maps produced in this system can be viewed by any generic topic map viewing application, such as TMNav (<http://tm4j.org/tmnav.html>).

Two use cases from a student project meeting scenario were modelled in the form of topic maps. One of these use cases was electronic communication between students in meetings, which requires discovery of currently active meetings and of contact details of students attending those meetings. The other use case was of document annotation, allowing students to attach notes to documents they are working on, while keeping information about the authors of the documents and of the notes.

The context services as chosen in the experiment each managed pieces of context information, that would be available to them directly from external sensors or databases. The context services modelled represented:

- People participating in the meetings;
- Concepts associated with people such as their e-mail address, date of birth, etc;
- The meetings taking place;
- Concepts used within an organisation such as the idea of a meeting, and of a person attending a meeting;
- Documents produced in those meetings.

The information from the context services, made available as XML topic maps, was merged into a combined view using the TM4J engine (<http://tm4j.org/tm4j-engine.html>). TM4J is an open-source package for creating topic map processing applications. The core of TM4J is a set of Java APIs for parsing, manipulating and saving topic maps. The combined topic map contained all the context information from the constituent context services. Figure 2 provides a simple example of this. It shows the merging of separate topic maps `org.xtm` and `person.xtm` to form `room.xtm` showing Ruaidhri Power attending a meeting in that room. If another person was to attend this meeting, it would simply require the creation of an association between the `MzMtg` topic and the topic representing that person and the merge of that person's topic map.

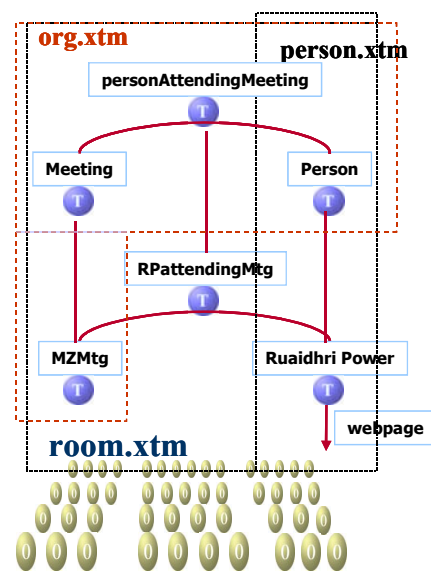


Figure 2. Example Topic Map Merge

2.4. Initial Results

The primary finding from this experiment is that topic maps hold promise as a basis for a context management system, in that they have the potential to bridge distributed heterogeneous and autonomous context information sources at runtime. In particular the possibility of dynamically generating topic maps from data provided by a context information source or the possibility of using the "occurrences" feature of topic maps may address the heterogeneity issue. The fact that topic maps can be independently authored and later easily merged has the potential to overcome the autonomy challenge.

However the authors believe that some additional features may need to be added to the topic map approach in order to provide full support for context management, especially to meet the dynamism/accuracy/timeliness characteristics previously outlined. One of these features is the

aggregation of low level data into higher-level context. For example, low-level data from information sources such as sensors might give the location of a person to an accuracy of a few feet. However, this information might be too fine-grained and quickly changing to be useful to a consumer of context. They might instead prefer to only see a person's location to the level of which room or building they are in. We therefore see a need to create topics within a topic map that are aggregations of lower-level context.

Security and privacy concerns must also be addressed. It will not be sufficient to simply allow all context information managed by a CIS to be accessible by any consumer of context, be it an application or another ubiquitous computing environment. Access control mechanisms must decide what context information to export and to which clients. These access control mechanisms should allow as much information to be passed to the clients as possible for them to accomplish their task without compromising sensitive information. This balance will not be easy to strike, and maintaining these access control rules will be non-trivial. There may however be scope for automated maintenance of these rules based on policy specifications.

The final addition to topic maps that is foreseen would be the concept of adding a degree of confidence to associations. This could be used for example to slowly expire context information that is not being refreshed over time. For example if a temperature sensor stopped giving readings it might be appropriate to decrease the confidence associated with the last reading it gave over time until the confidence approached zero.

TM4J was successfully used to merge all the topic maps in this experiment, and manipulate them programmatically. No deficiencies have yet been found in the engine that would make it inappropriate for use in context management. However, the current example makes use of approximately thirty topics and fifteen associations, which was not enough to stress test the TM4J engine. It is planned to undertake simulations that increase the number of topics in future work.

The tolog topic map query language was successfully used to express all the necessary queries. The queries tested in this experiment tested the basic functionality of the tolog query language. Approximately six tolog queries were issued upon the combined map, which provided results such as a list of people attending a particular meeting, or the notes on a particular document. The longer inference chains needed for more complicated queries can be tested only on topic maps with higher numbers of topics.

3. Concept Matching

3.1. Background

Sheth classifies the types of interoperability problems that can occur during the interchange between information systems as follows (Sheth 1998):

- System, heterogeneous hardware and operating systems;
- Syntactic, different representation languages and data formats;
- Structural, heterogeneous model representations;
- Semantic, different meaning of terms used in the interchange.

System and Syntactic interoperability problems are more easily dealt with. However, achieving Structural and Semantic interoperability (which together will be termed semantic interoperability in this paper) in information interchange between information systems of different parties continues to be a difficult problem. In order to achieve semantic interoperability in a heterogeneous environment, the meaning of the information that is interchanged has to be understood across the systems. A simple example would be where one party might call the invocation charge for a particular web service "Fee" with the currency type "US Dollars" and the other party calls the charge "Cost" and expects a "Euro" representation.

More generally, Ceri and Widom identify four categories of semantic conflicts (Ceri and Widom 1993):

- Naming conflicts where different names used to represent the same concepts, either homonyms and synonyms.
- Domain conflicts occur when different reference systems are used to measure a value. Examples are different currencies.
- Structural conflicts occur when different systems use different data organization to represent the same concept.

Metadata conflicts occur when concepts are represented as one type within the modeling type of one system and a different type within the other system (e.g. as at a schema level in one database and as an instance in another).

In the past, such semantic conflicts have typically been dealt with at "design time": through careful schema design in distributed database solutions; through the hand crafting of interoperability gateways (with system integrator solutions); or by forcing each system to conform to a standard mechanism for interchange (e.g.

ebXML). Although these traditional approaches have been successful in well understood/static interchange environments, each of these approaches are inappropriate for systems that want to interchange in dynamic environments (Cui et al 2002): the schema design solution fails due to the rapidly changing nature of the interchanges required; the handcrafting of gateways solution fails as it does not scale to large numbers of information systems; and the standards solution fails due to the lack of certainty as to whether there is a common interpretation of the standard.

Ontologies can be used to describe the semantics of information sources and make the content explicit, and thus can be used to discover semantic equivalence between information concepts. The use of ontologies as a possible solution to the semantic interoperability problem has been studied over the last six or seven years. Wache et al reviewed and categorized twenty five approaches that have been proposed over this period and concluded that reasonable results for integration can be achieved through ontology based approaches (Wache et al 2001). Like the traditional approaches most of the approaches proposed are "design time" solutions towards integration. However as ontologies represent semantics, then semantic interoperability can be achieved by runtime comparison of and inference about ontological information.

The role of ontologies is becoming important in order to realize the vision of the Semantic Web (www.semanticweb.org) and several XML based representation languages are emerging. DAML and DAML+OIL (DAML 2003), and OWL (W3C 2003) are based on W3C's RDF Resource Description Framework. XML Topic Maps (XTM 2003) are based on the ISO 13250 standard (which was originally defined to facilitate merging of indexing schemes) All of the above representation languages are seen as candidate web-based XML languages for expressing and interrelating ontologies. Obrst and Liu provide a good insight into the relationship between Knowledge Representation and Ontological Engineering, and the emergence of web based languages (Orbst and Liu 2003).

Of course in a dynamic environment such as the one that we envisage, people will have freedom of choice in terms of how they structure their semantics and the representation language that they use to do so. We will term this the "ontology heterogeneity" problem. Visser et al focuses on ontology heterogeneity and classifies mismatches that may occur under headings of Conceptualization (on how concepts are classified) and Explication (on how concepts are specified). They then go on to discuss how easy/hard it is to deal with each type (Visser et al 1997). Several papers propose particular approaches or architectures for resolution

of ontological heterogeneity, mainly at design time. For example Corcho and Gomez-Perez propose a system that will semi-automatically integrate ontologies of different types in the e-business domain (Corcho and GomezPerez 2001). With respect to runtime solutions, Campbell et al outline a number of issues that typically need to be addressed (Campbell et al 1995):

- How to find a common frame of reference in both ontologies around which algorithms can traverse?
- Identifying what is or is not a match? What constitutes an exact match or a "good enough" match and how is this expressed?
- What online resources that might help identify equivalent terms (e.g. WordNet) should be used and when?

In summary, whereas the use of ontologies has been shown to overcome the semantic interoperability problem between parties that know they want to interact a priori, the use of ontologies to achieve semantically interoperable interactions between parties at runtime in a very dynamic situation is still in its infancy and must overcome a number of challenges.

3.2 Rationale for Semantic Matching Utility (SMU)

It has been argued that ontologies are "overkill" as a solution to support runtime interactions. However, Kim summarizes the situation nicely that in general XML DTD/Schemas will be good enough to support runtime interactions in systems where the pressing need is reduction of complexity and common agreement amongst various parties can be achieved, whilst if the pressing need is reduction of uncertainty with respect to common understanding of concepts used in the interchange, then ontologies are necessary (Kim 2002). The authors would argue that because of the kind of diversity and dynamics that is expected in ubiquitous computing environments, it is unlikely that there will be common agreement on one representation language or one set of terminology for the markup of all application elements. Therefore each element destined for use in a ubiquitous computing environment must encode knowledge about itself using explicit reference to an appropriate ontology. Thus, when such elements are integrated at design time or even at runtime for a ubiquitous computing system, several ontologies may need to be navigated and analyzed in order to dynamically bridge understanding between the constituent elements.

Mismatches between ontologies are the key type of problems that hinder the combined use of

independently developed ontologies in semantic interoperability solutions. "Language Level Mismatches" distinguishes mismatches of the language primitives that are used to specify the ontology. "Ontology Level Mismatches" distinguishes the differences in the way the domain of ontologies are modelled (Klein 2001). Current algorithms such as those used in Chimaera (McGuinness et al 2000) and PROMPT (Noy and Musen 2000 & 2001), solve Ontology Level mismatches, are heuristic in nature, and determine a match based on the content and structure of the ontologies (e.g. matching classes through their attributes, relationships, range values). They generally use lexically based matching algorithms, such as those proposed in (Hovy 1998), to support term matching. To date these algorithms have been deployed in tools that suggest matches to a human expert in a step by step manner where the intention is to undertake a complete integration/merging of the ontologies before the combination can be used.

In contrast, the authors propose a Semantic Matching Utility (SMU) that can be used at runtime in a ubiquitous computing environment with no human intervention at runtime, and minimal intervention at SMU configuration time.

Given a concept reference and a pointer to the Foreign Ontology (FO) where the concept is defined, the goal of the Semantic Matching Utility (SMU) is to return a matching concept reference as defined in the Target Ontology (TO) together with a semantic match confidence measure. The application that calls the SMU can then use the confidence measure to determine whether the match suggested is sufficient for its purposes. Leaving the decision to the calling application allows for the SMU to be independent of the calling applications. This independence is crucial given that it is envisaged that the SMU will be of general utility to a wide range of ubiquitous computing applications (e.g. task decomposition, service composition, context sharing).

3.3. The Experiment

It is generally accepted that the runtime comparison of concepts based on an arbitrary set of ontologies without prior full integration of the ontologies is impossible due to the range of mismatch problems that have been outlined earlier. However the authors propose that runtime comparison of concepts is feasible with a small amount of prior screening of the ontologies.

The first objective of the experiment was to examine whether a set of screening guidelines could be identified and applied to the ontologies, which would allow for runtime comparison across those ontologies without the need for full a priori

integration. Two sets of guidelines have been identified. The first set guides the human expert in determining whether the ontologies are amenable for processing by the SMU by examining the maturity, quality, degree of commonality and domain overlap of the ontologies. The second set guides the human expert in the identification of the minimum mappings necessary to allow for automated matching within the SMU to occur. For example these include naming convention mappings, relationship characterisation, and identification of common points of reference. The most critical of this latter set is the guideline that identifies the common points of reference between the ontologies. Rather than requiring a full integration of the ontologies to be specified, only a set of concepts which are asserted to be equivalent between the two ontologies need to be specified. This set of equivalent concepts are called "anchor points".

The second objective of the experiment was the implementation of an algorithm to support the runtime matching of concepts across the ontologies.

The implemented algorithm uses a combination of anchor point information and the "is-a" hierarchy within each ontology to narrow its search space and to identify candidate matches for a concept. The candidate matches are then examined in detail by comparing the properties and relationships of the candidate concept match with those of the original concept.

Two ontologies that model the domain of "research concepts" from the University of Manchester and Stanford University model were chosen for the initial experiment. The Stanford University ontology uses a XML RDF representation, comprising of 62 concepts with 88 slots. The University of Manchester ontology uses a XML OWL representation, comprising of 96 concepts with 120 slots.

As illustrated in Figure 3, navigation over the diversity of ontology representations is achieved through the use of XQueries (an XML Query language).

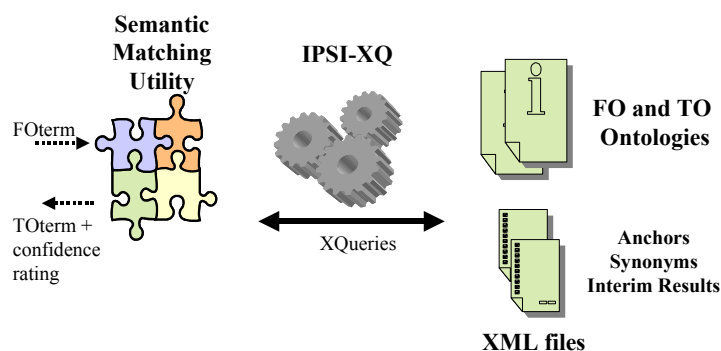


Figure 3: Experiment Architecture

3.4. Initial Results

At this time, the authors do not believe that semantic matching at runtime over an arbitrary set of ontologies is possible. However we have shown that a Semantic Matching Utility (SMU) is possible that can be used at runtime in a ubiquitous computing environment with no human intervention at runtime, and with minimal intervention at SMU configuration time.

In general it was found that the process of sourcing pairs of ontologies that focused on the same domain is possible but can be difficult and time consuming, for the following reasons:

A lot of good ontology modelling is proprietary in nature;

- Use of XML for ontology modelling is still in its early stages;
- XML based ontologies that are publically available (e.g. via DAML web site) vary greatly in quality and scope. Some ontologies published are "homework" assignments for example;
- In addition, a wide range of XML representations have been used (e.g. DAML, RDF) but the Protege tool that we are using to examine the ontologies in a common format has a very sensitive import facility that fails easily where poor XML syntax has been used.

It was also found that there is a need to have tools to support the exposition of certain functional and non-functional properties of an ontology, as these properties are not inherently provided in the specification of the ontology. Such properties include the coverage of the ontology, the breadth/width of the ontology, the connectedness of the concepts in the ontology etc.

The implementation behaved appropriately when the following test cases were applied:

A Foreign Concept that equated to an anchor point, returned the corresponding Target Ontology Concept immediately. Tested with 2 different concepts.

A Foreign Concept that was requested which did not have an anchor point as a descendent in its Ontology, returned no match. Tested with 2 different concepts.

A Foreign Concept which had an anchor point as a descendent (tested with 6 different concepts): found the anchor ancestor "isa" path within the Foreign Ontology and found the anchor ancestor "isa" path with the Target Ontology

In the case where there was an exact lexical match for the Foreign Concept in the Target Ontology path, the TO concept was returned.

In the case where there was a synonym match for the Foreign Concept in the Target Ontology path, the TO concept was returned

In the case where there was neither a exact lexical or synonym match, then no match was returned.

For each match found, an evaluation was undertaken to determine the confidence measure of the match by looking at number of attribute and relationship matches between the concepts. The match was returned with the confidence measure.

Finally it was found that the use of XQueries to enable the SMU to cope with the representation differences of the ontologies was very successful.

4. Conclusions and Future Work

The ubiquitous computing vision assumes that a wide range of application/services will interoperate seamlessly. Reaching global agreements on interoperability standards (which is the current assumption) will be impossible. It is the authors belief that application interoperability solutions are needed which are metadata & ontology based, loosely coupled and runtime based. In particular two major interoperability use cases for applications in ubiquitous computing were examined through experiments. The first experiment examined whether the Topic Map representation would be suitable as the basis of a Context Information System that would enable applications share context information. It has been concluded that Topic Maps show potential as the basis for a Context Information Service. The second experiment investigated whether the matching of concepts based on semantics at runtime would be feasible. The conclusion in this case is that semantic matching at runtime based on heterogeneous ontologies is feasible but requires some prescreening of the ontologies.

Further work is needed to validate the initial conclusions. Focus will be placed in the Context Information System on developing a query processing architecture that will cope with an environment where context is rapidly/frequently changing and is sometimes inaccurate or incomplete. Refinement of properties of ontologies, guidelines and tools will be undertaken for the Semantic Matching Utility.

References

- Campbell et al (1995), Campbell A, Chalupsky H, and Shapiro S. "Ontological mediation: An analysis." Unpublished manuscript.
- Ceri and Widom (1993), Ceri S., Widom J., "Managing Semantic Heterogeneity with Production Rules and Persistent Queues", Proceedings of the 19th VLDB Conference, Dublin, Ireland, pp. 108-119.
- Corcho and Gomez-Perez (2001), Corcho, O. and Gomez-Perez, A., "Solving Integration Problems of E-commerce Standards and Initiatives through Ontological Mappings", In: Proceedings of the Workshop on E-Business and Intelligent Web at the 17th International Joint Conference on Artificial Intelligence (IJCAI2001) , Seattle, USA, August 5, 2001.
- Cui et al (2002), Cui Z., Jones D. and O'Brien P., "Semantic B2B integration: issues in ontology-based approaches", ACM SIGMOD Record, vol 31 no 1, pages 43—48, ACM Press.
- DAML (2003), DARPA Markup Language, www.daml.org, Visited Apr 2003.
- Kim (2002), Kim H., Predicting how ontologies for the semantic web will evolve, Communications of the ACM, vol 45 no 2, pages 48-54, ACM Press.
- Klein (2001), Klein M. "Combining and relating ontologies: an analysis of problems and solutions". In IJCAI-2001 Workshop on Ontologies and Information Sharing, pages 53--62, Seattle, WA, 2001.
- Mitchell (2002), Mitchell K., A Survey of Context-Awareness, Available: <http://www.comp.lancs.ac.uk/~km/papers/ContextAwarenessSurvey.pdf> [2002, Nov. 27]
- Obrst and Liu (2003), Obrst L. and Liu H., "Knowledge Representation, Ontological Engineering and Topic Maps", Chapter 7, XML Topic Maps, Jack Park (ed.), Addison-Wesley.
- O'Sullivan and Lewis (2003), O'Sullivan D., Lewis D., Semantically Driven Service Interoperability for Pervasive Computing, Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access, San Diego, USA, 19 September 2003, ACM Press, pp 17-24
- Sheth (1998) Sheth A.P. , "Changing Focus on Interoperability in Information Systems: From System. Syntax, Structure to Semantics", M. F. Goodchild, M. J. Egenhofer, R. Fegeas, and C. A. Kottman (eds.) Interoperating Geographic Information Systems, Kluwer.
- Visser (1997), Visser, P.R.S., Jones, D.M., Bench-Capon, T.J.M., and Shave, M.J.R., "An analysis of ontology mismatches; heterogeneity versus interoperability." AAAI 1997 Spring Symposium on Ontology Engineering, 1997.
- W3C (2003), W3C Ontology Web Language (OWL), <http://www.w3.org/2001/sw/>, Visited Apr 2003.
- Wache (2001), H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner, "Ontology-based Integration of Information - A Survey of Existing Approaches," In: Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, 2001, Vol. pp. 108-117.
- Weiser (1991), Mark Weiser; The Computer for the 21st Century; Scientific American 265(3): 66-75, 1991.
- XTM (2003), XTM Authoring Group, XTM: XML Topic Maps 1.0. www.topicmaps.org. Visited Apr 2003

