Foundations for an Expert System in Domain-Specific Traceability

Jin Guo
School of Computing
DePaul University
Chicago, IL, 60604, USA
jguo9@mail.depaul.edu

Jane Cleland-Huang
School of Computing
DePaul University
Chicago, IL, 60604, USA
jhuang@cs.depaul.edu

Brian Berenbach
Siemens Corporate Technology
New Jersey, USA
brian.berenbach@siemens.com

***Abstract*—Attempts to utilize information retrieval techniques to fully automate the creation of traceability links have been hindered by terminology mismatches between source and target artifacts. Therefore, current trace retrieval algorithms tend to produce imprecise and incomplete results. In this paper we address this mismatch by proposing an expert system which integrates a knowledge base of domain concepts and their relationships, a set of logic rules for defining relationships between artifacts based on these rules, and a process for mapping artifacts into a structure against which the rules can be applied. This paper lays down the core foundations needed to integrate an expert system into the automated tracing process. We construct a knowledge base and inference rules for part of a large industrial project in the transportation domain and empirically show that our approach significantly improves precision and recall of the generated trace links.**

***Index Terms*—requirements traceability, logic, rules**

I. Introduction

Traceability plays an essential role in the software and systems development life-cycle of safety-critical systems. It is used to demonstrate that all identified hazards have been sufficiently mitigated in the delivered system, and that the system is safe for use. Following an initial risk analysis in which hazards and contributing faults are identified for the system, the project team specifies a set of mitigating requirements and then designs and implements a solution to satisfy those requirements [19], [21]. Trace links are incrementally constructed between hazards, contributing faults, mitigating requirements, design, code, and test cases in order to demonstrate that the risks have been addressed in the delivered system [19], [21], [24]. In many domains, the initial risk analysis is augmented by a set of regulatory codes, against which all systems in the domain must comply.

In practice, the trace links needed to demonstrate compliance are often not created due to time pressure demands, coordination issues, and lack of sufficient tooling [15]. This problem is especially evident in large software-intensive systems engineering projects where the cost and effort involved in establishing and maintaining accurate trace links can be inordinately expensive [4]. For example, in a recent study, we investigated tracing practices evident in the documentation of medical devices submitted to the US Food and Drug Administration (FDA) for approval [24]. Our study revealed that many projects lacked sufficient and/or strategic trace links needed to clearly demonstrate that the system was safe-to-use. In practice, the task of creating, evaluating, and approving trace links is frequently deferred until very late in the project, where it is often conducted by people other than the original developers, testers, and requirements analysts. As a result trace links are often incomplete and/or inaccurate [24].

Given the time-consuming nature of manually creating and maintaining trace links, many researchers have focused their efforts on developing techniques that provide semi-automated support for trace creation and maintenance. These techniques use standard information retrieval methods such as the Vector Space Model (VSM), Latent Semantic Indexing (LSI), and probabilistic networks [1], [10], [17]. Other researchers have developed trace capture mechanisms in which software engineering environments and configuration management tools are instrumented to automatically or semi-automatically capture relationships between artifacts and then to infer and generate trace links [3]. Despite these advances, automated techniques often return only 85-90% of the targeted links at precision rates of 10-50% [17]. These precision rates have hindered the use of trace retrieval techniques in industry and mean that despite a number of pilot studies [4], there has not yet been any significant mainstream adoption in industry.

One of the greatest barriers in the automated tracing task is the mismatch of terminology which often exists between source and target artifacts [5]. As a result of this mismatch, some critical trace links are not successfully retrieved, and/or are retrieved based upon supporting terms instead of core terms. This leads to the precision problem in which the only way to achieve high recall is to spread a wide net and to retrieve target documents with low-levels of similarity to the source artifact. Drawing on an example from the health-care industry, a HIPAA technical safeguard that states "Implement electronic procedures that terminate an electronic session after a predetermined time of inactivity" might incorrectly be traced to the product requirement "The system shall provide the ability to send an electronic prescription to the pharmacy" simply because both artifacts share the term *electronic*. Conversely, the same HIPAA requirement might fail to trace to the requirement "Log out automatically if no keystrokes or mouse movements are detected after 3 minutes," because there are no

978-1-4673-5765-4/13/$31.00 © 2013 IEEE

significant shared terms even though the requirement clearly supports the HIPAA technical safeguard.

This mismatch can be addressed through the use of an ontology which models a set of domain concepts and equivalences, generalizations, and containment relationships [28]. Such an ontology could provide the means for inferring links between related artifacts even when those artifacts do not share the same terms [16], [22]. Furthermore, an expert system can be built around an ontology (referred to as a knowledge base (KB) from now on), and given the capability to emulate the decision-making ability of a human analyst [18].

This paper describes a rudimentary expert system which captures the concepts and relationships of the domain, presents the rules that two human analysts discovered and applied to these concepts in order to construct accurate trace links, models these rules using Prolog, and then defines a process for mapping artifacts into a structure against which the rules can be applied. We refer to our approach as **Do**main-**C**ontextualized **E**xpert **T**raceability (DoCET). Previous researchers, in particular Breaux, Anton, et al. have applied similar techniques to evaluate regulatory compliance of product requirements to standards. Their focus has been on the areas of security, privacy [9], and accessibility [8] and has emphasized elements such as *rights*, *obligations*, and *permissions*. Their work evaluated compliance of industrial requirements to accessibility standards by extracting requirements from the standards using an approach they refer to as the frame-based method, and then manually performing a gap analysis between the two sets of requirements [8]. However, our work focuses on providing support for automated (or semi-automated) tracing within a specific domain. To this end, we require a structured representation that supports automated reasoning to determine whether two requirements should be linked or not. In later sections of the paper we discuss specific overlaps and differences between our work and existing techniques.

To achieve complete trace automation in the long-term, it will be necessary to automate all three parts of the tracing process, including (1) KB construction, (2) mapping unstructured natural language requirements into the more formal structures interpretable by the expert system, and (3) the integration of the expert system into the tracing process. In this paper we focus upon describing the structure of the KB, and the rules needed to infer relationships between concepts. Automation efforts are limited to utilizing the constructed KB to generate trace links. We believe efforts towards more complete automation are only worthwhile once the ideal KB structure and an appropriate set of rules have been discovered.

The remainder of the paper is structured as follows. Section II describes the context of our study while section III provides an overview of the DoCET approach. Sections IV to VI describe the KB construction, artifact structuring, and inference rules in greater detail. Section VII describes a series of four experiments we conducted to evaluate our approach. Section VIII discusses threats to validity, and section IX describes related work. Finally, section X concludes with a discussion of our findings and plans for the next phase of our work.

II. Study Context

The study described in this paper was conducted for a specific sector of the transportation domain. The project included 224 *System Requirements* (SRs), 945 *System Design artifacts* (SDs), and 582 *SubSystem Requirements* (SSRs). Given the labor intensive nature of our study, we focused on a small subset of functional requirements that included 30 SRs which were written at a high-level of abstraction, and 24 SDs written at a much lower-level of abstraction.

Due to the proprietary nature of the project we are not able to disclose the specific domain of the project, and therefore we have obfuscated all of the examples into the *Driver-Optional Highway System* Domain (DOHS). The DOHS provides environmental controls used by both manned and unmanned vehicles to propose routing, prevent vehicles entering accident zones, and to provide a wide range of directives.

To verify that the design of the system fulfills all relevant regulatory codes, requirements engineers need to construct trace links between requirements and regulatory codes, and between design artifacts and requirements. However, as the documents are not written at the same level of abstraction, they lack common terminology making them difficult to trace using traditional trace retrieval techniques. For example, the SR includes the requirement that *the Highway Wayside Segment shall monitor signal, road work directive, and hazard detector information from field devices*, while the correctly linked SD states that *during lamp-out conditions the WIU shall send the current state of the highway signal,* representing a clear mismatch of terms. To establish a trace link between these two artifacts an analyst needs pertinent domain knowledge. For example, she would need to know that a *WIU* is a *Highway Wayside Interface Unit*, and that *WIUs* are located in *Highway Wayside Segment* and are connected to *field devices*. Monitoring the information from *field devices* is achieved when the *WIU* transmits data, such as information about the current status of a signal, from *field devices* to the *Highway Wayside Segment*.

III. Overview of the DoCET System

DoCET involves four different phases described below, and depicted in Figure 1.

• **Knowledge Base Construction** involves capturing and modeling the vocabulary and facts of the domain. Once constructed, it is expected that a KB can be used across multiple projects from the same domain.

• **Artifact Structuring** involves mapping the text in an artifact into a structured format interpretable by the expert system. For purposes of this paper we performed the task manually; however in future work we will investigate the use of Natural Language Processing (NLP) techniques to automate the task.

• **Inference Rule discovery** identifies a set of rules which can be applied to the structured artifacts in order to define specific relationships between requirements and design artifacts. These rules are generic in nature and are expected to be applied

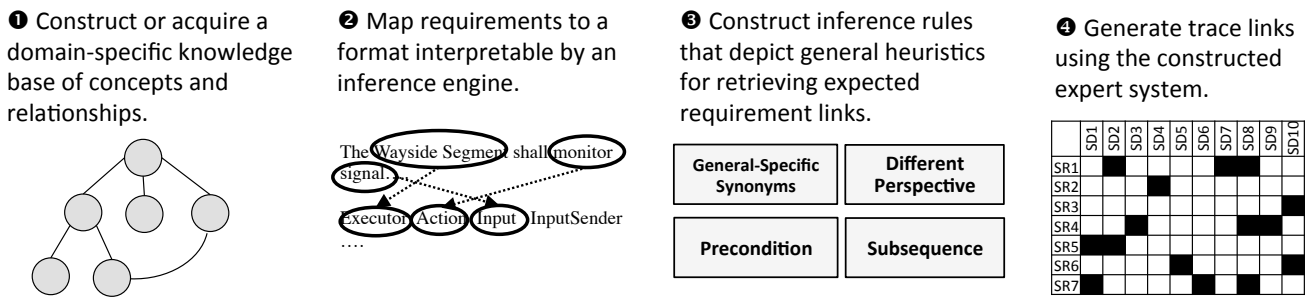

Fig. 1. Overview of the four Phases of constructing and using the Domain-Specific Expert System for Traceability

across multiple domains. However additional rules can be added as-needed.

• **Trace Link Generation** involves applying the inference rules to the mapped artifacts and the KB in order to generate a set of trace links.

IV. Constructing a Domain-Specific Knowledge Base

In its most general form, a KB is an information repository that is used to collect, organize, and search data [6]. Many KBs are built in a manner that supports automated deductive reasoning. Such KBs are composed of a set of data that includes basic terms that define the vocabulary of the domain, and a set of sentences that describe the relationships between those terms. The data is often represented in the form of an ontology [16] in which knowledge is constructed using logical operators such as *and* and *or*, as well as *implication* and *negation* to build complex ideas from more primitive concepts.

If represented using a declarative language, the KB supports automated deductive reasoning and can be used in the traceability domain to infer relationships between concepts found in source and target artifacts. Continuing with our previous example of the *WIU*, deductive reasoning applied to the KB would deduce that a *highway signal* is a type of *field device* connected to a *WIU*. This would contribute to the overall reasoning that the SD should be linked to the SR.

A. Knowledge Representation

There are several approaches that can be taken to represent knowledge and to reason over it, including logical programming languages such as Prolog, functional programming languages such as Lisp, and even procedural languages such as C++. We chose Prolog for our experiments because it is suitable for representing domain knowledge which involves structured objects and their interrelationships [6], and therefore is a good match for the characteristics of the tracing domain. Additionally, knowledge can be specified concisely in a highly readable format in Prolog, but is more difficult and verbose to specify in Lisp or C++. Using Prolog, we constructed the KB by first creating a vocabulary and some simple facts, and then composing these facts into more complex concepts. We describe each of these constructs below.

• **Vocabulary:** Every domain has its own individual vocabulary to describe its primary concepts. These concepts can be identified manually or in an automated way. For example, in prior work we developed a tool that used Q-Tag to extract nouns and noun-phrases, and then compared the frequency at which these elements occurred in the domain-specific documents against the frequency with which they occurred in the American National Corpus in order to identify terms and phrases that occurred significantly more frequently in the domain than in the general language [14], [31]. These terms and phrases constitute the vocabulary of the domain and are represented by constants in the KB. Examples from the DOHS domain include *automobile_segment*, *control_subsystem*, and *road_bulletin*.

It is necessary to represent more general types of objects such as *system*, *subsystem*, and *message*, as well as common attributes for objects such as *operational* and *instant*. They are also represented by constants. Furthermore, to express relationships that occur between two or more objects, we need additional vocabulary such as *synonym* and *has_part*. These vocabularies are represented by predicate symbols.

• **Basic Facts** After capturing the vocabulary of the domain, basic facts are constructed through writing atomic sentences. For example, to specify that *DH* and *driverless highway* are synonyms, we could construct the atomic sentence *synonym(the_DH, driverless_highway_system)*. Similarly, *has_part(the_DH, highway_wayside_segment)* means that the highway wayside segment is one part of the DH system, while *have_mode(onboard_module, cut_out)* means that *cut_out* is one mode of the *onboard_module*.

While some degree of automated support may be possible for constructing basic facts, domain expertise is needed to identify and/or confirm basic facts. For purposes of our experiments we constructed all facts manually through inspecting trace links between SR and SD artifacts.

• **Complex Facts:** Some facts are too complex to be captured by atomic sentences and must therefore be represented by more complex formulas. For instance, a fact such as *All diagnostic tests, except for on-demand ones, are self-diagnostic* could be written as:

```
is_a(X, self_diagnostic) :-
    is_a(X, diagnostic),
    X \= on_demand_diagnostic.
```

where the uppercase letters represent variables.

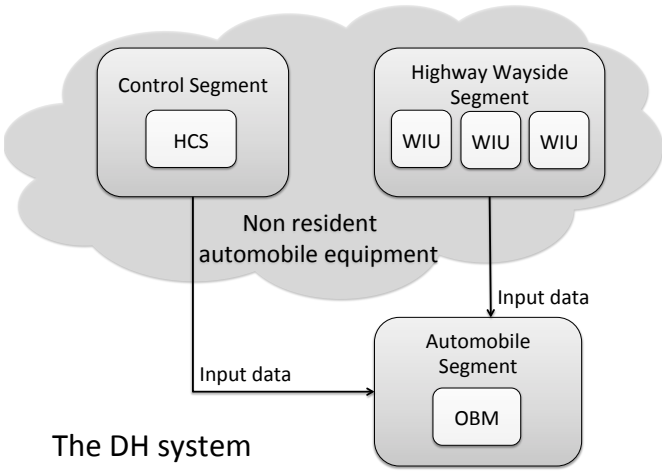

Fig. 2. Information in Sample Requirement

• **Terminological Facts:** In addition to facts about objects and their relationships, domain knowledge also needs to include facts about relationships among previously defined predicate symbols. These relationships are referred to as *terminological facts* and include *Disjointness*, *Subtypes*, *Symmetry*, *Inverses*, and *Type restrictions* [6]. For example, the relationship between predicates *father_of* and *child_of* are inversive and can be expressed as follows:

father_of(X, Y) :-
child_of(Y, X).

B. Example

Using these concepts, we can incrementally construct a KB through inspecting upfront documentation matter and requirements. For example, the *system requirement specification* for the DOHS includes the following information:

The core of the DH system is formed by the onboard module (OBM), which is part of the automobile segment. Non-resident automobile equipment consists of wayside interface units (WIUs, part of the highway wayside segment) and a highway control server (HCS, part of the control segment), supplying input data to the onboard system.

This can be translated into the following facts in the KB and is depicted visually in Figure 2:

has_part(the_DH_system, automobile_segment).
has_part(the_DH_system, highway_wayside_segment).
has_part(the_DH_system, control_segment).
has_part(automobile_segment, the_OBM).
has_part(highway_wayside_segment, the_WIU).
has_part(control_segment, the_HCS).
is_a(the_WIU, non_resident_automobile_equipment).
is_a(the_HCS, non_resident_automobile_equipment).

V. Structuring traceable data

There are two primary approaches to utilizing the KB to generate trace links. The first approach, which we refer to as *relaxed inferencing*, infers general relationships between any term found in a source artifact and any term found in the target artifact, regardless of the role that the term plays. As a simple example, if the term *WIU* appears in the source artifact and the phrase *Wayside Interface Unit* appears in the target artifact, then it increases the likelihood of the two artifacts being linked. This approach takes evidence from the KB and uses it to establish indirect relationships. The major drawback to adopting this approach is that it still treats an artifact as a bag of terms and phrases and therefore generates numerous false links due to naivete concerning the role played by each term.

A. Linguistic Model

The second method, which we refer to as *structured inferencing* takes a far stricter approach that requires specific parts (roles) of a source artifact to be matched to specific parts in the target artifact. Based on our observations of how tracing tasks are performed, it is apparent that the domain experts look for matches between specific roles. For example an SR stating that *the vehicle must send its GPS coordinates to the command center* is very similar to the SD that *the command center must send destination GPS coordinates to the vehicle*; however the meaning is completely different because the *sender* and *receiver* are switched and in the SR the GPS coordinates are sent for reporting/tracking purposes, while in the SD, the GPS coordinates are intended as route planning directives. Traditional trace retrieval algorithms, which act upon bags of words, would have a difficult time differentiating between these two artifacts.

To address these problems, it is necessary to format requirements in a fairly structured way, and apply rules that facilitate the matching of specific roles. One option for structuring artifacts involves a part-of-speech approach in which sentences or phrases were mapped to the basic subject, verb, and object linguistic roles. A similar approach was used by Breaux et al. in their earlier work, which modeled natural language regulatory documents [9].

For the purpose of this study, the approach we took was to identify a structure that worked most appropriately for the specific requirements of the transportation project and then, as an initial proof-of-concept, to evaluate its generalizability to an additional domain. The proposed structure was developed through a systematic evaluation of requirements in the transportation domain conducted by one member of our team from DePaul and evaluated by a Subject Matter Expert from the case company. We refer to it as the Linguistic Model (LM).

The LM contains seven distinct parts: Executor, Action, Input, Input Sender, Output, Output Receiver, and Condition. This model is used to structure each artifact, i.e. requirement or design element [8]. In cases where the artifact is non-primitive, i.e. contains more than one requirement, we construct one model for each distinct part of the requirement.

• The **executor** performs a single action. It is normally the subject of a sentence or clause. For example, in *once the Dispatcher grants the release*, the words "the Dispatcher" serve as the executor. However, there is often no executor in a passive sentence. For example, in *The highway signal*

should be sent to the highway wayside segment, the executor is absent.

• The **action** refers to the specific activity that the executor performs. It is normally the primary verb of a sentence or clause. For example, in *once the Dispatcher grants the release*, the word "grant" is the action. The action should be as specific as possible. For example, in *The system shall support reception and decomposition of the Status Messages*, the action should be "receive" and "decompose", rather than "support". For the same reason, in *The system shall provide the mechanism to receive and decompose the Status Messages* the action should also be "receive" and "decompose", rather than "provide".

• The **input** refers to an object that is taken as input by the executor. Depending on the meaning of the verb, the input could be the object of the sentence. For example, in *receive message from the subsystem*, the "message" is the input.

• The **input sender** represents the actual object that sends the input to the executor. For example, in *receive message from the subsystem*, "the subsystem" is the input sender.

• The **output** represents information sent (or output) by the executor often as a result of some action that the executor has performed. Depending upon the meaning of the verb the output could be the object of the sentence. For example, in *send message to the subsystem*, the term "message" is the output.

• The **output receiver** is the object that accepts output or influence from the executor. It is usually represented as the noun following a preposition such as *to*, *for*, etc. For example, in *send message to the subsystem*, the output receiver is "the subsystem".

• The **condition** describes a state which must hold in order for the requirement to be relevant and for an action to be performed. For example, in *Any critical failure during the Disengaged Mode will force the OBM to enter the Failed Mode*, the phrase "during the Disengaged Mode" serves as the condition.

B. More Complex Artifacts

Encoding the artifacts using the linguistic model is not always sufficient for representing all of the necessary information. Based on the underlying meaning of the words and the project domain, it is sometimes necessary to include additional information for tracing purposes. For example, when the SR states that *the system shall enforce information provided by road bulletins*, the actual functions that the system should perform include: (1) acquire road bulletins, (2) parse the road bulletins to determine the rules written in the road bulletins; and finally (3) enforce the rules. Without the relevant domain knowledge, this essential information will not be considered during the tracing process and therefore trace links that should be created between high and low-level artifacts will be missed. Therefore, the linguistic model is expanded to accommodate information that goes beyond the literal content of each requirement.

This decomposition can be achieved using the KB. The knowledge represented in the above example can be generalized as: decompose the function *A enforce B* to (1) *A acquire C*, (2) *A parse B from C*, and (3) *A enforce C* in the condition of *C can provide B*.

C. Mapping Effort

Mapping each requirement into this linguistic structure clearly takes a significant amount of time and effort. If this mapping task is performed manually (as in this paper), the cost and effort of tracing will be excessive. However in order to truly mimic the way a human performs a tracing task it is essential to create this type of structure and then to map artifacts onto it. In some respects this paper therefore takes a backward step in the quest towards automated traceability; however from the broader perspective, this backward step is necessary in order to build a strong foundation that defines the way in which human analysts trace, and therefore creates a clear roadmap specifying the direction in which automation should be focused in the future. Furthermore, in the case of tracing to regulatory codes, the regulatory codes could be pre-coded into the required linguistic format, meaning that the real-time mapping effort will be focused upon the project-specific requirements only.

VI. Inference Rules

Finally, based on the KB and the linguistic model, we constructed a set of rules that can be used to generate trace links. These rules, depicted in Figure 3, were derived from our observations of how domain knowledge is used to help create and/or reject trace links in the transportation domain project. Several rules were created as a result of analyzing the artifacts and trace links provided to us by our industrial source. Some of these rules have also been observed in previous studies [8], [9]. We do not claim that these rules are complete with respect to the entire domain; however we present them as an initial set of rules discovered for the subset of data studied.

A. Rule Definition

We illustrate each rule with examples from the DOHS domain. In the following discussion, X_1 denotes the X part of the linguistic model in one artifact while X_2 denotes the corresponding part in the other artifact. We consider X_1 to be *consistent* with X_2 if either $X_1 = X_2$, or X_1 and X_2 are synonyms, or X_1 and X_2 have a composition or ancestor-descendant relationship (or vice-versa) according to the KB.

• **Rule 1: General-Specific (G-S)** This rule aims to capture links between artifacts when an element in one artifact (e.g. the executor) is described in a more general form than the matching element in the other artifact. This rule is the most commonly used rule in all the traces we studied. The same rule was previously observed by Breaux et al. and named *Generalized Concept* or *Refined Concept* depending upon the direction in which the rule was applied [8]. For instance, consider a source artifact written as: *the DH subsystems shall*

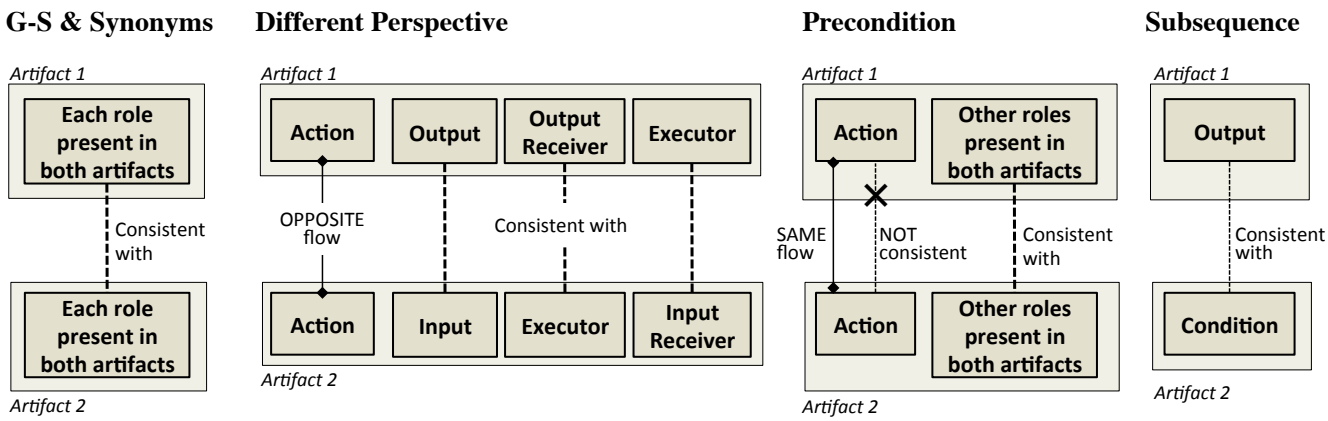

Fig. 3. The five trace inference rules discovered for the transportation sector dataset

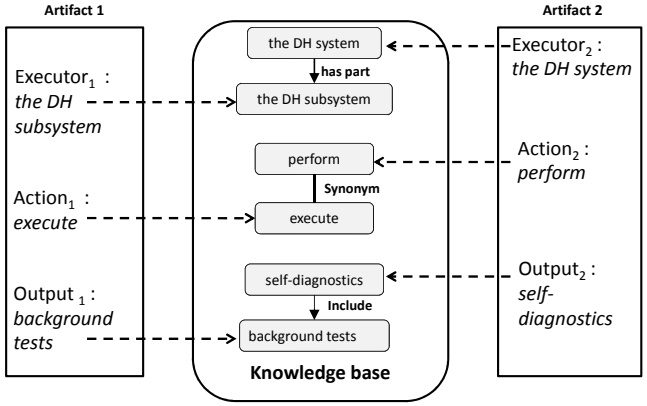

Fig. 4. Example for the General-Specific and Synonym Rules

automatically execute background tests for critical functionality without impact on current operation, while the target artifact is written as: *the DH system shall be capable of performing self-diagnostics.* Because *background tests* is one type of *self-diagnostics*, and *the DH system* includes *the DH subsystem*, these artifacts should be linked. In this case the source artifact is a more concrete example of the target artifact.

This rule is executed by comparing each part of $Executer$, $Action$, $Input$, $InputSender$, $Output$, and $OutputReceiver$ in the linguistic model between the source and target artifact separately. The source and target artifacts are linked if the content of each corresponding part is *consistent*. Elements missing in either the source or target artifacts are ignored.

• **Rule 2: Synonyms** This rule reflects the fact that different requirements often express similar ideas using synonyms. The importance of matching synonyms has been recognized in prior traceability research [23], [31] and shown to improve the quality of generated traces. The *Synonym* rule is applied in exactly the same way as the *General-Specific* rule.

Figure 4 illustrates the application of the *General-Specific* rule and the *Synonym* rule for the previous example. In this case, only the executor, action, and output are found in both the source and target artifacts and are therefore compared using information from the KB.

• **Rule 3: Different Perspective** This rule aims to capture relationships between artifacts where the text describes the same functionality but is written from two different perspectives. For example, one artifact describes the function from the information sender's perspective while the other describes it from the receiver's perspective. This rule is supported by a concept referred to as the *re-topicalization concept* by Breaux et al. [9]. For example, if a requirement is written as: *automobile segment shall send the highway signal to the central control system*, while the target artifact is written as: *the control segment shall receive information from the automobile segment*, then the artifacts should be linked because the functionality they describe is the same.

This rule is executed by comparing the $Action$ parts of the two artifacts, and then comparing $Output_1$ with $Input_2$, $OutputReceiver_1$ with $Executor_2$, $Executer_1$ with $InputSender_2$. If the contents of the $Action$ parts have the opposite direction of information flow, i.e. one sends information out, while the other receives information (according to the relationship between terms captured in the KB), and all other corresponding parts are *consistent* with each other, then the source and target artifacts are linked.

• **Rule 4: Precondition** This rule aims to capture links between artifacts in which the successful completion of the function described in one artifact serves as a precondition to the function described in the other artifact. The rule is also supported by Breaux et al.'s observations of the *precedes* concept [7]. For example a source artifact written as *the DH system shall provide the ability of each subsystem to upload diagnostic data to the HCS*, and a target artifact stating that *the DH system shall provide centralized logging of diagnostic information*, should be linked based on the following rationales. *HCS* is the *central segment* and in order to *log* diagnostic data to the central data store it is necessary to first *upload* the data from the non-central segment to the central segment.

This rule is executed by testing the $Executor$, $Action$, $Input$, $InputSender$, $Output$, and $OutputReceiver$ parts of the source and target artifacts for consistency. If (1) the $Action$ parts are not absent, (2) they are **not** *consistent*, (3) they exhibit the same direction of information flow (based on

information in the KB) i.e. either both receive information or both send information, and (4) all other corresponding parts are *consistent*, then the source and target artifacts are linked.

• **Rule 5: Subsequence** This rule captures links between two artifacts when the function described in one artifact follows the function described in the other. The rule is also supported by Breaux et al.'s *follows* concept [7]. This rule generates trace links required in some projects but deemed unnecessary in other projects. However, this situation is easily accommodated by turning the *Subsequence* rule on or off. For example, one artifact requires the system to *enter Initializing Mode at the start of a new mission, and occupy this mode until road database update and departure test procedures have completed successfully*, while the other artifact states that *upon completion of the initialization tasks, the OBM shall transition into Disengaged Mode*. In this case, the artifacts should be linked because they describe the same sequence of events but from a "before" and "after" perspective. This rule is somewhat similar to the precondition rule; however in the subsequence rule the function executed in artifact 1 is optional, while in the precondition rule it is mandatory.

This rule is executed by comparing $Output_1$ with $Condition_2$. If they are not absent and are *consistent* with each other, the source and target artifacts are linked.

B. Rule Implementation

Each rule is implemented in Prolog. For example, the *General-Specific* rule is implemented as follows:

```
general_specific_rule(Artifact1, Artifact2) :-
  linguistic_model(Artifact1, Executer1, Action1, Input1,
    InputSender1, Output1, OutputReceiver1, Condition1),
  linguistic_model(Artifact2, Executer2, Action2, Input2,
    InputSender2, Output2, OutputReceiver2, Condition2),
  consistent_with(Action1, Action2),
  consistent_with(Input1, Input2),
  consistent_with(InputSender1, InputSender2),
  consistent_with(Output1, Output2),
  consistent_with(OutputReceiver1, OutputReceiver2),
  consistent_with(Executer1, Executer2).
```

The predicate in the first line represents the name of the rule, while the input variables *Artifact1* and *Artifact2* are uniquely defined artifact names. The elements following the ":-" symbol, represent the conditions under which the predicate will be true. The two lines beginning with "linguistic_model" bind all the variables to the components of the linguistic model for the input artifacts. Each pair of bounded variables are then compared based on the specific rules described above. In this way the *General-Specific* rule is used to check each consistency condition and will output "true" when all conditions are satisfied.

VII. Experimental Validation

In this paper we focus on evaluating whether the structure of the KB, the identified linguistic model, and the inference rules are effective for automated tracing purposes. Four specific research questions are defined to evaluate (1) whether the expert system, as constructed, improves tracing results, (2) whether all five rules are effective, (3) the extent to which the linguistic model and five inference rules are generalizable to other domains, and (4) the extent to which a KB constructed using a subset of the requirements in a project is useful for tracing other requirements in the project.

To construct the expert system for the transportation project we randomly picked 30 trace links between SDs and SRs from the industrial transportation project. These links were derived from 30 design artifacts and 24 requirements, representing a possible 720 links. As this set of design artifacts and requirements include 11 additional links, the final dataset included 41 correct trace links. The KB was constructed by incrementally inspecting source and target artifacts related to each of the originally identified 30 true links. For each linked artifact pair, the DePaul researcher reasoned about the rationale for the link, and then added necessary facts to the KB in order to document the rationale. All rationales were evaluated by a Subject Matter Expert (SME) from the industrial project. In some cases, the SME provided additional insights which led to further additions to the KB. As a result of this process we generated 109 basic facts, 13 complex facts, and 20 terminological facts. The expert system used in the following experiments was therefore comprised of the constructed KB and the rules described in section VI.

A. Does the Expert System improve traceability?

In the first experiment we used the KB constructed from the 30 SDs and 24 SRs to generate trace links between them. Trace retrieval results are typically evaluated using recall and precision metrics [11]. Recall measures the fraction of relevant links that are retrieved and is computed as:

$$Recall = \frac{|RelevantLinks \cap RetrievedLinks|}{|RelevantLinks|} \quad (1)$$

while precision measures the fraction of retrieved links that are relevant [11] and is computed as:

$$Precision = \frac{|RelevantLinks \cap RetrievedLinks|}{|RetrievedLinks|} \quad (2)$$

For experimental purposes, a threshold score is established such that all links above or at the threshold are retrieved and all links below the threshold are rejected. Because it is not feasible to achieve identical recall values across every trace query, it can be difficult to compare recall and precision results across experiments. Another metric, known as the F-Measure, computes the harmonic mean of recall and precision is therefore often used for comparative purposes. In this paper we adopt a variant of the F-Measure, known as the F2-Measure, which weights recall values more highly than precision. This measure is commonly used to evaluate experiments in the traceability domain where high recall values are essential. The F2 Measure is computed as follows (where $\beta = 2$):

$$F_\beta Measure = \frac{(1+\beta^2) * Precision * Recall}{\beta^2 * Precision) + Recall} \quad (3)$$

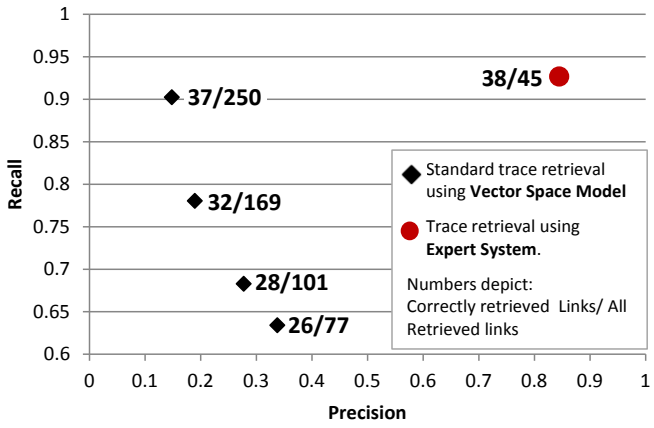

Fig. 5. Comparison of Trace Results using an Expert System versus Traditional Trace Retrieval

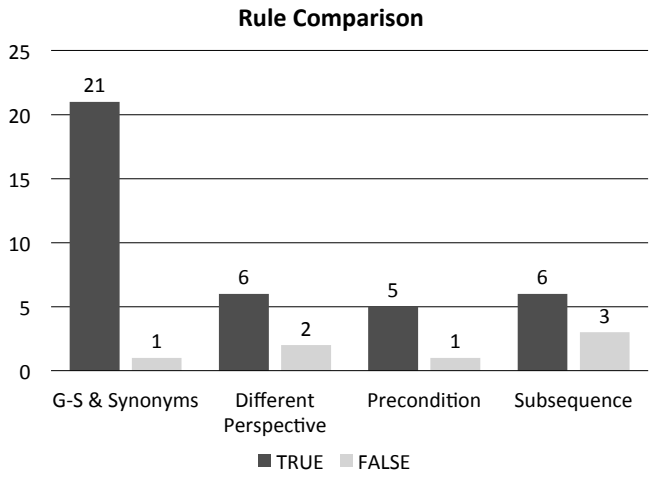

Fig. 6. True and False Links Retrieved by Each Rule

1) Standard Trace Retrieval: To provide a comparative baseline, trace links were generated for the selected artifacts using a VSM approach [17]. The artifacts were pre-processed by removing common words using a standard stopword list, and then stemming remaining words to their morphological roots using Porter's Stemming Algorithm [11]. Each artifact was represented as a vector of terms and the similarity between each design-requirement pair was computed using the cosine similarity formula [11]. A downloadable version of the VSM is available at http://coest.org/mt/13/150.

Results are depicted in Figure 5. At recall levels of approximately 0.9, and precision of 0.18, an F2-measure of 0.50 was achieved. The highest F2-Measure of 0.56 was achieved at recall of only 0.667 (which is unacceptably low for tracing purposes) and precision of 0.338. This is highly representative of the kinds of results traditional trace retrieval algorithms can return.

2) Using the Expert System: The constructed KB, rules, and mappings were then used to generate trace links for the same dataset. The expert system categorizes each source-target pair as linked or unlinked. It successfully identified 38 out of 41 true links, while only retrieving 7 false positives. This resulted in recall of 0.93, precision of 0.84, and F2-Measure of 0.91. This result is also depicted in Figure 5, appearing as the circle in the top right hand corner of the graph. Results obtained from using the expert system are clearly superior to those obtained using the standard trace retrieval approach. At recall levels around 0.9, there was an improvement in precision of 0.66. It should be noted that the expert system was customized for tracing these particular artifacts; however the promising results suggest that focusing automation efforts on constructing a KB and on mapping artifacts into a structured format offers significant potential for a breakthrough in addressing the limitations of term-based approaches such as LSI and VSM.

B. Are all five inference rules effective?

We counted the number of correct and incorrect links retrieved by each rule. As synonyms and generalization co-occurred in multiple artifacts we report the *General-Specific* and *Synonym* rules together. There was no overlap between any other rules. Figure 6 reports the number of true and false links retrieved by each rule. The results show that approximately 55% of the true links were retrieved using the *General-Specific* and *Synonyms* rules, 16% using the *Different perspective* rule, 16% using the *Subsequence* rule, and 13% using the *Precondition* rule. The *General-Specific* and *Synonyms* rules had a precision of 0.955, the *Different perspective* rule had precision of 0.75, and the *Precondition* and *Subsequence* rules had precisions of 0.833 and 0.667 respectively. These results suggest that the *Subsequence* rule, in particular, needs further investigation and might benefit from additional refinements or supporting rules, perhaps in the form of exceptions.

C. Are the inference rules generalizable to other domains?

As a preliminary evaluation of whether the rules are generalizable to other domains we randomly selected ten trace links from a set of traces between requirements for the World Vista electronic health record and health information system and certification requirements taken from the Certification Commission for Health Information Technology (CCHIT). All source and target requirements for the ten links were mapped to the linguistic model, and the artifact-pairs were then evaluated against the five inference rules we had previously created. The *General-Specific* rule was able to capture six of the ten links, while the *Different perspective* rule captured an additional three. The missing link was not expressible using any of the existing five rules. These results suggest that at least two of the rules might be applicable across other domains, but also highlight the need for studying multiple domains in order to identify a broadly applicable set of general rules.

D. Can the KB be applied to additional requirements?

While it was obvious that the constructed KB was far from complete because it had been constructed based only upon a very small subset of SRs and SDs, we evaluated it against an additional set of ten SDs. These SDs were linked to the

original SRs (to avoid additional manual mapping efforts) and resulted in 14 true links and 226 false ones. Without adding any additional knowledge to the KB, the expert system performed poorly and returned only 5 true links and 1 false link. In a further step we extended the KB by adding the following additional items:

synonym(field_device, field_element)
synonym(broadcast, send)
synonym(automobile_segment, car)
synonym(authorization, permission)
action_feature(use, flow_in)
include(complete, perform)
synonym(central_system, centralized_storage)
include(log_file, diagnose_information)
include(send, upload)

With the extended KB, 13 out of the 14 true links, and only one false link, were retrieved. It is worth noting that four of the 14 true links were not included in the original trace matrix but were confirmed as correct by a domain analyst from the case company. DoCET was therefore again able to find trace links previously missed by human analysts.

VIII. Threats to Validity

Threats to validity can be classified as construct, internal, external validity, and reliability. Construct validity evaluates the extent to which the construct that was intended to be measured was actually measured. Our experiments compared the quality of trace links generated using the KB versus those generated using a standard VSM approach [17]. Results were measured utilizing metrics commonly used in traceability research, and broadly accepted to differentiate between high and low quality links [17]. The generated links were compared against a golden-answer set provided by our industrial collaborators. In cases where DoCET retrieved additional links we presented them to the domain analyst for review. As trace links created by domain analysts tend to be fairly accurate but incomplete it is possible that additional correct links exist but were not found.

Internal validity reflects the extent to which a study minimizes systematic error or bias, so that a causal conclusion can be drawn. We hypothesize that the lack of domain knowledge and the inability to relate concepts that use dissimilar terms causes low trace accuracy in current tracing algorithms [17]. Our controlled experiment was designed to compare results obtained using VSM vs. DoCET (which utilizes domain knowledge to generate trace links). The results reported in section VII-A support our hypothesis.

External validity evaluates the generalizability of the approach. The major threat stems from the fact that we constructed the KB based upon a small subset of artifacts associated with a randomly selected set of trace links. However, the purpose of our study is to show that if a suitable and complete KB is constructed for a certain domain, then this KB can be used to significantly improve trace quality. The future challenge is to find ways to construct a domain-specific KB that can be reused across multiple projects in the domain, so that high quality trace links can be generated without the need to build a KB during the tracing process. In addition, our work focused primarily on a single industrial datasets; however we conducted a small proof of concept experiment which suggested the ability of the identified rules to generalize across an additional domain.

Finally, a threat to reliability lies in the complexity of the domain, and the fact that the initial KB construction was conducted by one of the DePaul researchers whose domain knowledge was acquired through reading documentation for purposes of this project. Any misunderstanding of domain concepts would lead to incorrect and/or missing facts in the KB. However, this threat was partially mitigated through an onsite visit by a domain expert from the case company who provided clarifications which led to modifications to the KB. This feedback, coupled with the high quality trace retrieval results, suggests that the threat was sufficiently mitigated.

IX. Related Work

Previous researchers have explored the idea of using ontology for tracing purposes but have not conducted rigorous empirical analyses. Kof et al. [20] used ontology to trace between high level and low level requirements. Their approach involved extracting domain-specific concepts, mapping similar concepts through using WordNet to find synonyms, and finally establishing trace links through the shared concepts. While their work makes significant contributions in the area of extracting ontologies from requirements [20] it does not provide any rigorous analysis of whether the ontology actually improves the quality of generated trace links. Work by Zhang et al. entitled "Ontological approach for the semantic recovery of trace links between software artefacts" [30] is somewhat misleading, as the authors use ontology to represent concepts such as *design patterns*, *sentences*, *paragraphs*, and *classes*, as opposed to domain concepts such as *train* or *signal* found in the domain being traced. Assawamekin et al. utilize Natural Language Processing techniques to construct separate ontologies from each stakeholder perspective and then find a mapping between them [2]. The ontology is expressed in first-order logic and the mapping is solved by a SAT-solver. Ultimately, the ontology is used to create trace links between different perspectives. Unfortunately, the authors have not provided any rigorous empirical analysis of their approach beyond a relatively small example in which the ontology is constructed for purposes of a very small project.

Several authors have investigated techniques for automated ontology building from requirements [12], [13], [26] and these techniques will be considered in the next phase of our work to construct an appropriate ontology for the expert system. Other researchers have developed techniques for transforming relatively unstructured textual information into more formally structured requirements [9]. Young defined attributes of the legal documents as actor, action, object, object's source, target, purpose, and condition [27], identified these attributes from legal documents, and then, with the help of templates

generated structured requirements. In future work we will evaluate whether their approach can be used to automate the DoCET's LM mapping phase. Maxwell and Anton developed an approach named "Production Rule Models" which uses Prolog to represent domain knowledge [25]; this is similar to the KB construction phase of our study. Because their work focused on the artifacts of legal text, they explicitly modeled rights, obligations, and permissions. It should be noted however, that their "rules" refer to a representation of the legal text, whereas our rules are used to represent the rationales that are used to establish links during the tracing process.

In the traceability domain Spanoudakis et al. [29] developed a rule-based approach for generating traceability relations; however their approach has focused upon the structured domain of tracing to UML models.

X. Conclusion

We have developed a preliminary expert system for retrieving trace links in the transportation domain. Results show that our approach has the potential to improve the accuracy of trace retrieval in industrial settings. However, significant effort is going to be required to build viable expert systems for each targeted domain. This investment is worthwhile in domains such as transportation, healthcare management systems, and infusion pumps where traceability is required for safety or confidentiality purposes, and where the expert system could be used across multiple projects. While we recognize that our approach is currently very labor intensive, and therefore in its present form does not alleviate the enormous cost and effort of tracing, we see it as a first-step in a new direction for addressing the traceability problem. We envision a future in which domain-specific KBs are available for supporting traceability in a wide-variety of domains, and in which natural language processing techniques support mapping of artifacts into a format interpretable by an expert system.

Acknowledgments

The work in this paper was partially funded by the US National Science Foundation grant # CCF-0810924.

References

[1] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering traceability links between code and documentation. *IEEE Trans. Softw. Eng.*, 28(10):970–983, 2002.

[2] N. Assawamekin, T. Sunetnanta, and C. Pluempitiwiriyawej. Ontology-based multiperspective requirements traceability framework. *Knowl. Inf. Syst.*, 25(3):493–522, 2010.

[3] H. Asuncion and R. N. Taylor. Capturing custom link semantics among heterogeneous artifacts and tools. In *Traceability in Emerging Forms of Software Engineering*, 2009.

[4] B. Berenbach, D. Gruseman, and J. Cleland-Huang. Application of just in time tracing to regulatory codes. In *Proceedings of the Conference on Systems Engineering Research*, 2010.

[5] T. J. Biggerstaff, B. G. Mitbander, and D. E. Webster. Program understanding and the concept assignment problem. *Commun. ACM*, 37(5):72–82, May 1994.

[6] R. Brachman and H. Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann - Elsevier.

[7] T. Breaux and D. Gordon. Regulatory requirements traceability and analysis using semi-formal specifications. In *Intn'l Working Conf. on Requirements Eng.: Foundation for Software Quality*, 2013.

[8] T. D. Breaux, A. I. Antón, K. Boucher, and M. Dorfman. Legal requirements, compliance and practice: An industry case study in accessibility. In *RE*, pages 43–52, 2008.

[9] T. D. Breaux, A. I. Antón, and J. Doyle. Semantic parameterization: A process for modeling domain descriptions. *ACM Trans. Softw. Eng. Methodol.*, 18(2):5:1–5:27, Nov. 2008.

[10] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Enhancing an artefact management system with traceability recovery features. In *International Conference on Software Maintenance*, pages 306–315, Washington, DC, USA, 2004. IEEE Computer Society.

[11] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Printice Hall, Englewood Cliffs, NJ, USA, 1992.

[12] R. Gacitua and P. Sawyer. Ensemble methods for ontology learning - an empirical experiment to evaluate combinations of concept acquisition techniques. In *ACIS-ICIS*, pages 328–333, 2008.

[13] R. Gacitua, P. Sawyer, and P. Rayson. A flexible framework to experiment with ontology learning techniques. *Knowl.-Based Syst.*, 21(3):192–199, 2008.

[14] M. Gibiec, A. Czauderna, and J. Cleland-Huang. Towards mining replacement queries for hard-to-retrieve traces. In *ASE*, pages 245–254, 2010.

[15] O. Gotel and C. Finkelstein. An analysis of the requirements traceability problem. In *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pages 94 –101, apr 1994.

[16] T. Gruber. Ontology. In *Encyclopedia of Database Systems*, pages 1963–1965. 2009.

[17] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Trans. Softw. Eng.*, 32(1):4–19, 2006.

[18] P. Jackson. *Introduction To Expert Systems (3 ed.)*. Addison Wesley, 1998.

[19] Jane Cleland-Huang, M. Heimdahl, J. Huffman Hayes, R. Lutz, and P. Maeder. Trace queries for safety requirements in high assurance systems. In *Intn'l Working Conf. on Requirements Eng.: Foundation for Software Quality*, 2012.

[20] L. Kof, R. Gacitua, M. Rouncefield, and P. Sawyer. Concept mapping as a means of requirements tracing. In *MaRK'10)*, 2010.

[21] N. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press, 2011.

[22] Y. Li and J. Cleland-Huang. Ontology-based trace retrieval. In *Traceability in Emerging Forms of Software Engineering (TEFSE2013*, San Francisco, USA, May 2009.

[23] J. Lin, C. C. Lin, J. Cleland-Huang, R. Settimi, J. Amaya, G. Bedford, B. Berenbach, O. B. Khadra, C. Duan, and X. Zou. Poirot: A distributed tool supporting enterprise-wide automated traceability. In *RE*, pages 356–357, 2006.

[24] P. Mäder, P. Jones, Y. Zhang, and J. Cleland-Huang. Strategic traceability for safety critical systems. *IEEE Software*, 30(3), 2013.

[25] J. Maxwell and A. Anton. Developing production rule models to aid in acquiring requirements from legal texts. In *Requirements Engineering Conference, 2009. RE '09. 17th IEEE International*, pages 101–110, 2009.

[26] A. D. Nicola, M. Missikoff, and R. Navigli. A software engineering approach to ontology building. *Inf. Syst.*, 34(2):258–275, 2009.

[27] J. Y. Schmidt. Specifying requirements using commitment, privilege, and right (cpr) analysis. 2012.

[28] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Trans. Knowl. Data Eng.*, 25(1):158–176, 2013.

[29] G. Spanoudakis, A. Zisman, E. Pérez-Miñana, and P. Krause. Rule-based generation of requirements traceability relations. *Journal of Systems and Software*, 72(2):105–127, 2004.

[30] Y. Zhang, R. Witte, J. Rilling, and V. Haarslev. Ontological approach for the semantic recovery of traceability links between software artefacts. *Software, IET*, 2(3):185 –203, june 2008.

[31] X. Zou, R. Settimi, and J. Cleland-Huang. Improving automated requirements trace retrieval: a study of term-based enhancement methods. *Empirical Software Engineering*, 15(2):119–146, 2010.