We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

11

Ontological Knowledge Representation for Resolving Semantic Heterogeneity in Software Requirements Traceability

Namfon Assawamekin[1], Thanwadee Sunetnanta[2,3] and Charnyote Pluempitiwiriyawej[2,3]
[1]*School of Science, University of the Thai Chamber of Commerce*
[2]*Department of Computer Science, Faculty of Science, Mahidol University*
[3]*Faculty of Information and Communication Technology, Mahidol University*
THAILAND

1. Introduction

Heterogeneity in software requirements has regularly been discussed as a challenging problem, especially in the areas of requirements traceability. In practice, requirements are fundamentally expressed by customers in terms of natural language which inevitably inherits ambiguity. Pieces of requirements may be expressed in such a way that is best suited the view of an individual. Vocabularies and terminologies used in requirements expression therefore can be varied, habitually depending on customer roles, their background knowledge, perspectives and levels of understanding on system problems. Often, system analysts may capture requirements expressed by using different vocabularies or terminologies, yet conveying similar meaning.

Taking account of diversity of how software requirements can be expressed and who expresses the requirements, system analysts may use different techniques to elicit the requirements from customers. Requirements elicitation techniques range from typical ones like introspection, questionnaires, interviews, focus group and protocol analysis (Goguen & Linde, 1993) to modern one like scrum and agile requirements modeling (Paetsch et al., 2003). No matter which elicitation techniques are deployed, requirements from different customers often overlap, possibly are intertwined and inconsistent. As systems become more complex it becomes increasingly difficult for system analysts to resolve heterogeneity in software requirements so that the requirements can be verified and validated easily and effectively.

The impact of heterogeneity is even more crucial in distributed and collaborative software development environment since the heterogeneity is an inherent characteristic in such environment. With the advents of outsourcing and offshoring software development, software specifications can be collaboratively constructed by a team of developers in multiple sites, possibly with various development methods and tools. It is therefore important that system analysts must understand and be able to resolve the analogy and

poly-forms to requirements expression and representation to better communicate, check consistency and trace between pieces of requirements in a distributed manner.
In view of that, we propose to deploy ontology as a knowledge representation to intervene mutual "understanding" in requirements tracing process. By "ontological knowledge representation", we provide a basis modeling view for system analysts to express "a domain of discourse" for software requirements elicitation as well as the basic categories of requirements elements and their relationships. With our ontological knowledge representation, ontology matching is applied as a reasoning mechanism in automatically generating traceability relationships without restricting the freedom in expressing requirements differently. The relationships are identified by deriving semantic analogy of ontology concepts representing requirements elements. We will exemplify our ontological knowledge representation for software requirements traceability and compare our work to the applicability of other knowledge representations for the same purpose. Section 2 contains our literature reviews that lead to the development of ontological knowledge representation in this work. Section 3 presents our main idea of ontological knowledge representation for expressing software requirements. Section 4 elaborates how we can automate requirements traceability through ontology matching process. Section 5 concludes our contributions and further directions of our work.

2. Managing Semantic Heterogeneity in Software Development Life Cycle with Knowledge Representation

Software development is the processing of knowledge in a very focused way (Robillard, 1999). Knowledge acquisition is underlying cognitive process of software requirements gathering and elicitation to obtain information required to solve problems. Software models are forms of knowledge representation resulting from transitory construction of knowledge built up for presenting software solutions in software analysis process. Likewise, application programs are also forms of knowledge representation of software solutions that can be interpreted and processed by computer processors. Knowledge representation is therefore a key vehicle for organizing and structuring information in software development life cycle so that the information can be easily understood, systematically verified and validated by system developers and by the end users.
To manage semantic heterogeneity in software development, it is important to select knowledge representation that has sufficient expressive power as follows. Firstly, it is required that such knowledge representation should be able to recognize semantic differences in requirements expression and various software models. Secondly, it should preserve the meaning of the expression and the models, without restricting how the requirements are stated and the choices of software models that system developers want to use. In view of that, the basic constructs of the knowledge representation should be able to recognize type and instance definitions in requirements elements so that it can differentiate the meaning of requirements from its syntactic forms. Lastly, the knowledge representation should naturally support reasoning and inferences to resolve semantic heterogeneity arising in software development process.
There currently exists a collection of knowledge representations that are application to software development. Notable works are *RML*-the object-based knowledge representation for requirements specifications in (Borgida et al., 1985), *Cake*-the knowledge representation

and inference engine based on logics and plan calculus for software development in (Rich & Feldman, 1992) and *Telos*-the object-based knowledge representation with integrity constraints and deduction rules in (Mylopoulos et al., 1990). These works provide a solid proof on the application of knowledge representation to software development. However, each of these knowledge representations do not emphasize on their resolution in managing semantic heterogeneity that may arise. Although a certain degree of semantic inference may be derived from structuring mechanisms such as generalization, aggregation and classification as in Telos, the consistency of knowledge entered are verified through the an explicit constraint rules.

In our view, semantic heterogeneity can be both implicit and explicit in requirements expression. By semantic heterogeneity the meaning of words and understanding of concepts may differ or be interpreted differently from one community to another, regardless of syntax which refers to the structure or the schema by which the words or the concepts are represented. In view of that, it is not possible to explicitly define constraint rules or relationships that can be completely resolved semantic heterogeneity. Most of the times, semantic heterogeneity is implicit and is not known to the person who expresses such semantics of words or concepts.

Towards that view, we further explore the principle of ontology as explicit and formalized specifications of conceptualizations to extract and formalize the semantics. In the field of software engineering, there are many works that have applied and used ontology to different processes or phases in software development life cycle, starting from software requirements analysis (Kaiya & Saeki, 2005), cost estimation in project planning (Hamdan & Khatib, 2006) to re-engineering (Yang et al., 1999). There is also a particular set of work related to using ontology for multi-site distributed software development (Wongthongtham et al., 2005; Wongthongtham et al., 2008). From the literature, these works focus on using a single ontology to share a common understanding, manual construction of ontology and applying the ontology to specific application domains. In contrast to the above relevant works, our approach is concerned with ontology interoperability that does not force many stakeholders into a single ontology, but supports multiple ontologies for expressing multiperspective requirements artifacts. To be more precise, we aim to give various stakeholders with the freedom to communicate among each other based on their own defined ontologies. Additionally, our approach provides an automatic construction of multiple ontologies that is applicable to represent multiperspective requirements artifacts of any specific application domains. Next section will further describe ontology application in our work.

3. Ontological Approach to Knowledge Representation for Software Requirements

An ontology is an explicit formal specification of a shared conceptualization (Gruber, 1993; Borst, 1997; Studer et al., 1998). The ontology captures consensual knowledge, which is described in the terms of a formal model. In the ontology, a set of concept types and a set of formal axioms are explicitly defined with both human-readable and machine-readable text. Ontologies provide a common vocabulary of an area and define – with different levels of formality – the meaning of the terms and relations between them. Generally speaking, knowledge in the ontologies is formalized using five kinds of components: classes, relations,

functions, axioms and instances (Gruber, 1993). Classes in the ontology are usually organized in the taxonomies.

The ontology is widely used as an important component in many areas, such as knowledge management (Jurisica et al., 2004), electronic commerce (Hepp et al., 2007), distributed systems (Haase et al., 2008), information retrieval systems (Jung, 2009) and in new emerging fields like the Semantic Web. Ontology can prove very useful for a community as a way of structuring and defining the meaning of the metadata terms that are currently being collected and standardized. Using ontologies, tomorrow's applications can be "intelligent", in the sense that they can more accurately work at the human conceptual level.

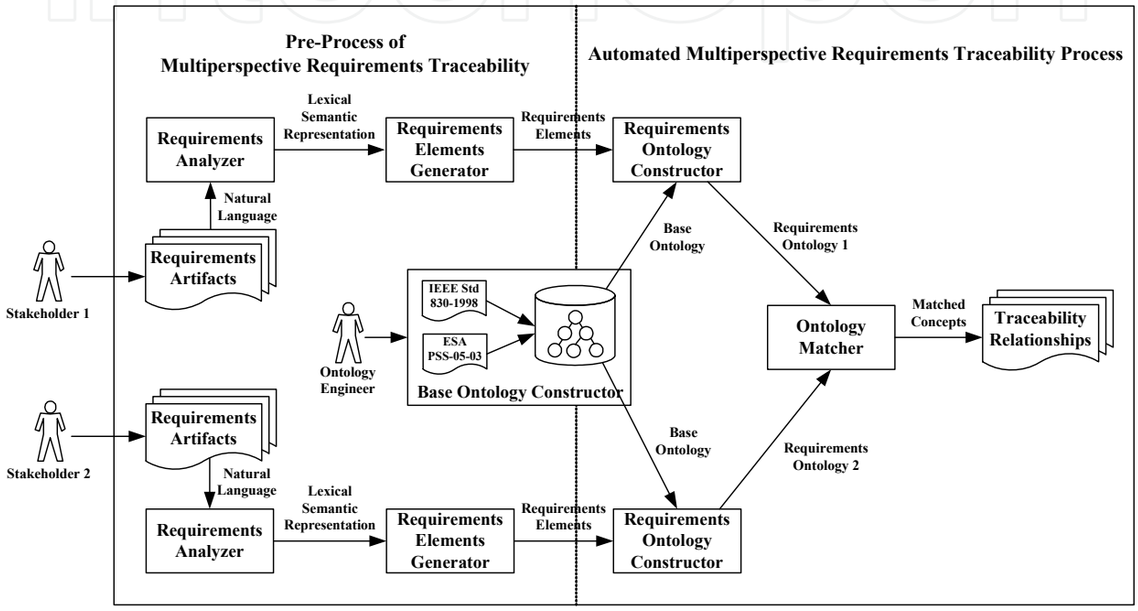


Fig. 1. Multiperspective requirements traceability (MUPRET) framework

We apply ontology concept to our multipersepctive requirements traceability (MUPRET) framework which merges the natural language processing (NLP) techniques, rule-based approaches and ontology concepts in order to resolve the heterogeneity in multiperspective requirements artifacts. Figure 1 illustrates our MUPRET framework containing five main modules: requirements analyzer (RA), requirements elements generator (REG), base ontology constructor (BOC), requirements ontology constructor (ROC) and ontology matcher (OM). The details of all modules deployed in the MUPRET framework are presented in depth elsewhere in our previous papers (Assawamekin et al., 2008a; Assawamekin et al., 2008b). The five main modules can be briefly explained as follows:

1. The RA module obtains a set of requirements artifacts represented in terms of natural language or plain English text. It uses the NLP techniques to syntactically analyze these artifacts and generate lexical semantic representation as the output.
2. The REG module utilizes rule-based approaches to automatically extract requirements elements from requirements artifacts.
3. The BOC module constructs a base ontology to classify requirements types of requirements artifacts in the domain of software requirements.

4. The ROC module attaches requirements elements into the base ontology to automatically construct requirements ontology of each stakeholder as a common representation for knowledge interchange purposes.
5. The OM module applies ontology matching technique in order to automatically generate traceability relationships when a match is found in the requirements ontologies.

In summary, we propose our MUPRET framework which deploys *ontology* as a knowledge management mechanism to intervene mutual "understanding" without restricting the freedom in expressing requirements differently. *Ontology matching* is applied as a reasoning mechanism in automatically generating traceability relationships. The relationships are identified by deriving semantic analogy of ontology concepts representing requirements elements.

4. Matching Ontologies for Requirements Traceability

As briefly discussed in the introductory part, large-scaled software development inevitably involves a group of stakeholders, each of which may express their requirements differently in their own terminology and representation depending on their perspectives or perceptions of their shared problems. Such requirements result in *multiperspective requirements artifacts*. These artifacts may be enormous, complicated, ambiguous, incomplete, redundant and inconsistent. However, they must be traced, verified and merged in order to achieve a common goal of the development. Moreover, requirements artifacts are frequently subject to changes. Planning, controlling and implementation of requirements changes can be tedious, time-consuming and cost-intensive. Determining of effects caused by requirements changes on software systems is based on *requirements traceability* (Gotel & Finkelstein, 1994).

The traceability of multiperspective requirements artifacts has regularly been discussed as a challenging problem, particularly in the requirements change management (Grunbacher et al., 2004). The heterogeneity of multiperspective requirements artifacts makes it difficult to perform tracing, verification and merging of the requirements. More specifically, it can be very problematic when multiperspective requirements artifacts are expressed with synonyms (i.e. different terminologies representing the same concept) and homonyms (i.e. the same term representing different concepts) and various stakeholders want to share these artifacts to each other. In this situation, ontology can play an essential role in communication among diverse stakeholders in the course of an integrating system.

To be able to achieve our goal, this section presents ontology matching process executed in the following four steps to reason on traceability that arises between requirements.

Step 1: Compute concepts of labels, which denote the set of concepts that one would classify under a *label* it encodes.

Step 2: Compute concepts of nodes, which denote the set of concepts that one would classify under a node, given that it has a certain *label* and *position* in the graph. For object concepts, the logical formula for a concept at node is defined as a conjunction of concepts of labels located in the path from the given node to the root. For relationship concepts, the concept at node is identified as a conjunction of domain, range and relationship concepts. For process concepts, the concept at node is defined as a conjunction of actor, input, output and process concepts.

Step 3: Compute the relations between concepts of labels, called *element matching*. In this work, we contribute *a base ontology* to define the types of concepts. If two concepts have

different types, the relation between two concepts is mismatch. We also use external resources (i.e., domain knowledge and WordNet (Miller, 1990; Miller, 1995)) and string matching techniques (i.e., prefix, suffix, edit distance and n-gram) with threshold 0.85. Lexical relations provided by WordNet are converted into semantic relations according to the rules as shown in Table 1.

Step 4: Compute the relations between concepts of nodes, called *concept matching*. Each concept is converted into a propositional validity problem. Semantic relations are translated into propositional connectives using the rules described in Table 1.

Lexical relations	Semantic relations	Propositional logic	Translation of formula into conjunctive normal form
Synonym	$a = b$	$a \leftrightarrow b$	$axioms \wedge \forall(context_1 \wedge \neg context_2)$ $axioms \wedge \forall(\neg context_1 \wedge context_2)$
Hyponym or meronym	$a \subseteq b$	$a \rightarrow b$	$axioms \wedge \forall(context_1 \wedge \neg context_2)$
Hypernym or holonym	$a \supseteq b$	$b \rightarrow a$	$axioms \wedge \forall(\neg context_1 \wedge context_2)$
Antonym	$a \perp b$	$\neg(a \wedge b)$	$axioms \wedge \forall(context_1 \wedge context_2)$
	$a \cap b$	$(a \wedge b) \vee (a \wedge \neg b) \vee (\neg a \wedge b)$	$axioms \wedge \exists(\neg context_1 \vee \neg context_2) \wedge \exists(\neg context_1 \vee context_2) \wedge \exists(context_1 \vee \neg context_2)$

Table 1. The relationships among lexical relations, semantic relations and propositional formula

The criterion for determining whether a relation holds between concepts is the fact that it is entailed by the premises. Thus, we have to prove that this formula *(axioms)* → *rel(context₁, context₂)* is valid. A propositional formula is valid iff its negation is unsatisfiable. A SAT solver (Berre, 2006) run on the formula fails.

We use types of overlap relations defined in (Spanoudakis et al., 1999) to generate traceability relationships in our work. The traceability relationships can be generated when a match is found in the requirements ontologies. Thus, the semantic relations will be mapped to traceability relationships as shown in Table 2.

Semantic relations	Traceability relationships
Equivalence (=)	overlapTotally (=)
More or less general ($\supseteq$, $\subseteq$)	overlapInclusively ($\supseteq$, $\subseteq$)
Mismatch ($\perp$)	noOverlap ($\perp$)
Overlapping ($\cap$)	overlapPartially ($\cap$)

Table 2. Conversion of semantic relations into traceability relationships

The distinction and implication among different types of traceability relationships is important not only because these relationships have different impact on the requirements traceability status of two requirements artifacts but also because the corrections of requirements changes occurring due to each of these types of traceability relationships might not be the same. In our work, we order traceability relationships as they have been

listed, according to their binding strength, from the strongest to the weakest. The more general and less general have the same binding strength. Hence, *overlapTotally* is the strongest relationship since the sets of source concept have exactly the same as the sets of target concept. The source and target concepts are *overlapInclusively* if one of the designated sets is proper subset of the other. Both source and target concepts are *overlapPartially* if their designated sets have both concepts in common and non-common concepts. More importantly, we discard *noOverlap* relationship which is the weakest relationship in this work because there is no effect on multiperspective requirements artifacts changes.

As a prototype of the processes in the MUPRET framework, we have developed the MUPRET tool which is a Java-based tool with Prolog and WordNet-based semantic inference engine. This tool aims to support our framework and to demonstrate its feasibility for distributed, collaborative and multiperspective software development environment. The details of MUPRET tool are presented in depth elsewhere in our paper (Assawamekin et al., 2009). This tool runs on PCs running MS-Windows as a standalone environment. Our design of the MUPRET tool primarily focuses on demonstrating "proof-of-concept" rather than on optimizing technique used in the framework. The aim of our approach is to build a generic support environment for the MUPRET framework. This approach is constructed with specialized tools and techniques that either demonstrate the feasibility of the approach or address a particular requirements traceability issue.

The MUPRET tool facilitates the automatic extraction and construction of requirements elements of an individual stakeholder into a so-called requirements ontology. As a result, multiperspective requirements artifacts of different stakeholders are captured in a common taxonomy imposed by the sharing base of requirements ontology. The tool then automatically generates traceability links by matching requirements ontologies.

To demonstrate how to use the MUPRET tool, we will illustrate how to generate traceability relationships via two different requirements artifacts with respect to two different perspectives. These two requirements artifacts describe parts of a hospital information system. More specifically, they describe a doctor investigation system (DIS) and an in-patient registration system (IPRS). These requirements artifacts are written in format of plain English text as follows.

Requirements 1: (DIS perspective)

Each patient has a unique hospital number (HN) and a name. A patient is admitted by a doctor. Nurses and doctors are considered as staffs. A nurse has a name. The nurse's name consists of a first name, an initial and a last name. A doctor is identified by an identification number and a name.

Requirements 2: (IPRS perspective)

Physicians and nurses are staffs. Staffs have an ID, a name and an address. A surgeon is a physician.

Both requirements are presented as a source (DIS) and a target (IPRS) in our *MUPRET browser*. After both requirements are passed to the RA and REG modules, the ROC module will attach requirements elements into the base ontology. Accordingly, the DIS and IPRS requirements ontology are automatically constructed as depicted in Figures 2 and 3 respectively.

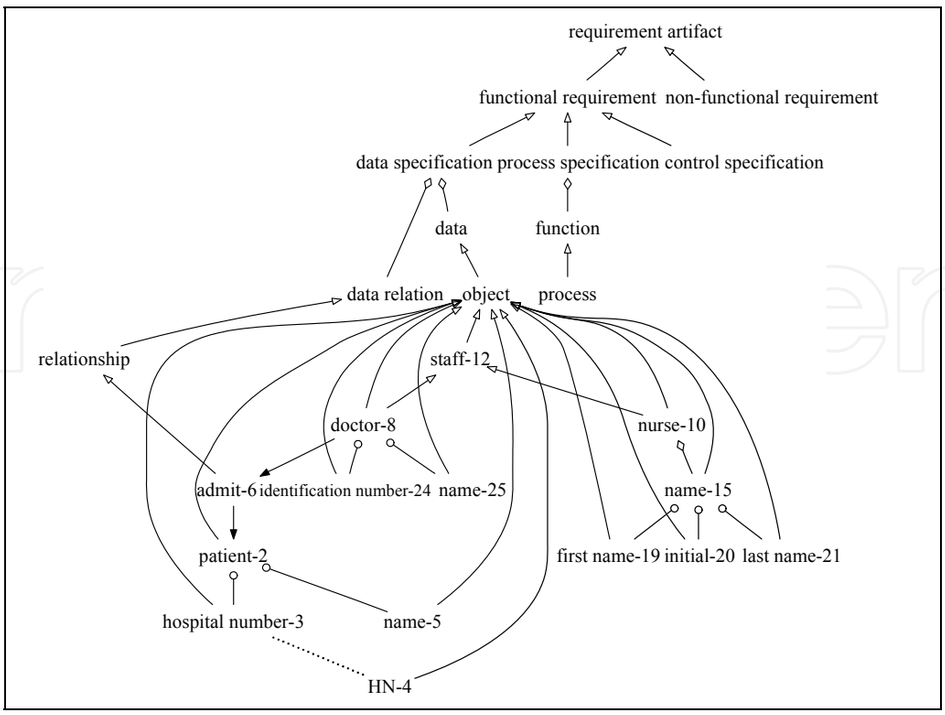


Fig. 2. Doctor investigation system (DIS) requirements ontology

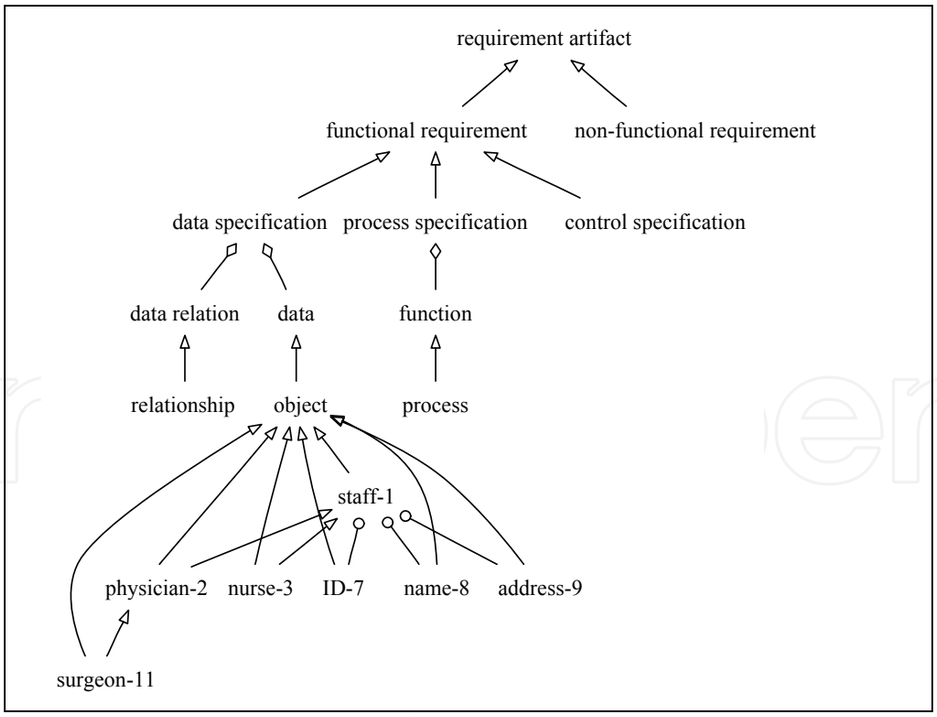


Fig. 3. In-patient registration system (IPRS) requirements ontology

A part of traceability relationships between DIS and IPRS requirements artifacts can be expressed in the first-order logic (FOL) or predicate terms for machine-readable text as shown below.

overlapTotally(Requirements 1/S2T7, S3T3, S6T2/doctor | Requirements 2/S1T1, S3T5/physician)
overlapInclusively(Requirements 2/S2T4/ID | Requirements 1/S2T7, S3T3, S6T2/doctor)
overlapInclusively(Requirements 2/S2T7/name | Requirements 1/S2T7, S3T3, S6T2/doctor)
overlapInclusively(Requirements 2/S2T10/address | Requirements 1/S2T7, S3T3, S6T2/doctor)
overlapInclusively(Requirements 2/S3T2/surgeon | Requirements 1/S2T7, S3T3, S6T2/doctor)
overlapInclusively(Requirements 1/S3T1, S4T2, S5T2/nurse | Requirements 2/S1T5, S2T1/staff)
overlapPartially(Requirements 1/S2T7, S3T3, S6T2/doctor | Requirements 2/S1T3/nurse)

From the example, overlapTotally(Requirements 1/S2T7, S3T3, S6T2/doctor | Requirements 2/S1T1, S3T5/physician) means that *doctor* of sentence 2 token 7, sentence 3 token 3 and sentence 6 token 2 in the Requirements 1 (DIS requirements artifacts) overlaps totally with *physician* of sentence 1 token 1 and sentence 3 token 5 in the Requirements 2 (IPRS requirements artifacts). Using the Figures 2 and 3, trying to prove that $doctor_1$ in DIS requirements ontology is less general than $physician_2$ in IPRS requirements ontology, requires constructing the following formula.

$$((staff_1 \leftrightarrow staff_2) \wedge (doctor_1 \leftrightarrow physician_2)) \wedge (staff_1 \wedge doctor_1) \wedge \neg(staff_2 \wedge physician_2)$$

The above formula turns out to be unsatisfiable, and therefore, the less general relation holds. It is noticeable that if we test for the more general relation between the same pair of concepts, the corresponding formula would be also unsatisfiable. As a result, the final relation for the given pair of concepts is the equivalence.
Equally, an example screen of traceability relationships can be depicted in Figure 4 for human-readable text and user-friendly. The totally, (superset or subset) inclusively and partially overlapped target can be represented with green, red, cyan and yellow color respectively while the grey color means the source of requirements. As seen as an example in this figure, *doctor* in the Requirements 1 (DIS requirements artifacts) overlaps totally with *physician*, overlaps inclusively (superset) with *ID*, *name*, *address* and *surgeon*, overlaps inclusively (subset) with *staff* as well as overlaps partially with *nurse* in the Requirements 2 (IPRS requirements artifacts).

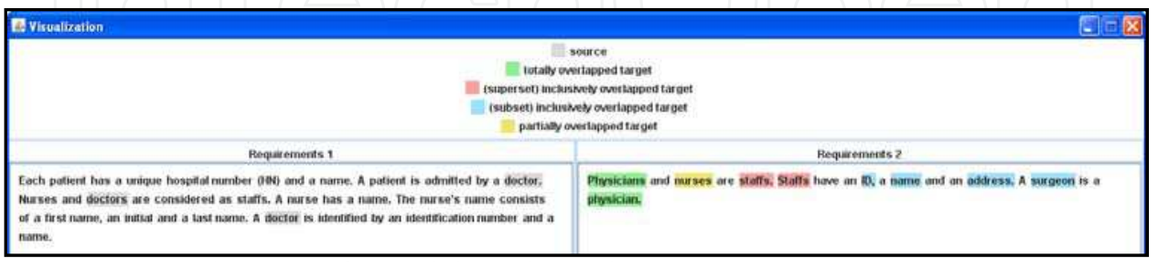


Fig. 4. An example screen of traceability relationships

Let us consider again the example of Figure 4, the overlap between *doctor* in the Requirements 1 and *physician* in the Requirements 2 is total. In the view of traceability, if *doctor* in the Requirements 1 is changed then the modification of *physician* in the

Requirements 2 must be needed. On the other hand, if *doctor* in the Requirements 1 is changed then *staff* in the Requirements 2 may be modified since *doctor* in the Requirements 1 overlaps inclusively (subset) with *staff* in the Requirements 2. Additionally, if *doctor* in the Requirements 1 is modified then the modification of *nurse* in the Requirements 2 may be needed with respect to overlap partially relationship. In contrast, if *patient* in the Requirements 1 is changed then there is no modification needed for *physician* in the Requirements 2 due to no overlap relationship.

To sum up, the MUPRET tool automatically constructs requirements ontologies from multiperspective requirements artifacts with the aim of generating traceability relationships. The ontology matching technique is executed without any user interaction in order to achieve this goal. Suppose that the relations between element matching are correct, the relations between concept matching can generate the precise semantic relations. In view of that, traceability relationships are also accurate.

5. Conclusions and Ongoing Work

This chapter points out the semantic heterogeneity problems found in multiperspective requirements artifacts and introduces the ontological knowledge representation to help resolve such problems. The resolution is described via our MUPRET framework and tool. Our MUPRET framework merges the NLP techniques, rule-based approaches and ontology concepts to automatically generate traceability relationships of multiperspective requirements artifacts, which can be applied to any software requirements domain. In MUPRET, the base ontology representing the fundamental concepts is defined and used to classify requirements types of requirements artifacts. Regarding the base ontology, multiple requirements ontologies can be developed and virtually integrated through ontology matching process. The result of the ontology matching is a set of traceability relationships of software requirements.

Although the current stage of the MUPRET framework and tool emphasizes on tracing multiperspectives in requirements analysis phase and focuses on requirements that are expressed in terms of natural language or plain English text. It is possible to extend MUPRET to cover multiperspective software artifacts expressed in terms of typical analysis models. This can be done by adding semantics of those model elements to the base of the MUPRET's requirements ontology. In addition, we also aim at exploring further how to apply our MUPRET to support traceability throughout a complete software development process.

6. References

Assawamekin, N.; Sunetnanta, T. & Pluempitiwiriyawej, C. (2008a). Automated Multiperspective Requirements Traceability Using Ontology Matching Technique, *Proceedings of the Twentieth International Conference on Software Engineering and Knowledge Engineering (SEKE 2008)*, pp. 460-465, Hotel Sofitel, Redwood City, San Francisco Bay, C.A., U.S.A., July 1-3, 2008

Assawamekin, N.; Sunetnanta, T. & Pluempitiwiriyawej, C. (2008b). Resolving Multiperspective Requirements Traceability Through Ontology Integration, *Proceedings of the Second IEEE International Conference on Semantic Computing (ICSC 2008)*, pp. 362-369, Santa Clara Marriot Hotel, Santa Clara, C.A., U.S.A., August 4-7, 2008

Assawamekin, N.; Sunetnanta, T. & Pluempitiwiriyawej, C. (2009). MUPRET: An Ontology-Driven Traceability Tool for Multiperspective Requirements Artifacts, *Proceedings of the 8th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2009)*, pp. 943-948, Pine City Hotel, Shanghai, China, June 1-3, 2009

Berre, D.L. (2006). A Satisfiability Library for Java. Available at http://www.sat4j.org, June 15, 2006

Borgida, A.; Greenspan, S. & Mylopoulos, J. (1985). Knowledge Representation as the Basis for Requirements Specifications. *IEEE Computer*, Vol. 18, No. 4, pp. 82-91

Borst, W.N. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*, Doctoral Dissertation, Enschede, NL-Centre for Telematics and Information Technology, University of Tweenty

Goguen, J.A. & Linde, C. (1993). Techniques for Requirements Elicitation, *Proceedings of IEEE International Symposium on Requirements Engineering*, pp. 152-164, January 4-6, 1993

Gotel, O.C.Z. & Finkelstein, A.C.W. (1994). An Analysis of the Requirements Traceability Problem, *Proceedings of the 1st International Conference on Requirements Engineering (ICRE 1994)*, pp. 94-101, Colorado Springs, Colorado, U.S.A., April 18-22, 1994

Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, Vol. 5, No. 2, pp. 199-220

Grunbacher, P.; Egyed, A. & Medvidovic, N. (2004). Reconciling Software Requirements and Architectures with Intermediate Models. *Journal for Software and System Modeling (SoSyM)*, Vol. 3, No. 3, pp. 235-253

Haase, P.; Siebes, R. & Harmelen, F.v. (2008). Expertise-Based Peer Selection in Peer-to-Peer Networks. *Knowledge and Information Systems*, Vol. 15, No. 1, pp. 75-107

Hamdan, K. & Khatib, H.E. (2006). A Software Cost Ontology System for Assisting Estimation of Software Project Effort for Use with Case-Based Reasoning, *Innovations in Information Technology*, pp. 1-5

Hepp, M.; Leukel, J. & Schmitz, V. (2007). A Quantitative Analysis of Product Categorization Standards: Content, Coverage, and Maintenance of eCl@ss, UNSPSC, eOTD, and the RosettaNet Technical Dictionary. *Knowledge and Information Systems*, Vol. 13, No. 1, pp. 77-114

Jung, J.J. (2009). Consensus-Based Evaluation framework for Distributed Information Retrieval Systems. *Knowledge and Information Systems*, Vol. 18, No. 2, pp. 199-211

Jurisica, I.; Mylopoulos, J. & Yu, E. (2004). Ontologies for Knowledge Management: An Information Systems Perspective. *Knowledge and Information Systems*, Vol. 6, No. 4, pp. 380-401

Kaiya, H. & Saeki, M. (2005). Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach, *Proceedings of the Fifth International Conference on Quality Software (QSIC 2005)*, pp. 223-230

Miller, G.A. (1990). WordNet: An On-line Lexical Database. *International Journal of Lexicography*, Vol. 3, No. 4, pp. 235-312

Miller, G.A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, Vol. 38, No. 11, pp. 39-41

Mylopoulos, J.; Borgida, A.; Jarke, M. & Koubarakis, M. (1990). Telos: Representing Knowledge about Information Systems. *ACM Transactions on Information Systems (TOIS)*, Vol. 8, No. 4, pp. 325-362

Paetsch, F.; Eberlein, A. & Maurer, F. (2003). Requirements Engineering and Agile Software Development, *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2003)*, pp. 308–313, June 9-11, 2003

Rich, C. & Feldman, Y.A. (1992). Seven Layers of Knowledge Representation and Reasoning in Support of Software Development. *IEEE Transactions on Software Engineering*, Vol. 18, No. 6, pp. 451-469

Robillard, P.N. (1999). The Role of Knowledge in Software Development. *Communications of the ACM*, Vol. 42, No. 1, pp. 87-92

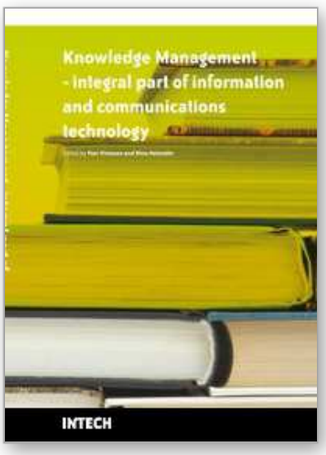
Spanoudakis, G.; Finkelstein, A. & Till, D. (1999). Overlaps in Requirements Engineering. *Automated Software Engineering*, Vol. 6, No. 2, pp. 171-198

Studer, R.; Benjamins, V.R. & Fensel, D. (1998). Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*, Vol. 25, pp. 161-197

Wongthongtham, P.; Chang, E. & Cheah, C. (2005). Software Engineering Sub-Ontology for Specific Software Development, *Proceedings of the 2005 29th Annual IEEE/NASA Software Engineering Workshop (SEW 2005)*, pp. 27-33

Wongthongtham, P.; Kasisopha, N.; Chang, E. & Dillon, T. (2008). A Software Engineering Ontology as Software Engineering Knowledge Representation, *Proceedings of the Third International Conference on Convergence and Hybrid Information Technology (ICCIT 2008)*, pp. 668-675, November 11-13, 2008

Yang, H.; Cui, Z. & O'Brien, P. (1999). Extracting Ontologies from Legacy Systems for Understanding and Re-Engineering, *Proceedings of the Twenty-Third Annual International Conference on Computer Software and Applications*, pp. 21-26



Knowledge Management

Edited by Pasi Virtanen and Nina Helander

ISBN 978-953-7619-94-7
Hard cover, 272 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

This book is a compilation of writings handpicked in esteemed scientific conferences that present the variety of ways to approach this multifaceted phenomenon. In this book, knowledge management is seen as an integral part of information and communications technology (ICT). The topic is first approached from the more general perspective, starting with discussing knowledge management's role as a medium towards increasing productivity in organizations. In the starting chapters of the book, the duality between technology and humans is also taken into account. In the following chapters, one may see the essence and multifaceted nature of knowledge management through branch-specific observations and studies. Towards the end of the book the ontological side of knowledge management is illuminated. The book ends with two special applications of knowledge management.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Namfon Assawamekin, Thanwadee Sunetnanta and Charnyote Pluempitiwiriyawej (2010). Ontological Knowledge Representation for Resolving Semantic Heterogeneity in Software Requirements Traceability, Knowledge Management, Pasi Virtanen and Nina Helander (Ed.), ISBN: 978-953-7619-94-7, InTech, Available from: http://www.intechopen.com/books/knowledge-management/ontological-knowledge-representation-for-resolving-semantic-heterogeneity-in-software-requirements-t

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen