Revisiting the Fundamentals of Traceability between Requirements and its Realization

Anitha Murugesan, Michael Whalen, Elaheh Ghassabani and Mats Heimdahl
Department of Computer Science and Engineering
University of Minnesota, 200 Union Street, Minneapolis, Minnesota 55455
{anitha, whalen, ghass013,heimdahl}@cs.umn.edu

***Abstract*—Establishing associations, known as a *trace links*, between each requirement and the artifacts that realize,or satisfy, it is essential to, e.g., evaluating the impact of changes to requirements or implementation, or establishing satisfaction arguments. While research into this type of traceability, that we call *Requirements Satisfaction Traceability*, has been an active research area for some time, none of the literature discusses the distinction between establishing ”a” set of trace links between a requirement and its implementation (tracing *one* way the requirements is implemented) vs. ”all” trace links (tracing *all* the ways a requirement is implemented), and the possible ramifications for how the trace links can be used in analysis. We examine how this distinction changes the way traceability is perceived, established, maintained, and used. We introduce and discuss the notion of ”complete” traceability, which considers *all* trace links between the requirements and the artifacts that work to satisfy the requirements, and contrast it with the partial traceability common in practice. The intent of this RE@Next! paper is to highlight the aspect of completeness in traceability and discuss the ramifications of incomplete (or partial) traceability.**

I. Introduction

Requirements traceability – the ability to describe and follow the life of a requirement in system development [?] – has been a topic of great interest in research and practice for several decades. Intuitively, it concerns establishing relationships, called *trace links*, between the requirements and one or more artifacts of the system. Among the several different development artifacts and the relationships that can established from/to the requirements, being able to establish trace links from requirements to artifacts that realize or *satisfy* those requirements – particularly to entities within those artifacts, that we will call *target artifacts* in the rest of the paper – has been enormously useful in practice. For instance, it helps analyze the impact of changes in one artifact on the other, assess the quality of the system, aid in creating assurance arguments for the system, etc. In this paper, we focus our attention to this subset of requirement traceability, that we call *Requirements Satisfaction Traceability.*

Instead of just recording the trace links from each requirement to the target artifacts, *Satisfaction Arguments* [3] offer a semantically rich way to establish them. Originally proposed by Zave and Jackson, satisfaction argument is an approach to demonstrate how the behaviors of the system and its environment together satisfy the requirements. While one could trivially establish this argument between the entire system-environment and the requirement, the intent of the satisfaction argument is to capture only those parts of the system and environment that are necessary to satisfy the requirement. From a traceability perspective, these arguments help establish trace links (the *satisfied by* relationship) between the requirements and those parts of the system and environment (the target artifacts) that were necessary to satisfy the requirements; We call those target artifacts the *set of support* for that requirement. Mathematically, if we think of the argument as a proof in which the requirement is the claim, then the set of support are the axioms (or clauses) that were necessary to prove the claim and the trace links are means to associate the claim to those clauses. However, such proofs are not, in general, unique, and often there are multiple sets of clauses that could be used to construct the proof. Neither Zave and Jackson nor (in our examination) the existing traceability literature discusses multiple alternative satisfaction arguments or sets of trace links for one system design. This casts a doubt on how traceability is perceived in practice.

While it is has been a general practice to refer to traceability as capturing the relationship between artifacts, in our opinion, atleast when discussing requirements satisfaction traceability it is beneficial to distinguish between trace links to “a” set of support vs. “the” sets of support. In so doing, it allows a substantial change in understanding what traceability *means* and how traceability be used. For instance, in safety critical system domain, for fault tolerance one could intentionally add multiple ways to satisfy a requirement or one could unintentionally add system functionality that leads to multiple ways of satisfying a requirement. Although, to argue or prove that the system satisfies its requirement it might be suffice to get one set of support, it is insufficient for other analysis performed using trace information. For example, when there is a change in a requirement, the lack of knowledge of all sets of support might result in either misplaced confidence in the system since the old requirement might still be satisfied by alternate means or spend enormous amount of time to identify all the alternate satisfactions and re-engineer them. *Anitha: I need to rethink if this example emphasises the point.*

Establishing trace links to all sets of support, that we call *complete* traceability, provides us insightful information about the elements of the set of support - ones that are absolutely necessary, the ones that are optional and the ones that do not contribute to satisfying a requirement that we categorize them as *Must*, *May* and *Irrelevant* support elements for each requirements. This categorization of support elements is useful in several ways, such as (a) to precisely perform impact analysis, (b) to analyse if changing a part of the system affects the way requirements are satisfied, (c) to identify which parts of the system does not contribute to satisfying any requirement of the system, the so called "Gold Plating", (d) to locate parts of the system that are most critical in terms of satisfying most requirements, (e) to point to system assumptions that are used by most requirements, and the list goes on. In fact, it helps identify the unintended ways the requirement was satisfied by the system and its environment.

In this paper, we examine how this notion of completeness in traceability changes the way it is perceived, established, maintained, and used. We introduce and discuss the notion of completeness in traceability, which considers *all* *satisfied by* trace links between the requirements and the target artifacts that work to satisfy the requirements, and contrast it with the partial traceability common in practice. The intent of this RE@Next! paper is to highlight the aspect of completeness in traceability and discuss the ramifications of incomplete (or partial) traceability. While establishing complete traceability has been considered impossible or extraordinarily difficult to establish in practice [2], our our hypothesis is that, in the realm of formal methods and model based developments, it can be achieved in an automated and efficient manner. Our hypotheses is based on the initial results we got from our recent work on automatically extracting the set of support when formally verifying the requirements of systems using a model based approach.

II. Motivating Example

Consider a safety device whose requirement is to *raise an alarm if a hazardous condition occurs and is persistent for 5 seconds*. The device is composed of two redundant sensors (for fault tolerance) whose requirement is to *detect the hazardous condition and report if it is persistent for 5 seconds* and a buzzer that *sounds an audible alarm if either one of sensors report the hazard*; *the buzzer shall otherwise remain silent*. Since the sensors are redundant, to establish that the system level requirements are satisfied, one of the sensors and the buzzer is sufficient. Say, we only capture the trace from the requirement to the requirements of one of the sensors and the buzzer. Now, if there is a change in device requirement to *raise an alarm if the hazardous condition persists for 3 seconds*, the traceability that we have established only helps identify that one sensor's behaviour should be changed. Since the alternate satisfaction is not documented, if the sensor that was modified for the new requirement fails then the requirement is no longer met and may causes safety hazards. Further, an astute reader might have noticed that the buzzer has two requirements of which only the first one contributes to satisfying the system requirement whereas the other does not. If we were to establish all the trace links between the requirements and its realization, one could use it to identify elements of the realization that do not contribute to satisfying any requirement. While we illustrated with a toy example, in practice systems are complex with hundreds of system requirements and numerous components with potentially thousands of component requirements that may induce several ways to satisfy the requirements. Without insight into the all those ways, in our opinion, the results of analysis using such (partial) traceability with be misleading. Unfortunately, neither the completeness of traceability nor its benefits are discussed in the existing literature.

III. Formal Representation of Traceability

In its simplest sense, traceability is capturing the relationship between artifacts. There are three building blocks for traceability - a *source artifact* from which relationship should be established, *the target artifact* to which the source artifact be related and the *trace relationship*, that describes the association between the artifacts. For requirements satisfaction traceability, the source artifacts are the requirements, the target artifacts are the elements that realize the requirement such as lines of code, design elements, assumptions or low level requirements, and the trace relationship is *satisfied by* (Formally $S \vdash R$ should be read as: the set of target artifacts S satisfies R). While the trace direction could be bidirectional, for now lets consider trace from requirements to its realization. In the next section we explain how this can be used to establish the other direction.

Let R denote the set of all requirements of the system and I denote the set of all target artifacts. As mentioned earlier, the target artifact could be lines of source code, model elements (in model based development), assumptions etc.

A satisfaction argument establishes the validity of a requirement through a sufficient set of target artifacts. *Anitha: am I saying the right thing?*. Formally it is represented as,

$$S \vdash r$$

$$\textit{where } r \in R \textit{ and } S \subseteq I$$

We assume that the satisfaction argument *Anitha: or S?* is monotonic on I, that is

$$(S \subset S' \textit{ and } S \vdash r) \Rightarrow S' \vdash r$$

A *requirement satisfaction trace (RST)*, that is uses a satisfaction argument as a basis, is the association between the requirement to the set of target artifacts that together can establish the satisfaction of the requirements. While traditional way of establishing a trace concerns relating a requirement to a target artifact, the requirements satisfaction trace relates the requirement to a set of target artifacts that can demonstrate the satisfaction of requirements, thus establish a sematic basis for the traceability.

$$RST : r \times S$$

$$\textit{where } \quad r \in R \textit{ and } S \subseteq I \textit{ and } S \vdash r$$

Anitha: I know I asked this before, but just to confirm, is it enough that we mentioned that we assume monotonicity earlier in this section; but somehow it reads as if we discuss about it only in that context of satisfaction argument?

While the set of target artifacts in a requirement satisfaction trace can be include all elements of I, of interest are those target artifacts that are (only) necessary to satisfy the requirement; this avoids a trivial satisfaction argument as well as irrelevant traces. Hence, we define a *minimal requirements satisfaction trace(RST_m)* that associates a requirement to a set of target artifacts that are necessary to satisfy that requirement. We call this minimal set of target artifacts as the *set of support* for that requirement.

$$RST_m : r \times S$$

$$where \quad r \in R \; and \; S \subseteq I \; and \; S \vdash r \; and$$

$$\forall e \in S \cdot S \setminus e \nvdash r$$

While RST_m maps one set of support to a requirement, as mentioned earlier, there could be many sets of support for a requirement. To capture that notion, we define, *complete minimal requirement satisfaction trace ($CRST_m$)* for a requirement as an association to all its set of support.

$$CRST_m(r) : \{S \mid (r, S) \in RST_m(r, S)\}$$

The $CRST_m$ for a requirement, can be envisioned as a matrix in which each row represents a target artifact, columns represents the requirement, and the relationship between the two are recorded at the cells of their intersection. The information in each column in this matrix provides a satisfaction argument for that requirement.

The $CRST_m$ for all requirements (formally $\forall (r \in R) \; CRST_m(r)$), represents the complete traceability of the system. One can envision this as a 3 dimensional traceability matrix, where the third dimension is each requirement.

IV. Complete Traceability

Through formal definitions in the previous section we highlight two critical facets – the semantics and completeness – in establishing traceability. In our experience, we found that just being able to trace a requirement to a target artifact that satisfies it, say a line of code, is not useful; a holistic view of how that line of code in conjunction with other related lines of code satisfy the requirement provides meaningful information to perform analysis such as assessing the impact of a change. By defining a trace from requirements to the set of target artifacts, we advocate that traceability be captured in a way that upholds its semantic rationale. Further, we also found that most analyses that are performed using the established traceability without a clear picture of the actual adequacy of traces is not, in reality, useful and trustworthy. By defining a complete trace for every requirement, we recapitulate that it is critical to assess the thoroughness of the established traces. In this section, we elaborate on how rethinking about these facets helps understand, assess and use traceability precisely.

A. Categorizing the Set of Support

Establishing $CRST_m$ for a requirement, one gets a clear picture of the all possible ways that requirement is satisfied. This information helps categorize each target artifact into one of the following groups for that requirement ($r \in R$), also illustrated using a Venn diagram in Figure 1,*Anitha: not using the figure enough during explanations*

- **MUST** elements - the target artifacts that are present in all the sets of support for a requirement.

$$MUST(r) = \{\bigcap CRST_m(r)\}$$

- **MAY** elements - target artifacts in some sets of support, but not all.

$$MAY(r) = \{CRST_m(r) \setminus MUST(r)\}$$

- **IRRELEVANT** elements - target artifacts that are not in any of the set of support.

$$IRR_x = \{I \setminus CRST_m(r)\}$$

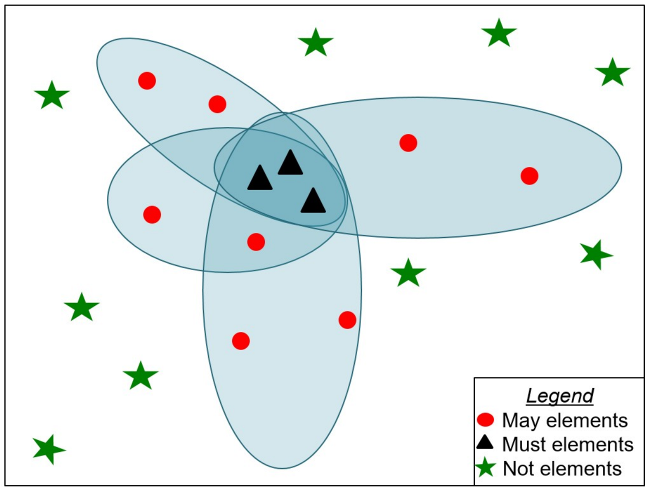

Fig. 1. May - Must - Irrelevant Set of Support

This categorization helps identify the role and relevance of each target artifact in satisfying a requirement. The MUST elements are those target artifacts that are absolutely necessary for the requirement satisfaction. Hence, any change to these elements will most likely impact on each other. On the other hand, MAY elements indicate those target artifacts that satisfy the requirement in one of the possible ways. Any change to just one of these elements will not affect the satisfaction of that requirement. The IRRELEVANT elements are not related to satisfying the requirements, hence neither a change in them affects the requirement satisfaction, nor does a change in requirement induce a change in these elements.

B. Using Traces for Precise Analysis

The categorization of the set of support to be useful in several types of analysis.

a) Impact Analysis: : It helps understand how a change in the requirement change will affect the target artifacts and vice versa. This categorization precisely helps identify those parts of the system that definitely have to change when there is a change in requirement. The MUST elements are those target artifacts that are highly likely to change with any change in the requirement, whereas not all MAY elements need to be changed to satisfy the requirement. Obviously, one need not even worry about the IRRELEVANT elements. If we take all the sets of support for all the requirements and then categorize them, then the elements in the MUST set points to target artifacts that are critical for every requirement. This helps assess how a change in one element may affect other requirements.

Reversing the direction of trace, if we now, take each target artifact and trace to the requirements it satisfies, one can get clear idea about all the requirements it satisfies. If we decide to change a target artifact as long as it is not in any of the MUST sets for any requirement, it becomes evident that it will not affect the satisfaction of any requirement.

b) Verification and Validation: This categorization helps tailor the Verification and Validation in systems. For instance, if most requirements have a certain target artifact in their MUST set, say an particular assumption, it reveals the criticality of focusing attention to validate that artifact. Along the same lines, for a system with a complex architecture (components that each have functionality) such as system of systems, this categorization helps identify components that is critical to satisfy most requirements. This helps plan verification strategies.

Further, the notion of getting complete traces helps access if the requirements are satisfied by the system in an unintended manner. It is well known that issues such as vacuity [**?**] can cause requirements to be satisfied in a trivial manner. Even for non-vacuous requirements, it is possible to over-constrain the *environment* or unintentionally added functionality in the system that caused the requirements to be satisfied. By capturing all the set of support and categorizing them, one could evaluate if there are unintended target artifacts that might cause the requirements to be satisfied.

c) Completeness Checking: By getting all the sets of support for all requirements of the system and categorizing them, it gives a clear picture about the role of target artifacts in satisfying the requirement. By reversing the direction of the complete requirement satisfaction trace, one can find if there are target artifacts that do not trace to any requirement. This is a possible indication of “gold plating” or missing requirements. In other words, it helps assess if there requirements of the system are completely specified with respect to describing all the behaviors of the system. Being able to assess the coverage of requirements over the model is crucial in the safety critical system domain.

d) Bench marking: Anitha: I am not happy with this now.... But I do want to say something along these lines... To the best of our knowledge, the existing benchmarks and ways to establish benchmarks to assess the effectiveness of traceability techniques or evaluate the trace links is not guaranteed to be flawless. The correctness and thoroughness of the benchmarks are in someway assumed and not rigorously assessed. A number of research work compute the precision and recall of their traceability approaches by using such benchmarks. We believe that to evaluate the completeness and precision of those benchmarks we need to first rigorously characterize it. The requirements satisfaction trace defined in the previous section helps us move forward in that direction. Since the trace is based on establishing a satisfaction argument it is straightforward to verify/validate the precision of each trace (which is 100%). The complete requirements satisfaction trace, as the name suggests, is defined to capture traces to all sets of support. While this may sound theoretical, once established it will be the perfect benchmark to compare results to.

Contrary to the general notion that it is impossible or extraordinarily difficult to identify all trace link in practice [2], some of the initial results from our recent efforts [1] indicate that such complete requirements satisfaction trace can be established, in fact automatically in the realm of formal methods and model based developments. In the next section, we breifely describe our prior work and our plans to extend it to establish complete traceability. While the details the approach and initial results are not in scope of this paper, an interested reader is directed to [1].

Anitha: I am working on the following sections...

V. Discussion

VI. Conclusion

References

[1] E. PAPER. Elaheh paper. In *ELAHEH PAPER*, March 2016.

[2] D. Strašunskas. Traceability in collaborative systems development from lifecycle perspective. In *Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE02). ACM, New York*, 2002.

[3] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM transactions on Software Engineering and Methodology (TOSEM)*, 6(1):1–30, 1997.