# Recommending a Framework for Comparison of Requirements Tracing Experiments*

Jane Huffman Hayes
Computer Science Department
Lab for Advanced Networking
University of Kentucky
hayes@cs.uky.edu

Alex Dekhtyar
Computer Science Department
University of Kentucky
dekhtyar@cs.uky.edu

James M. Carigan
Division of Computer and Technical Sciences
Kentucky State University
jcarigan@gwmail.kysu.edu

## Abstract

*To improve enhancement, maintenance, and reuse of systems, we need automated requirements tracing tools. To build the best possible tracing tools, we need a way to conduct and compare experiments on requirements traceability. To that end, we suggest such a framework.*

## 1. Introduction

When traceability matrices are not built by the original developer, or not to the level of detail needed, we are faced with a difficult but important task. The traceability matrix is needed to support such vital activities as maintaining an existing system and regression testing a modified system. The building of traceability matrices by those other than the original developers is an arduous, error prone, prolonged, and labor intensive task. Thus, after the fact requirements tracing is a process where the right kind of automation can definitely assist an analyst. Recently, a number of researchers have studied the application of various methods, often based on information retrieval, to after the fact tracing. The studies are diverse enough to warrant a means for comparing them easily as well as for determining areas that require further investigation. It is our position that a framework is needed for comparing these experiments. To that end, we present here an experimental framework for evaluating requirements tracing and traceability studies. Recent experimental requirements tracing journal and conference papers are catalogued using the framework. We compare these studies and identify areas for future research. Finally, we provide suggestions for how the field of tracing and traceability research may move to a more mature level.

The paper is organized as follows. Section 2 discusses related work. The proposed framework is presented in Section 3. Section 4 describes application of the framework to four studies. Section 5 provides observations from application of the framework. Finally, conclusions and future work are presented in Section 6.

## 2. Related work

Requirements tracing is defined as "the ability to describe and follow the life of a requirement, in both a forward and backward direction, through the whole systems life cycle [12]." Requirements tracing is important to the software engineering field for a number of reasons: traceability matrices (a) assist us in assuring that all requirements have been implemented, (b) assist in mapping test cases to requirements, (c) are used by management for "what if" scenarios, (d) assist us when we maintain software or reuse software, and (e) form a part of the safety case for safety-critical software requiring certification. Unfortunately, traceability matrices are often not built, or not to the level of detail required, during the development effort. As a result, they must be built after the fact by non-developers. Tools and techniques to assist with the automation of this time consuming, highly error prone, unpleasant task are needed.

On a positive note, in the last three to five years, there have been an increased number of research papers in the area of requirements tracing. Specifically, many of these papers [1, 11, 8, 7] apply information retrieval (IR) methods to the requirements tracing problem in a variety of settings. In particular, Information Retrieval methods are used to compare the texts of a pair of requirements from two documents in the project document hierarchy for the purpose of determining their similarity. This work uses well-accepted IR measures of recall and precision to evaluate the effectiveness of their techniques. Recall is the percentage of actual matches that are found and precision is the percentage of correct matches as a ratio to the total number of candidate links

* This work is funded by NASA under grant NAG5-11732.

returned [7]. Beyond the use of these two measures, the papers in this field have little in common. As more and more traceability studies become available, the need for a clearly outlined experimental framework that allows side-by-side comparison is emerging. Fenton, Pfleeger, and Glass [5] point out that far too often software engineering researchers rely on intuition and not empirical research and data. Basili, Shull, and Lanubile [4] examine a number of papers related to software engineering experimental frameworks. They note: "the important common characteristic of all these frameworks is that they document the key choices made during experimental design, along with their rationales." Further, the frameworks allow the comparison of studies and "allow the primary question of an experiment to shift from 'Is a particular process effective?' to 'What are the factors that make a particular process effective or ineffective?' [4]."

The need for such a framework in software engineering experimentation has been acknowledged. Similarly, the importance of applying such a framework to experiments of defect-detection techniques has also been demonstrated [10]. There is a need for a framework or structure for empirical studies on requirements tracing.

We propose **a framework for developing, conducting and analyzing experiments on requirements traceability**. This framework allows description of various existing and emerging research on requirements tracing and traceability. The contribution of the framework is to help achieve the goal of an infrastructure for experimental software engineering experiments that evaluate requirements tracing techniques.

## 3. Proposed framework

The framework builds on work that appeared in [6, 10, 3]. The framework encompasses **definition** of the experimental study, **planning** of the study, **realization** of the study, and **interpretation** of the study.

### 3.1 Definition

**Definition** refers to the project definition phase, when a researcher decides the scope and objective of the project. There are several parts to the definition phase, including *motivation*, *purpose*, *object*, *hypothesis*, *perspective*, *domain*, *scope*, and *importance*.

There may be many *motivations* for an experiment on requirements tracing techniques. Researchers may be seeking to, for example, understand why certain elements are never traced to any other elements, or confirm results that were seen on a previous experiment, or assess a specific measure (e.g., recall) for a particular requirements tracing technique.

The *purpose* of an experiment may include testing a tool or algorithm, or evaluating the effectiveness of a model or technique. Other examples include, but are not limited to understanding a process better; assessing the compliance of a tool or technique with a process, guideline, or criterion; validating the results of a previous experiment.

The *object* of study will generally be a product or model, although some experiments will examine the requirements tracing process or the usefulness of a posited metric.

The *hypothesis* (or hypotheses) should be stated in such a way as to be verifiable. The premise of the researcher, usually that "our new requirements tracing technique is better than someone else's technique as shown by higher recall and precision," will be stated as the alternative hypothesis. The null hypothesis will be that no difference between the requirements tracing techniques exists.

Though most experiments are from the *perspective* of the researcher, they may be from many other perspectives such as maintainer, customer, user, or manager.

The *domains* that typically apply to experiments are user engineers who will be employing the requirements tracing techniques, or projects or programs on which the techniques will be applied.

Basili et al [3] classify experimental study *scopes* by looking at the size of the domains considered, as does this experimental framework. Some example combinations are (1) one or more objects across a set of teams and a set of programs; (2) objects across a set of teams and a single program; (3) objects across a single team and a set of programs, or (4) objects on a single team and a single program.

We distinguish two levels of *importance*: domain importance and object of study importance. In both cases, the importance is being evaluated on the following scale:

- safety-critical,
- mission-critical,
- quality of life, or
- convenience [9].

### 3.2 Planning and Realization

The experiment **planning** phase consists of three parts. Experimental *design* - experiments should be designed in such a way as to maximize internal and external validity, while evaluating the hypotheses. *Measurement* requires specification of the definition, validation,

collection (automatable or not), objectivity, and scale (nominal/classificatory, ordinal, interval, or ratio)[3] of metrics. The planning *product* section covers documentation, code, databases, and other artifacts -- in traceability experiments, the products are usually the items that are being traced while a model or process is being evaluated.

The **realization** phase is the time when the experiment is conducted. There are three parts to the realization phase: (1) *preparation*, which may include pilot studies, parsing requirements, defining expected results, building lookup tables, and converting data; (2) *execution* of data collection and validation; and (3) *analysis* that may include data analysis, plots and histograms, model assumptions, primary data analysis, and model application [9].

### 3.3 Interpretation

**Interpretation** refers to the phase when the researcher derives a result from the experimental study. There are four parts to the interpretation phase: (1) *interpretation context*; (2) *extrapolation*; (3) *results*; and (4) *impact*.

*Interpretation context* is the environment/circumstances that must be considered when interpreting the results -- possible contexts are (i) statistical, (ii) framework, (iii) study purpose, or (iv) field of research.

*Extrapolation* deals with sample representativeness. In most cases, the issue of concern for tracing experiments is the representativeness of the projects and artifacts examined with the tracing technique.

We separate *results* of the studies into two categories: hypothesis evaluation and acquisition of additional knowledge. We expect the primary result of any study to be either confirmation or rejection of the null hypothesis. In addition, a research study might lead to acquisition of some new knowledge

*Impact* pertains to the level of effect that a study has on a field of research and/or industry. Possible impacts include, but are not restricted to replication of the experiment by others, replication of another study, application of the results in industry, and visible publishing/presenting of the results [9].

## 4. Applying the framework

When we view four studies [1, 11, 2, 7] within the framework:

- all are performed from the researcher's *perspective*
- the *objects* of study are algorithms and models, or processes
- the *domain* is usually a program
- *hypotheses* can be determined
- all shared common *motivation* and *purpose*
- the *scope* of the studies was evenly divided between single projects and blocked subject-project, and
- the domain *importance* covered all but one of the likely choices, yet object importance was quality of life for all studies.

The independent variables varied, but not significantly. Three of the four studies examined traceability models or algorithms. The dependent variables were very similar for all studies with the exceptions being the addition of average similarity and performance for two of the studies. The product was the same for two of the studies. Preparation involved preparation of artifacts for all four studies, though the artifacts varied from open source artifacts to industry proprietary code and models. Interpretation context, results, and extrapolation were the same for all four studies. Impact ranged from studies that replicated other studies to studies whose results and tools are being utilized by industry now [9].

## 5. Observations

From the above, several observations can be made. First, it appears that other purposes might be considered when planning studies. For example, researchers might test specific tracing tools, improve existing algorithms, etc. Second, other objects might be studied. For example, a comprehensive study of metrics and their use/meaning/usefulness w.r.t. evaluation of tracing processes might be warranted, especially considering that new metrics [8] have been proposed recently. From the examination of the hypotheses, it appears that VSIR and PIR should be considered as baseline tracing methods for comparison purposes. It does not appear that methods such as grep need be examined further. Manual tracing methods cannot be dismissed though, as we require these for the human judgment task of the tracing process.

It appears that other perspectives should be considered in future traceability studies, such as project manager, developer, or customer. Studies should be undertaken that have safety-critical domain importance. Products of the studies are already diverse, but should explore other areas too such as non-textual artifacts. We should strive for all of our studies to be replicated and for the technologies under study to be adopted by industry.

As pointed out above, it appears that this field of research is emerging. To understand how we might move forward, let us consider the characteristics of a more mature field of software engineering, such as

reading techniques. There, experimental studies are performed from numerous perspectives, such as project manager or maintainer. Studies have moved beyond baseline method comparisons to comparison of field-tested, proven techniques. In particular, the methods have been field tested and proven and have often been implemented in "productized" tools used in industry. Many studies have been replicated. Industry has adopted many of the studied techniques and tools. A community of researchers studying these techniques has been formed and is successfully collaborating with practitioners in the field.

Based on this brief analysis, the work that is before us in the requirements tracing and traceability research area is clear. We need to move beyond baseline methods such as vector space model. We need larger, standardized, more robust datasets (with *answersets*) available for study. We need to study the human factors associated with the tracing process (study from different perspectives, study different objects, study with different motivations and purposes). Finally, we need industry to be more actively involved with tracing/traceability research to facilitate large-scale studies of the human factors in tracing [9].

## 6. Conclusions and future work

In this paper we set forth our position that a framework for comparing requirements tracing experiments is needed. We presented a framework for characterizing experiments that examine requirements tracing techniques. The framework should assist researchers in developing and conducting additional experiments of this type. It also facilitates the comparison of results from similar experiments. We used the framework to describe and compare four recent experimental studies. We used the framework to identify areas for future research as well as for future experimentation. We also identified suggestions for moving tracing research from an emerging field to a more mature field.

We have been actively pursuing these suggestions in our own work. We have developed a prototype tool that is being used by industry. We have experimented on a number of new, larger programs. We have developed additional measures. We plan to enhance the prototype tool that we have developed in order to productize it, and we plan to conduct human factors studies.

To encourage the replication of the experiment performed at the University of Kentucky, the dataset used along with the answer set has been posted on the Software Engineering Experimentation Web (SEEWeb) hosted by George Mason University at http://ise.gmu.edu:8080/ofut/jsp/seeweb/index.jsp.

Though the experiments presented here all achieved fairly consistent results in terms of recall and precision, replication of experiments can only serve to strengthen the results.

## 7. Acknowledgements

Our work is funded by NASA under grant NAG5-11732. Our thanks to Ken McGill, Stephanie Ferguson, Pete Cerna, Dave Pruett, Mike Norris, Bill Gerstenmaier, Bill Panter, the International Space Station project, Mike Chapman and the Metrics Data Program, and the MODIS project for maintaining their website that provides such useful data.

## 8. References

1. G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering Traceability Links between Code and Documentation. *IEEE Transactions on Software Engineering*, 28(10):970-983, 2002.
2. G. Antoniol, B. Caprile, A. Potrich, and P. Tonella. Design-Code Traceability for Object Oriented Systems. *Annals of Software Engineering*, 9:35--58, 1999
3. V. Basili, R. Selby, and D. Hutchens. Experimentation in Software Engineering. *IEEE Transactions on Software Engineering*, 12(47):733-743, 1986.
4. V. Basili, F. Shull, and F. Lanubile. Building Knowledge through Families of Software Studies: An Experience Report. *IEEE Transactions on Software Engineering*, 25(4)456-473, 1999.
5. N. Fenton, S. Pfleeger, and R. Glass. Requirements Tracing. *IEEE Software*, July 1994.
6. J. Huffman Hayes. Energizing Software Engineering Education through Real-World Projects as Experimental Studies, In *Proceedings of the Conference on Software Engineering Education and Training, CSEET*, 2002.
7. J. Huffman Hayes, A. Dekhtyar, and J. Osbourne. Improving Requirements Tracing via Information Retrieval, in *International Conference on Requirements Engineering, Monterey, California*, pages 151-161, 2003.
8. J. Huffman Hayes, A. Dekhtyar, S. Sundaram, and S. Howard. Helping Analysts Trace Requirements: An Objective Look. In *International Conference on Requirements Engineering (RE'2004)*, 2004.
9. J. Huffman Hayes and A. Dekhtyar. A Framework for Comparing Requirements Tracing Experiments, *University of Kentucky Technical Report, TR 408-04*, June 2004.
10. C. M. Lott and H. D. Rombach. Repeatable software engineering experiments for comparing defect-detection techniques. *Empirical Software Engineering: An International Journal*, 1(3):241-277, 1996.
11. A. Marcus and J. Maletic. Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing. In *Proceedings of the Twenty-Fifth International Conference on Software Engineering, ICSE 2003*, pages 125-135, 2003.
12. J. Matthias. Requirements Tracing. *Communications of the ACM*, 41(12), 1998.