Resolving Semantic Heterogeneity in Multiperspective Requirements Traceability Using Ontology Matching

Namfon Assawamekin
School of Science and Technology, University of the Thai Chamber of Commerce, Bangkok 10400, THAILAND
namfon_ass@utcc.ac.th, namfona@gmail.com

Abstract

In large-scaled software development projects, different stakeholders may deal with different pieces of software requirements depending on their perspectives or perceptions of their shared problems. Each of the users may define his/her requirements from his/her own point of view using different terminologies. System analysts may express or model the artifacts of the system by using different representation styles and methodologies. However, a group of stakeholders often needs to interact, collaborate or trace requirements among the group in order to achieve common goals in their development process.

To resolve the semantic heterogeneity problems arising from requirements tracing among various stakeholders, the enhancement of multiperspective requirements traceability (MUPRET) framework is proposed. The main objective of this paper is to concentrate on tracing multiperspective requirements artifacts represented in the form of the textual requirements and the requirements model, specifically the entity relationship diagram. The requirements ontology is used as a knowledge representation to represent multiperspective requirements artifacts of an individual stakeholder. Ontology matching is applied as a reasoning mechanism in automatically generating traceability relationships without restricting the freedom in expressing requirements differently.

Keywords: Requirements Traceability, Multiperspective Software Development, Ontology, Semantic Heterogeneity, Knowledge Representation

1. Introduction

Currently, multiperspective software development is essentially a collaborative effort among various stakeholders of a software project, especially for those directly involved in the software development process. The requirements artifacts (e.g., requirements specifications, analysis and design models, design specifications, test cases and code modules) are expressed by diverse system stakeholders according to their perspectives or perceptions of their backgrounds, skills, knowledge, expertise, environment, roles, responsibilities and commitments on a problem domain. However, a variety of stakeholders need to interoperate, collaborate or work together towards the common goal to produce a software solution. In addition, the requirements are frequently subject to changes during a software development process. Planning, controlling and implementation of requirements changes can be difficult, expensive, tedious, time-consuming and cost-intensive. The determining of effects caused by requirements changes on software systems is based on requirements traceability [1].

Semantic heterogeneity in software requirements has regularly been discussed as a challenging problem, especially in the areas of requirements traceability. Taking account of diversity of requirements modeling, system analysts may use different techniques to elicit the requirements from the customers. Consequently, pieces of requirements may be expressed in such a way that is best suited the view of an individual analyst. Moreover, they may be expressed by using different terminologies which convey similar meaning. The impact of semantic heterogeneity is even more crucial in distributed and collaborative software development environment since heterogeneity is an inherent characteristic in such development environment. It is important that various system stakeholders must understand and be able to resolve the analogy and poly-forms to requirements representation to better communicate, check consistency and trace between pieces of requirements in a distributed manner.
As part of our attempt in resolving the requirements traceability problems of multiperspective requirements artifacts, we have investigated some existing requirements traceability approaches and tools to manage software requirements and architecture. A number of research efforts have been carried out to tackle the requirements traceability problems. However, there are few works regarding the traceability of multiperspective requirements artifacts. More importantly, there are some problems as summarized below when we apply the existing approaches and tools for requirements traceability to a multiperspective context.

Firstly, the granularity of traceability is typically too coarse-grained or medium level. From the literature, various approaches focus on either coarse-grained level (e.g., commercial tools [2], REMAP [3], RETH [4], RADIX [5] and TOOR [6]) or medium level (e.g. EBT [7]). To resolve heterogeneity for tracing, verifying and merging multiperspective requirements artifacts effectively, the relationships between the artifacts should be identified at fine-grained level in terms of semantic links of requirements elements expressed differently in various perspectives.

Secondly, automated solutions to requirements traceability face a difficult challenge due to the need to handle the overlapping among multiperspective requirements artifacts, as well as the large number of traceability relationships existing between them. Most of existing approaches generate traceability links manually (e.g., commercial tools [2], REMAP [3], RETH [4], RADIX [5] and TOOR [6]) or semi-automatically (e.g., EBT [7], VBRT [8] and Trace Analyzer [9]). Defining large number of traceability relationships manually can be a tedious, labor-intensive and time-consuming task.

Lastly, although there are previous attempts in applying various techniques to automatically establish traceability links of software requirements, some problems still remain. For example, the works which apply information retrieval (IR) approach to requirements traceability, such as the work in [10], [11], [12], Poirot:TraceMaker [13, 14], RETRO [15, 16] and Trace Retrieval [17], typically ignore structural and semantic information that can be found in the requirements, therefore limiting both their precision and applicability. Spanoudakis et al. [18] apply rule-based approach to automatically generate and maintain traceability links between requirements artifacts but the traceability rules specify ways of matching syntactically related terms in the textual parts of a requirements statement or use case with related elements in the object model. Although those existing approaches as reviewed above tackle traceability relationships at fine-grained level, they handle the requirements artifacts defined or expressed by various stakeholders by using same set of vocabularies in the requirements descriptions. Accordingly, the openness and shared vocabulary restrict the freedom in expressing multiperspectives.

The rest of this paper is organized as follows. Section 2 presents some existing research related to our work. In Section 3, the enhancement of multiperspective requirements traceability (MUPRET) framework is proposed. Section 4 illustrates how to match the concepts between two requirements ontologies by using ontology matching technique for automatically generating traceability relationships. Finally, we conclude the paper with the discussion of our contributions and ongoing work in Section 5.
there is no intermediate result for aligning or matching the concepts between the ontologies because the source ontology does not remain after the merge process.

In the field of software engineering, there are a number of works that have applied and used ontology to different processes or phases in software development life cycle, starting from software requirements analysis [37], cost estimation in project planning [38] to re-engineering [39]. Wongthongtham et al. [40] apply ontology to multi-site distributed software development. Other attempts have been done in using ontology-based approach for traceability [41], [42], [43] and query purposes [44]. Nonetheless, those existing works focus on using a single ontology to share a common understanding, manual construction of ontology and applying the ontology to specific domain applications. Defining, verifying and merging large number of ontology concepts and their relationships manually by using some existing tools as reviewed above can be a tedious, labor-intensive and time-consuming task. In view of that, our work aims at applying ontology without requiring manual tasks in ontology construction and integration process. How this can be achieved will be elaborated in the next section which explains the details of the enhancement MUPRET framework.

3. Our Approach

The MUPRET framework was originally contributed and described in our previous work [45] with the aim of tracing multiperspective requirements artifacts represented in terms of natural language or plain English text. To resolve the semantic heterogeneity problems found in multiperspective requirements artifacts, we apply the natural language (NLP) techniques, the rule-based approach and the ontology concepts to trace between the textual requirements and the entity relationship (ER) diagram. This section aims at presenting an architectural view of the enhancement MUPRET framework. Figure 1 shows the enhancement of MUPRET framework which contains six main modules: a requirements analyzer (RA), a requirements elements generator (REG), a model elements generator (MEG), a base ontology constructor (BOC), a requirements ontology constructor (ROC) and an ontology matcher (OM). The details of these modules in the framework can be summarized as follows:

1. The RA 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 utilizes the rule-based approach to automatically extract requirements elements from the requirements artifacts.
3. The MEG automatically extracts model elements from the requirements models.
4. The BOC constructs a base ontology to classify requirements types of requirements artifacts in the domain of software requirements.
5. The ROC attaches requirements elements or model elements into the base ontology to automatically construct requirements ontology of each stakeholder as a common representation for knowledge interchange purposes.
6. The OM applies the ontology matching technique in order to automatically generate traceability relationships when a match is found in the requirements ontologies.

This paper rigorously focuses on constructing the requirements ontology as described in Section 3.1. Section 3.2 presents the ontology matching technique to automatically generate traceability relationships. These details will be explained in the following subsections respectively.

3.1. Automatic requirements ontology construction

To automatically construct requirements ontology of each stakeholder, the requirements elements and model elements are attached into the base ontology. The ontology constructed by our MUPRET framework is internally represented in terms of first-order logic (FOL). For readability purpose, we also provide a graphical tree view for the visualization of the ontology by using Graph visualization (Graphviz) software [46, 47] developed at AT&T research labs. We extend the base ontology to trace between requirements text and the ER diagram. As a result, the model elements of the ER diagram are added into the base ontology as illustrated in Figure 2. The graphical notations of the visualization view are defined in Figure 3. Note that some notations are enhanced from unified modeling language (UML)-like but they have more specific meaning used in this work. These relationships defined in MUPRET are used to derive traceability relationships from ontology matching.

![Figure 2. A graphical representation of a base ontology](image)

3.2. Ontology matching for automatic traceability relationships generation

We make use of ontology matching technique for automatically generating traceability relationships between two requirements ontologies. It takes two requirements ontologies and produces the correspondences (i.e., equivalence (=), more general (⊇), less general (⊆), mismatch (⊥) and overlapping (∩)) between the concepts of ontologies that semantically correspond to each other. When none of the relations holds, the special idk (I don’t know) relation is returned.
Before matching the concepts between two requirements ontologies takes place, we apply some rules to automatically discover implicit relationships from existing explicit relationships found in the ontologies. The implicit relationships are produced by using the following inheritance, transitive and pseudo-transitive rules detailed in [45].

Our ontology matching process is executed in the following four steps.

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 [48, 49], [50], [51], [52]) and string matching techniques (i.e., prefix, suffix, edit distance and n-gram) with threshold 0.85.

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 as described in Algorithm 1. We extend the overlapping relation in this algorithm.

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) $\rightarrow$ rel(context₁, context₂) is valid. A propositional formula is valid iff its negation is unsatisfiable. A SAT solver [53] run on the formula fails.

The pseudo code of a basic solution for the node matching algorithm is provided in Algorithm 1. In lines 110 and 140, the nodeMatch function constructs the formulas for testing the less general and more general relations. In lines 120 and 150, it converts them into conjunctive normal form (CNF), while in lines 130 and 160, it checks the formulas in the CNF for unsatisfiability. If both relations hold, then the equivalence relation is returned (line 180). Otherwise, the less general relation is returned (line 210) or the more general relation is returned (line 240). Finally, the same procedure is repeated for testing the disjointness and overlap relations. If all the tests fail, the idk relation is returned (line 380).
Algorithm 1. The node matching algorithm

100. String nodeMatch(String axioms, context1, context2)
110. String formula = And(axioms, context1, Not(context2));
120. String formulaInCNF = convertToCNF(formula);
130. boolean isLG = isUnsatisfiable(formulaInCNF);
140. formula = And(axioms, Not(context1), context2);
150. formulaInCNF = convertToCNF(formula);
160. boolean isMG = isUnsatisfiable(formulaInCNF);
170. if (isMG && isLG) then
180. return "=";
190. endif
200. if (isLG) then
210. return "≤";
220. endif
230. if (isMG) then
240. return "≥";
250. endif
260. formula = And(axioms, context1, context2);
270. formulaInCNF = convertToCNF(formula);
280. boolean isOpposite = isUnsatisfiable(formulaInCNF);
290. if (isOpposite) then
300. return "⊥";
310. endif
320. formula = And(axioms, Or(Not(context1), Not(context2)),
            Or(Not(context1), (context2)), Or((context1), Not(context2)));
330. formulaInCNF = convertToCNF(formula);
340. boolean isOverlap = isUnsatisfiable(formulaInCNF);
350. if (isOverlap) then
360. return "∩";
370. else
380. return "idk";
390. endif

We use types of overlap relations defined in [54] 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 1.

<table>
<thead>
<tr>
<th>Semantic relations</th>
<th>Traceability relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalence (=)</td>
<td>overlapTotally (=)</td>
</tr>
<tr>
<td>More or less general (≥, ≤)</td>
<td>overlapInclusively (≥, ≤)</td>
</tr>
<tr>
<td>Mismatch (⊥)</td>
<td>noOverlap (⊥)</td>
</tr>
<tr>
<td>Overlapping (∩)</td>
<td>overlapPartially (∩)</td>
</tr>
</tbody>
</table>

The distinction and implication among different types of traceability relationships is important not only because these 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, with more general and less general having the same binding power. 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.
4. An Illustrative Example: Multiperspective Requirements Traceability

This section provides an example to illustrate how the enhancement of MUPRET framework can resolve the semantic heterogeneity problems found in multiperspective requirements artifacts efficiently. We demonstrate that two different stakeholders (Stakeholder 1 and 2) produce Doctor Investigation System (DIS) and In-Patient Registration System (IPRS) expressed in terms of the textual requirements and the ER diagram respectively. They want to collaborate with each other by sharing their artifacts of two overlapping systems which are parts of a hospital information system.

Artifact 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. |

Artifact 2: (IPRS perspective)

Both artifacts are overlapping and generated with respect to different perspectives. They are passed into the REG, MEG and the ROC module to automatically construct the DIS and IPRS requirements ontology as depicted in Figure 4 (exclude a base ontology).

![Figure 4. DIS and IPRS requirements ontology](image-url)
We use the standard DPLL-based SAT solver [53] to check the unsatisfiability of a propositional formula done by the OM module. The following cases exemplify how to match the ontology concepts for generating different types of traceability relationships as defined in Table 1.

**Case 1: Equivalence relation (overlapTotally)**

From the example in Figure 4, 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.

\[
((staff_1 \leftrightarrow staff_2) \land (doctor_1 \leftrightarrow physician_2)) \land \\
(staff_1 \land doctor_1) \land \neg(staff_2 \land physician_2)
\]

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.

**Case 2: Subsumption relation (overlapInclusively)**

From the example in Figure 4, trying to prove that \( last\_name_1 \) of \( nurse_1 \) in DIS requirements ontology is less general than \( name_2 \) of \( staff_2 \) in IPRS requirements ontology, requires constructing the following formula.

\[
((staff_1 \leftrightarrow staff_2) \land (nurse_1 \leftrightarrow nurse_2) \land (name_1 \leftrightarrow name_2) \land (last\_name_1 \rightarrow name_2)) \land \\
(staff_1 \land nurse_1 \land name_1 \land last\_name_2) \land \\
(\neg(staff_2 \land name_2) \land \neg(staff_2 \land nurse_2 \land name_2) \land \neg(staff_2 \land physician_2 \land name_2) \land \\
\neg(staff_2 \land surgeon_2 \land name_2) \land \neg(staff_2 \land physician_2 \land surgeon_2 \land name_2))
\]

We use the implicit rules to inherit all attributes from super concept to sub concepts for generating implicit relationships. Hence, \( name_2 \) is also an attribute of \( nurse_2, physician_2 \) and \( surgeon_2 \) in IPRS requirements ontology. The above formula turns out to be unsatisfiable, and therefore, the less general relation holds between two concepts.

**Case 3: Mismatch relation (noOverlap)**

From the example in Figure 4, trying to prove that \( admit_1 \) in DIS requirements ontology mismatches \( surgeon_2 \) in IPRS requirements ontology, does not need to construct the propositional formula. Since \( admit_1 \) and \( surgeon_2 \) have different types of concepts based on using a base ontology, the mismatch relation is generated immediately for this case.

Another example in Figure 4, trying to prove that \( name_1 \) of \( patient_1 \) in DIS requirements ontology mismatches \( name_2 \) of \( staff_2 \) in IPRS requirements ontology, the idk relation holds between two concepts. However, both concepts have different hypernyms of the same synset, called homonym. As a result, the final relation for the given pair of concepts is the mismatch by doing the post-process of ontology matching for this case.

**Case 4: Overlapping relation (overlapPartially)**

From the example in Figure 4, trying to prove that \( nurse_1 \) in DIS requirements ontology is overlapping with \( physician_2 \) in IPRS requirements ontology, requires constructing the following formula.

\[
((staff_1 \leftrightarrow staff_2)) \land \\
(\neg(staff_1 \land nurse_2) \lor (staff_2 \land physician_2)) \land \\
(\neg(staff_1 \land nurse_2) \lor (staff_2 \land physician_2)) \land \\
((staff_1 \land nurse_2) \lor (staff_2 \land physician_2))
\]

The above formula turns out to be unsatisfiable, and therefore, the overlapping relation holds between two concepts.
5. Conclusions and Ongoing Work

The contribution of this paper is to propose the enhancement of MUPRET framework for tracing multiperspective requirements artifacts expressed in terms of natural language or plain English text and the requirements model, particularly the ER diagram, based on using the rule-based approach and the ontology-based approach. The traceability relationships are identified by deriving semantic analogy of ontology concepts representing requirements elements and model elements.

Although the current stage of the MUPRET framework focuses on tracing multiperspective requirements artifacts represented in terms of the requirements text together with the requirements model, it is possible to extend the framework to cover multiperspective requirements artifacts expressed in terms of other modeling or the source code. This can be done by adding the semantics of those model elements or source code elements to the base of the MUPRET’s requirements ontology. Additionally, we also aim at exploring further how to apply our MUPRET framework to support traceability throughout a complete software development process.

6. Acknowledgements

The author is supported by the grant from University of the Thai Chamber of Commerce for their financial support for this research.

7. References


