On the Effects of Traceability Links in Differently Sized Software Systems

Muhammad Atif Javed and Uwe Zdun
Software Architecture Research Group
University of Vienna, Austria
muhammad.atif.javed|uwe.zdun@univie.ac.at

ABSTRACT

Context: None of the published empirical studies on software traceability have comparatively examined the support for differently sized systems. **Objective:** This paper reports on two controlled experiments performed with two Enterprise Service Bus (ESB) systems that are comparable in terms of support features and system structure, but are different in their size, in particular, UltraESB Version 2.3.0 and PetalsESB Version 4.2.0, to investigate the effects of system size on the use of traceability links. **Method:** We conducted two controlled experiments in which the same impact evaluation activities were performed and measured how the control groups (provided with no traceability information) and the experiment groups (provided with traceability information) performed these activities in terms of the quantity and quality of retrieved elements. **Results:** Our findings show that the 133.71% larger size of one of ESBs does not have a significant influence on the quantity and quality of retrieved elements in the experiment groups. In the control groups, in contrast, this increase in system size significantly increases the quantity of incorrect elements and reduces the overall quality of elements retrieved, while no conclusive evidence concerning the quantity of missing elements was found. **Conclusion:** It is concluded that traceability is more important in larger software systems.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Maintenance, and Enhancement; D.2.9 [**Software Engineering**]: Management; D.2.11 [**Software Engineering**]: Software Architectures

General Terms

Design, Management, Documentation

Keywords

Traceability, Differently Sized Systems, Impact Analysis, Empirical Software Engineering, Controlled Experiment

EASE '15, April 27 - 29 2015, Nanjing, China

1. INTRODUCTION

Traceability has been defined in the IEEE glossary of software engineering as "the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship with each other" [17]. Traceability relations between software development artefacts are a solution often suggested to help people understand software systems and support their reuse, maintenance, evolution and quality control [8, 9, 11, 18]. The use of traceability relations is considered more important in larger software systems. To date, however, none of the published empirical studies on software traceability have comparatively examined the effects of system size on the use of traceability links.

Our previous results imply that using traceability links leads to significantly lower quantity of missing and incorrect elements in understanding activities, and overall, a higher quality of change impact analysis for evolution [12]. In this paper, we provide an additional contribution to the body of empirical knowledge on software traceability by investigating the support provided by traceability links for differently sized software systems. In particular, we compare results in terms of quantity and quality of the two control groups and the two experiment groups from two controlled experiments. In those experiments the same tasks have been performed (as shown in Table 3) for two Enterprise Service Buses (ESBs), i.e. very similar software systems, of different size. In particular, we intend to answer the following research question: Are the quantity and quality of the retrieved elements in impact evaluation activities differently influenced by the size of the software systems under study when making use of traceability links than when they are not used?

To answer this research question, we conducted two controlled experiments at the University of Vienna, Austria in May 2014. The experiments were conducted with two differently sized ESB systems, in particular, UltraESB Version 2.3.0 and PetalsESB Version 4.2.0, that consists of 709 and 1657 source code classes, respectively. In total, 107 students of the software architecture course took part: The first experiment was carried out with 51 students, whereas the other 56 students participated in the second experiment. The participants were asked to perform seven impact evaluation activities that are used for both software systems. This was possible as the two different ESBs share many similar functionalities and also have many correspondences in their designs and architectures.

In both experiments, one half of the students were

assigned to the control group (without traceability links), while the other half was assigned to the experiment group (with traceability links). The data from the experiments was analysed, and the quantity of missing and erroneous retrieved elements and their overall quality were compared. The results from the experiments demonstrate that using traceability links leads to a slight difference in the quantity and quality of retrieved elements for a 133.71% larger software system. This difference in the absence of traceability links, in contrast, significantly increases the quantity of incorrect elements and reduces the overall quality of elements retrieved in the larger software system, while no conclusive evidence concerning the quantity of missing elements was found.

The rest of this paper is organized as follows: Section 2 describes the related work. Section 3 discusses the design of the controlled experiments including the introduction of variables and hypotheses, while the subsequent Section 4 explains the details concerning the execution of the experiments. Section 5 presents the hypotheses tested and the analysis of the results of the study. Section 6 contains the interpretation of the findings and a discussion of threats to validity. Section 7 concludes the study and discusses future work.

2. RELATED WORK

Several empirical studies have been performed to evaluate the added value of traceability links. This section broadly distinguishes between traceability links based on IR-based traceability recovery tools, traceability visualization tools and empirical investigations without any specific traceability tool.

2.1 Empirical Investigations on IR-Based Traceability Recovery Tools

De Lucia et al. [5, 6], Cuddeback et al. [4] and Dekhtyar et al. [7] investigate the usefulness of IR-based traceability recovery tools. The first experiment of De Lucia et al. [6] was carried out with 16 master students, who had to perform two traceability recovery tasks on a development project, with and without the COCONUT tool. It focuses on the identification of traceability links between requirements and the source code. The results indicate that COCONUT significantly improves the similarity between source code and related requirements in presence of comments, while a practical improvement was also detected without considering comments. In a second study, De Lucia et al. [5] conducted a controlled experiment and a replication to investigate the support provided by ADAMS Re-Trace between use cases and the source code, and between interaction diagrams and test cases. The study involves 32 master students. The students were asked to perform two traceability recovery tasks, with and without ADAMS Re-Trace, on a software repository of a completed project. The results show that ADAMS Re-Trace significantly reduces the time spent on identifying trace links and improves the tracing accuracy of the software engineer.

Cuddeback et al. [4] and Dekhtyar et al. [7] evaluate human analyst performance in IR-based traceability recovery tools. Cuddeback et al. [4] conducted an experiment in which the subjects had to prepare the final set of traceability links, with different recall precision possibilities in the RETRO tool, on a project assignment carried out with 26 students of computer science. They focused on the traceability links between requirements and test cases. Their findings show that the participants failed to finalize the correct traceability links, while the participants provided with the lower recall and precision of traceability links make significant improvements. In addition, regardless of size and accuracy of the initial traceability links, the participants tend to guess the correct number of traceability links. Dekhtyar et al. [7] performed two more follow-up experiments to investigate usability and stability issues in the RETRO tool and prepare the final set of traceability links without tool assistance. The experiments were conducted with a total of 84 students. The results demonstrate that the accuracy of initial traceability links and time spent had significant interaction with the final traceability links, whereas no significant differences with regard to the tool used, effort applied in searching for missing links and traceability experience are observed.

2.2 Empirical Investigations on Traceability Visualization Tools

The works by Cornelissen et al. [3] and Shahin et al. [19] investigate the support provided by traceability visualization tools. Cornelissen et al. analyse the support provided by the ExTraVis tool, which offers two interactive views, the sequence view and the circular bundle view of large execution traces, for program comprehension. The experiment was carried out with 32 participants. The participants were asked to perform eight typical tasks aimed at gaining an understanding of a representative subject system and measured how a control group (using the Eclipse IDE) and an experiment group (using both Eclipse and ExTraVis) performed these tasks in terms of time spent and solution correctness. The results show that ExTraVis significantly decreases the time requirements and increases the correctness for program comprehension.

Shahin et al. investigate the usefulness of Compendium, a tool to visualize architectural design decisions and their rationale, as a kind of traceability information. The experiment involves 10 participants. The participants were asked to understand the existing design and to make the new design according the new requirement, with and without the Compendium tool. The results show that Compendium significantly improves the correctness of understanding architecture design in the architecting process and does not increase the total time for reading software architecture documentations and performing design task.

2.3 Empirical Investigations Without Any Specific Traceability Tool

Mader and Egyed [15] and Javed and Zdun [10, 12] analyse the support provided by traceability links without any specific tool. Mader and Egyed [15] conducted a controlled experiment to evaluate the support provided by traceability links between requirements and the source code. The experiment was conducted with 52 students of computer science. They were asked to perform eight maintenance tasks, half of the tasks with and the other half without traceability information, on two third-party development projects. The results demonstrate that requirement traceability saves downstream cost and can profoundly improve software maintenance quality.

In our own previous studies [10, 12] we conducted four controlled experiments to evaluate the support provided

by traceability links between architectural models and the source code. The first two experiments were carried out with 108 participants from industry and academia [10]. The participants were asked to answer twelve typical questions aimed at gaining an architecture-level understanding of a representative subject system, with and without traceability information. The findings show that the use of traceability links significantly increases the correctness of the answers of the participants, whereas no conclusive evidence concerning the influence of the experience of the participants are observed. Another two experiments were conducted with 107 students of the software architecture course using two different systems [12]. The students were asked to perform seven impact evaluation activities for the provided software systems, with and without traceability information. The results provide statistical evidence that a focus on traceability links significantly reduces the quantity of missing and incorrect elements, and increases the overall quality of architecture impact analysis for evolution. The major difference of this paper to our previous results is that the quantity and quality of retrieved elements in UltraESB Version 2.3.0 and PetalsESB Version 4.2.0 is investigated individually [12].

The contribution of this study is novel for two main reasons. First, there exist no comparative evidence related to the added value of traceability links in differently sized software systems. Second, most of the earlier works are based on some specific traceability tools, which does not enable a clear distinction between tool support and the usefulness of traceability links. In our experiments, for practical reasons and to study the foundational concepts rather than a specific tool, the participants were provided with hyperlink-based access of traceability links and the source code, to investigate the support provided by traceability links in impact evaluation activities of differently sized software systems, rather than the support provided by a specific tool.

3. DESIGN OF THE EXPERIMENT

For the design of the experiments, the guidelines for experiments by Kitchenham et al. [14] and Wohlin et al. [21] were used. The former present general guidelines for experimentation in software engineering and give some instructions concerning the context, design, data collection, analysis, presentation, and interpretation of empirical studies without going into detail. The latter present the experiment phases in more detail, and also discuss statistical tests and their suitability for different kinds of studies. Kitchenham et al.'s guidelines were primarily used in the planning phase of the experiments, while Wohlin et al.'s guidance was used as a reference for the analysis and interpretation of the results. The guidelines for reporting controlled experiments by Jedlitschka and Pfahl [13] are used to describe the experiments in this paper. Note that the subsequent subsections of the reporting template were omitted, because they were either not applicable, or their content was already mentioned in other sections: Relation to existing evidence is presented in Section 2; inferences and lessons learned are discussed in Section 6; interpretation and general limitations of the study are described in Section 6.2. The usage of this reporting template, however, introduces a certain level of redundancy, because a distinction between the design and the actual execution of the experiments is made.

3.1 Goal, Hypotheses, Parameters, and Variables

The goal of the experiments is to find out, if traceability links lead to lower difference in the quantity and quality of retrieved elements for differently sized software systems compared to the quantity and quality of elements that are retrieved without using traceability information. The experiment's goal led to the following null hypotheses and corresponding alternative hypotheses:

H_{01}: The use of traceability links leads to *higher difference* in the quantity of correctly retrieved elements for the larger software system compared to retrieval without traceability information.

H_1: The use of traceability links leads to *lower difference* in the quantity of correctly retrieved elements for the larger software system compared to retrieval without traceability information.

H_{02}: The use of traceability links leads to *higher difference* in the quantity of incorrectly retrieved elements for the larger software system compared to retrieval without traceability information.

H_2: The use of traceability links leads to *lower difference* in the quantity of incorrectly retrieved elements for the larger software system compared to retrieval without traceability information.

H_{03}: The use of traceability links leads to *higher difference* in the overall quality of retrieved elements for the larger software system compared to retrieval without traceability information.

H_3: The use of traceability links leads to *lower difference* in the overall quality of retrieved elements for the larger software system compared to retrieval without traceability information.

Description	Scale Type	Unit	Range
Quantity of correctly retrieved elements	Interval	Points	[0 - 1]
Quantity of incorrectly retrieved elements	Interval	Points	[0 - 1]
Overall quality of the retrieved elements	Interval	Points	[0 - 1]

Table 1: Dependent Variables

3.1.1 Dependent Variables

Three dependent variables were observed during the experiments, as shown in Table 1: the quantity of correctly and incorrectly retrieved elements, and their overall quality in the differently sized software systems. They were accessed by using the recall, precision, and F-measure, respectively. Because analysis and evaluation of software systems involves a list of system elements, two aspects were specifically taken into consideration to measure the recall and precision of the retrieved elements:

- The set of *correct elements* expected in the solution to *activity a* (C_a).
- The set of *elements retrieved* in the solution to *activity a* by *participant p* ($R_{p,a}$).

$$Recall_{p,a} = \frac{\mid C_a \cap R_{p,a} \mid}{C_a} \qquad Precision_{p,a} = \frac{\mid C_a \cap R_{p,a} \mid}{R_{p,a}}$$

Recall is the percentage of correct matches retrieved by an experiment subject, while precision is the percentage of retrieved matches that are actually correct. Because recall and precision measure two different concepts, it can be difficult to balance between them. Therefore, f-measure, a standard combination of recall and precision, defined as their harmonic mean, is used to measure the overall quality of retrieved elements from the experiments' participants.

$$F - measure_{p,a} = 2 * \frac{recall_{p,a} * precision_{p,a}}{recall_{p,a} + precision_{p,a}}$$

3.1.2 Independent Variables

The goal of the experiments was to discover the influence of traceability links on the quantity and quality of elements obtained from differently sized software systems. Therefore, two different treatments were defined for the participants in each experiment: One group of participants was explicitly told to perform the impact evaluation activities by using the information from the architectural documentation and the source code of the system. The participants in the other group performed the same activities, but additionally received the traceability links between architectural models and the source code. The first group is referred to as the control group, the latter as the experiment group. Note that the first experiment was conducted with a rather small system (UltraESB), while the second experiment has been performed with a larger system (PetalsESB).

Table 2 shows five independent variables that could have an influence on the dependent variables. They relate to the personal information (programming experience, architecture experience, affiliation), group affiliation (control group or experiment group) and time spent in the experiments. In the design of the study, these variables were eliminated by balancing the characteristics between the control groups and the experiment groups in both experiments.

Description	Scale Type	Unit	Range/Possible Values
Time	Ordinal	Minutes	90 minutes (Max)
Group Affiliation	Nominal	N/A	Control group, Experiment group
Programming experience	Ordinal	Years	4 classes: 0-1, 1–3, 3–7, >8
Architecture experience	Ordinal	Years	4 classes: 0-1, 1–3, 3–7, >8
Affiliation	Nominal	N/A	Academia, Industry, Other

Table 2: Independent Variables

3.2 Experiment Design

To test the hypotheses, we conducted two controlled experiments [1] at the University of Vienna, Austria, in May 2014.

3.2.1 Participants

The participants in the experiments were 107 individual students of the software architecture course held at University of Vienna. The first experiment was conducted with 51 students, while the other 56 students had participated in the second experiment. All the students had knowledge of software development, software architecture, as well as of software traceability.

3.2.2 Objects

The basis for the impact evaluation of differently sized software systems was UltraESB Version 2.3.0 and PetalsESB Version 4.2.0. These systems belong to the ESB domain. They provide an connectivity infrastructure to integrate the services within a service-oriented architecture. The choice of using the particular objects is motivated by the following factors:

- The UltraESB and PetalsESB are free open source systems, which enables us to conduct the experiments and disseminate their results.
- These systems are based on industry standards; that is, they have industrial relevance.
- The domain of the systems is well-known to the participants from previous lectures of at least the software architecture course.
- The UltraESB and PetalsESB are written in the Java programming language with which the participants were sufficiently familiar.
- The source code of the UltraESB and PetalsESB adheres to coding standards and is rather easy to understand for most potential subjects.
- The overall source code of the UltraESB and PetalsESB consists of 709 classes and 1657 classes, respectively; that is, both of them are not too large to be studied in the limited time-frame of the experiments' sessions and sufficiently different for performing two independent experiments.
- The UltraESB and PetalsESB belong to the same domain, are very similar in structure and implemented features, but significantly differ with respect to their sizes. This enables us to find out whether the larger size of PetalsESB has a positive or negative impact on the study results.
- The experimenters were familiar with the internals of the UltraESB and PetalsESB.

3.2.3 Blocking

To be able to explicitly analyse the influence of traceability links in differently sized software system, the participants in both experiments were randomly assigned to the two balanced groups. For each experiment, one group of participants was asked to answer the impact evaluation activities by using the information from the architectural documentation and the source code of the system, whereas the other group performed the same tasks, but additionally received the traceability links between architectural models and the source code. The first group is referred to as control group, the latter as experiment group.

3.2.4 Instrumentation

To obtain the necessary data related to the influence of traceability links in differently sized software system, the instruments discussed in the following paragraphs were used to carry out the experiments.

Three pages of architectural documentation about the used objects.

The participants in the first experiment were provided with the documentation for UltraESB, while the participants of the second experiment received the documentation for PetalsESB. The documentation describes the conceptual architecture and lists technologies and frameworks used in

the implementation. Besides text, a UML component diagram is used to illustrate the components, and their interrelationships in parts of the architecture.

Web-based access for the source code.

The participants in the first and second experiment were provided with the web-based access of syntax-highlighted source code for the UltraESB and PetalsESB, respectively. The cover page alphabetically lists the source code package names and their enclosed code classes, and provides a hyperlink-based support to 'jump' to specific elements (code classes or packages) located in the Git repository[1]. The participants in the experiment groups were also provided with the similar support for traceability links, represented as lists: Each entry in a list contains information about architectural components and their realized code classes, which represent individual traceability link. It may be noted that two doctoral researchers established the traceability links (including the first author of the paper) that are later validated by two other doctoral researchers (also referred to as experimenters) and a professor (the second author in the paper).

A questionnaire to be filled-in by the participants during the experiment.

At the first page of the questionnaire, the participants had to rate their programming experience, architecture experience and affiliation, while the subsequent pages contains the seven impact evaluation activities, as shown in Table 3. In the context of these activities, two important criteria are applied: (i) the activities should be representative for key impact analysis and evolution contexts for both UltraESB and PetalsESB, and (ii) they should be imaginatively constructed to measure the deeper impact understanding from participant groups. Note that the same impact evaluation activities listed in Table 3 were used for both UltraESB and PetalsESB, which was possible as different ESBs share many implemented features and also have similar design structures. The results expected from participants for each activity were sets of retrieved element names (i.e., names of individual source code classes or packages and components from the provided component models).

ID	Description
A1	Investigate the impact of extensions in the transport senders and listeners
A2	Investigate the consequences of extensions in the traffic monitoring
A3	Determine the ripple-effects of changes in the ESB configuration
A4	Investigate the impact of changes in the message interception
A5	Evaluate the effects of high availability and capacity of ESB server
A6	Investigate the consequences of new message endpoints
A7	Determine the impact of new deployment aspect implementation

Table 3: Impact Evaluation Activities (Used for Both UltraESB and PetalsESB)

3.2.5 Blinding

To eliminate subjective bias on the part of both experiments' participants and the experimenters, double blinding was applied in the experiments. Although, participants perceived that there are two different groups for each experiment, they were not aware about the purpose of group division and their group affiliation.

The results of the experiments were handed over to two independent researchers who did not know the real identity of the participants. This was done to prevent the experiments from being biased. To be able to compute the results for the quantity and quality of retrieved elements, the researchers were asked to compute the information retrieval statistics by matching the participants' answers with the original solution model. This allows us to objectively evaluate the retrieved elements for differently sized software systems rather than by intuitive or ad-hoc human measures.

3.2.6 Data Collection Procedure

After introduction and grouping, the participants received the instruments, mentioned in Section 3.2.4. The provided instruments had to be used to perform the impact evaluation activities. The participants were distributed over separate rooms according to their group membership. At least one experimenter was present in each room to answer the questions related to the instructions and to restrict the participants from consulting others and using forbidden material. The participants were given 90 minutes to perform the impact evaluation activities. After completion of the session, the filled-in questionnaires were collected by the experimenters and finally a discussion in the wrap-up phase was arranged to gather further information from the participant groups. All the participants were present during the discussion.

The discussion questions were concerning difficulties in the experiments and influence of traceability links in performing the impact evaluation activities. While this information is not required for testing the hypotheses, it is used for validation and interpretation of the results. The questions after the experiments are asked for three reasons: 1) because the traceability links between architectural models and the source code could have influenced the participants of the control groups prior to the experiments; 2) because we are interested in a way the participants performed the impact evaluation activities for differently sized software systems; 3) in the light of discussion, the participants of the control groups could have guessed that traceability links played an important role in the experiment groups, and consciously or unconsciously also focused on the traceability links.

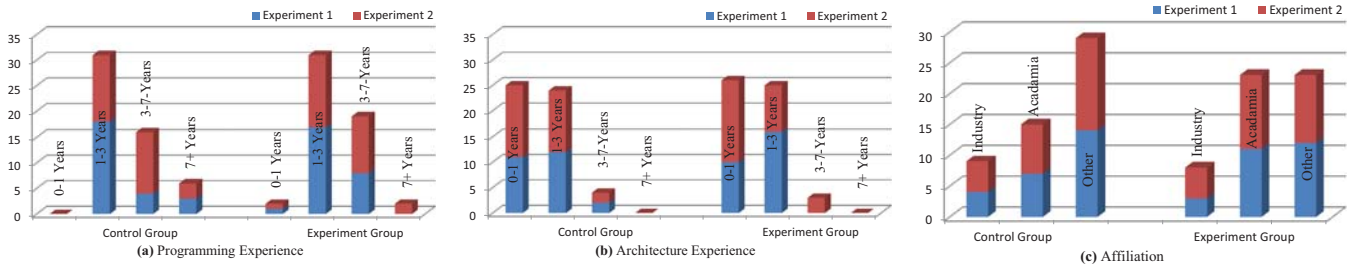
4. EXECUTION

4.1 Sample and Preparation

As described in Section 3.2, the experiments were conducted in two practical sessions at the University of Vienna, Austria. The first experiment took place with 51 students of the software architecture course; the second experiment was conducted with another 56 students of the same course.

Figure 1 shows the distribution of the participants based on their previous experience and affiliation, as assigned to the control group and the experiment group. The data presented in the figures was accumulated from all the participants in the two experiments, but also shows the separate data of the experiments. The Sub-figures (a) and (b) show the previous experience of the participants concerning programming and software architecture, while Sub-figure (c) shows the affiliation of the participants. Note that the previous experiences in the control groups is slightly better both regarding programming and architecture. In the experiment groups slightly more people with an academic affiliation and slightly less with an industry affiliation are present. However, overall the experiences and affiliations are rather well balanced in the two experiments.

[1]http://git-scm.com



(a) Programming Experience (b) Architecture Experience (c) Affiliation

Figure 1: Distribution of Participants

4.2 Data Collection Performed

The data collection procedure was performed as planned in the study design. There were no participants who dropped out and no deviations from the study design occurred.

4.3 Validity Procedure

The experiments were conducted in a controlled environment. The participants in both experiments were assigned to different rooms according to their group membership (control group or experiment group). The participants in each rooms were supervised by at least one experimenter during the whole duration, enabling them to ask clarification questions and restrict them from talking to each other or using forbidden material. All the participants had to return the questionnaire before leaving the room. The filled-in questionnaire were collected from the remaining participants after completion of experiments' sessions. No unexpected situation occurred during the experiments.

5. ANALYSIS

5.1 Descriptive Statistics

The descriptive statistics shows the results of the experiments as a first step in the analysis. The first two subsections concern the quantity of correctly and incorrectly retrieved elements respectively. The last subsection presents an analysis of the overall quality of retrieved elements in the two experiments.

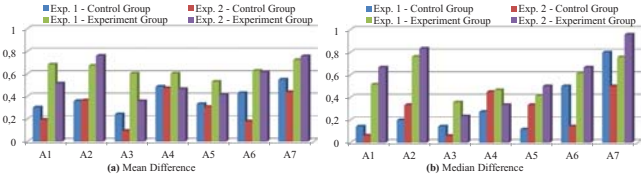
Group Affiliation	Execution	Mean	Median	Std. Dev.
Control Group	Experiment 1	2.701009 (0.3858584 %)	2.565584 (0.3665121 %)	1.808094 (0.2582992 %)
	Experiment 2	2.04751 (0.2925014 %)	1.908818 (0.2726883 %)	1.023914 (0.1462735 %)
Experiment Group	Experiment 1	4.439981 (0.634283 %)	4.868956 (0.6955651 %)	1.708463 (0.2440661 %)
	Experiment 2	4.114883 (0.5878404 %)	4.491484 (0.6416406 %)	1.421371 (0.203053 %)

Table 4: Descriptive Analysis of the Quantity of Correct Retrieved Elements

5.1.1 Quantity of Correctly Retrieved Elements

The descriptive statistics for the quantity of correctly retrieved elements for the control groups and the experiment groups from the two experiments are shown in Table 4 and Figure 2. The data in the table is based on the sum of the recall measures of the experiments' activities for each participant, while the figure concerns the recall measures for each experiment activity.

As we see from Table 4, the total difference in the quantity of correctly retrieved elements is lower in the experiment groups than in the control groups. This indicates that the quantity of correctly retrieved elements is less effected in the larger software system due to the traceability links. The results in Figure 2 show that the participants of the experiment group belonging to the second experiment have only outperformed the participants of the control group of the first experiment in Activity 4. This means that the quantity of correctly retrieved elements in the larger system (PetalsESB) is higher for all the impact evaluation activities except activity 4 by focusing on traceability links compared to the quantity of correctly retrieved elements in the rather small system (UltraESB) that is performed without traceability information.



(a) Mean Difference (b) Median Difference

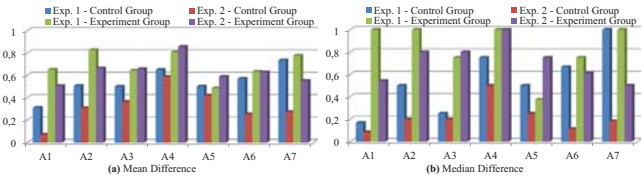
Figure 2: Quantity of Correctly Retrieved Elements for Each Experiment Activity

5.1.2 Quantity of Incorrectly Retrieved Elements

Table 5 and Figure 3 show the comparisons for the quantity of retrieved elements that are actually correct for the control groups and the experiment groups in the two experiments. The data in the table is based on the sum of the precision values of the experiments' activities for each participant. Note that the total difference in the quantity of retrieved elements that are actually correct is lower in the experiment groups than the control groups. As a consequence, this means that the quantity of incorrectly retrieved elements is higher in the larger software system when it is performed without traceability information.

Group Affiliation	Execution	Mean	Median	Std. Dev.
Control Group	Experiment 1	3.774333 (0.5391905 %)	3.208333 (0.4583333 %)	1.779565 (0.2542236 %)
	Experiment 2	2.278481 (0.3254973 %)	2.242929 (0.3204185 %)	1.191005 (0.1701435 %)
Experiment Group	Experiment 1	4.826603 (0.6895147 %)	4.6875 (0.6696429 %)	1.865917 (0.2665595 %)
	Experiment 2	4.454726 (0.6363894 %)	4.523485 (0.6462121 %)	1.593693 (0.2276704 %)

Table 5: Descriptive Analysis of the Quantity of Actually Correctly Retrieved Elements



(a) Mean Difference (b) Median Difference

Figure 3: Quantity of Actually Correctly Retrieved Elements for Each Experiment activity

The results in the Figure 3 concern the precision for each experiment activity, in which the participants of the experiment group of the second experiment only outperformed the participants of the control group of the first experiment

in Activity 7. It is interesting to note that the first experiment is conducted with 133.71% smaller system (UltraESB), while the second experiment was performed with larger system (PetalsESB).

5.1.3 Overall Quality of Retrieved Elements

The descriptive statistics for the overall quality of retrieved elements for the control groups and the experiment groups from the two experiments is shown in Table 6 and Figure 4. The results in the table are based on the sum of the overall quality of retrieved elements (i.e., the f-measure) of the experiments' activities for each participant, while the figure shows the f-measure results for each experiment activity. The data in the table and figure show that the average quality difference of retrieved elements in the experiment groups is lower than the average quality difference of retrieved elements in the control groups. The results also show that the participants of the experiment group of the second experiment (conducted with larger system – PetalsESB) have a higher overall quality for all impact evaluation activities than the participants of the control group of the first experiment (conducted with the rather small system – UltraESB).

Group Affiliation	Execution	Mean	Median	Std. Dev.
Control Group	Experiment 1	2.767399 (0.3953427 %)	2.516986 (0.3595694 %)	1.624837 (0.2321195 %)
	Experiment 2	1.755936 (0.2508479 %)	1.769355 (0.252765 %)	0.7923499 (0.1131928 %)
Experiment Group	Experiment 1	4.377607 (0.6253725 %)	4.275092 (0.6107274 %)	1.722879 (0.2461256 %)
	Experiment 2	3.66901 (0.5241442 %)	3.608125 (0.5154464 %)	1.617311 (0.2310445 %)

Table 6: Descriptive Analysis for the Overall Quality of Retrieved Elements

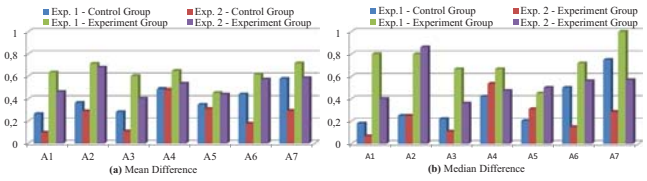


Figure 4: Overall Quality of Retrieved Elements for Each Experiment Activity

5.1.4 Dataset Reduction

Outliers in the dataset, i.e., data points that are either much lower or much higher than other data points, are potential candidates for dataset reduction. Thirteen of the participants from the two experiments did not perform all the activities. This results in nineteen missing data points in the experiments. As it seems that these participants have spend sufficiently longer time in exploring the source code, we have not excluded these data points from the study.

To find potential outliers, we also calculated the quantity and quality of the impact evaluation activities for each participant. Note that four of the participants from the experiment groups reached a considerable lower quantity and quality of retrieved elements than the other members of these groups. A closer analysis showed that they could not properly make use of traceability links to perform the impact evaluation activities. However, their results were not excluded as outliers, because the difference to the other participants is not strong enough. Excluding these data points would have introduced a potential vulnerability of the study results.

5.2 Analysis of the Opinion of Participants

This subsection summarizes the results of the wrap-up discussion phase which was arranged after each experiment session to gather further information from the participant groups.

The participants in the experiment groups from the two experiments and the control group of the first experiment have acknowledged that they had enough time to perform the impact evaluation activities. However, the participants in the control group of the second experiment showed concerns related to the provided time for performing the activities. This is because the second experiment was conducted with the 133.71% larger system (PetalsESB) compared to first experiment (UltraESB). The same happened also for the experience and difficulties of the participants: The participants of the control groups experienced more difficulties in performing the activities than the participants of the experiment groups, in addition, the participants with '0-1 years' of experience encountered more difficulties than the participants with '1-8+ years' of experience.

The participants were also asked about their familiarity with the application domain. The answers imply that enterprise service bus, which is the application domain of the UltraESB and PetalsESB, is well-known to the participants from previous lectures of the software architecture course.

The next two questions concerned the usage and helpfulness of traceability links for impact evaluation of the differently sized software systems. First, the participants were asked whether traceability links are useful in performing the provided activities. The answers reflect that the participants had knowledge about traceability links. The members of both groups generally consider traceability links as useful in analysis and evaluation of the software systems. In the next question the participants were asked whether they used traceability links before. The answers show that only a very few participants have previously used traceability links for understanding of software elements outside of the lecture in which the experiments took place.

Finally, the participants were asked to briefly describe how the impact evaluation activities were performed. This was primarily done to confirm that the experiment groups used the traceability links and to find out if the control groups used any other systematic way to perform the activities. The answers of the control groups reveal a focus on an intuitive approach, which was mainly driven by personal experience or judgments. The respondents stated that they performed the activities by reading the textual description in the architecture document and intuitively exploring the code classes. They acknowledged that it is hard to find the correct links between architecture and implementation artefacts. This might stem from the fact that software architecture is not explicitly represented in the code classes, for example, as packages and classes or similar code-level abstractions. The answers of the experiment groups show a focus on the traceability links. The respondents of the experiment groups stated that they used traceability links to identify the architecture artefacts in the code classes and vice versa. They confirmed that they primarily used this additional knowledge for performing the impact evaluation activities for the provided systems.

5.3 Hypothesis Testing and Results

5.3.1 Quantity of Correctly Retrieved Elements

To be able to test the first null hypothesis H_{o1}, the quantity of correctly retrieved elements in the control groups

and the experiment groups is measured. In the analysis of the experiments, the Shapiro-Wilk normality test [20] and Wilcoxon Rank-Sum test [16] are used. First, the Shapiro-Wilk normality test is used to find out whether equal variances of the level of correctness can be assumed. Second, as a consequence of non-normal distributions, the corresponding non-parametric test, Wilcoxon Rank-Sum test, is used to test the significance of the found results. Note that the results of tests were interpreted as statistically significant at $\alpha = 0.05$ (i.e., the level of confidence is 95%).

Group Affiliation	Factor	Wilcoxon Rank-Sum Test
Control Group	Experiment 1 vs. Experiment 2	W = 406, p-value = 0.3256
Experiment Group	Experiment 1 vs. Experiment 2	W = 406.5, p-value = 0.4672

Table 7: Wilcoxon-test for Quantity of Correct Retrieved Elements

The results from the Wilcoxon rank-sum test are shown in Table 7. It provides evidence that H_{o1} can be rejected. This means that the use of traceability links in our experiments leads to a lower difference in the quantity of correctly retrieved elements for the larger software system compared to retrieval without traceability links. It can be noted that we are unable to show that this result is significant when elements in the differently sized software systems are retrieved without traceability information.

5.3.2 Quantity of Incorrectly Retrieved Elements

Hypothesis H_{o2} was also evaluated with a Wilcoxon rank-sum test. The results are shown in Table 8. The table shows that the quantity of incorrectly retrieved elements in the control groups and the experiment groups provide strong evidence that H_{o2} can be rejected. This means that the use of traceability links in our experiments leads to a lower difference in the quantity of incorrectly retrieved elements for the larger software system compared to retrieval without traceability links. Moreover, it is noticeable to see that this result is significant when elements in the differently sized software systems are retrieved without traceability information.

Group Affiliation	Factor	Wilcoxon Rank-Sum Test
Control Group	Experiment 1 vs. Experiment 2	W = 522, p-value = 0.002244
Experiment Group	Experiment 1 vs. Experiment 2	W = 409.5, p-value = 0.4354

Table 8: Wilcoxon-test for Quantity of Incorrect Retrieved Elements

5.3.3 Overall Quality of Retrieved Elements

The Wilcoxon rank-sum test is also used to evaluate the Hypothesis H_{o3}. The results are shown in Table 9. The table shows that the difference in the control groups and the experiment groups provide strong evidence that H_{o3} can be rejected. This means that the use of traceability links in our experiments leads to a lower difference in the overall quality of retrieved elements for the larger software system compared to retrieval without traceability links. It is interesting to note that this result is also significant when elements in the differently sized software systems are retrieved without traceability information.

Group Affiliation	Factor	Wilcoxon Rank-Sum Test
Control Group	Experiment 1 vs. Experiment 2	W = 468, p-value = 0.03553
Experiment Group	Experiment 1 vs. Experiment 2	W = 445, p-value = 0.1647

Table 9: Wilcoxon-test for Overall Quality of Retrieved Elements

6. INTERPRETATION

6.1 Evaluation of Results and Implications

6.1.1 Quantity of Correct Retrieved Elements

Hypotheses H_{o1} and H_1 concern the influence of traceability links on the quantity of correctly retrieved elements for the larger software system. The results pointed out in Section 5 show that the null hypothesis H_{o1} can be rejected. Thus, according to our experiments, there is evidence that the difference in the quantity of correct retrieved elements is lower in the larger software system if traceability links are used compared to retrieval without traceability information.

The very small difference in the control groups is surprising to us because we have assumed that the quantity of correct retrieved elements is significantly lower in the larger system compared to the quantity of correctly retrieved elements in the rather small system. However, the participants of the control groups from the both experiments performed rather poorly.

6.1.2 Quantity of Incorrect Retrieved Elements

Hypotheses H_{o2} and H_2 concern the influence of traceability links on the quantity of incorrectly retrieved elements for the larger software system. The results pointed out in Section 5 provides strong evidence that the null hypothesis H_{o2} can be rejected. Thus, according to our experiments, there is evidence that the difference in the quantity of incorrectly retrieved elements is lower in the larger software system if traceability links are used compared to retrieval without traceability information.

6.1.3 Overall Quality of Retrieved Elements

Hypotheses H_{o3} and H_3 concern the influence of traceability links on the overall quality of retrieved elements for the larger software system. The results pointed out in Section 5 provides strong evidence that the null hypothesis H_{o3} can be rejected. Thus, according to our experiments, there is evidence that the difference in the overall quality of retrieved elements is lower in the larger software system if traceability links are used compared to retrieval without traceability information.

6.2 Threads to Validity and Limitations of the Study

Multiple levels of validity threats have to be considered in the experiments. We have considered the classification scheme for validity in experiments by Cook and Campbell [2]. The internal validity concerns the cause effect inferences between the treatment and the dependent variables measured in the experiments. External validity refers to the generalizability of the results for a larger population. Construct validity is about the suitability of the study design for the theory behind the experiment. Finally, conclusion validity focuses on the relationship between treatment and outcome and on the ability to draw conclusions from this relationship. All validity threats in the experiments are categorized based on this classification.

6.2.1 Internal Validity

- The variation in human performance might distort the results of the experiments, and then the performance differences would not arise from the difference in treatments. In these particular experiments, the partici-

pants' experience is quite comparable among the control groups and the experiment groups (as shown in Fig. 1). Thus, this factor is not seen as a strong threat to validity.

- Another potential threat to validity is that the participants' population in the experiments might not be sufficiently competent. This might influence the results of the experiments. In these particular experiments, all the participants' had knowledge about software development, software architecture and the ESB domain, as well as of software traceability. They all studied the previous lectures and practical assignments of at least the software architecture course.
- Finally, the analysts could have been biased towards the experiment groups. We tried to exclude this threat to validity by not revealing the identity of the participants or in which of the two groups they have participated to the analysts. Hence, it is rather unlikely that this threat occurred.

6.2.2 External Validity

- As discussed in Section 3.2, the experiments were conducted with rather inexperienced participants, the students of a software architecture course. Nevertheless, the results of our previous studies, where we compared the results from a controlled experiment with students and professionals, imply that the participants' experience does not have a significant influence on the external validity of results [10]. Therefore, we conclude that it is likely the limited level of experience of the participants in the two experiments does not distort the study results.
- The instrumentation in the experiments might have been unrealistic or old-fashioned. In this case, the change impact analysis was based on the hyperlinks. In practice, different tools would be used: These tools are primarily used to formulate and maintain the traceability or dependence relationships between the related software elements. In these experiments, for practical reasons and to study the foundational concepts rather than a specific tool, the source code of the software systems and traceability links were readily provided in a web-based format. We assume that the measured effect of the experiment groups during traceability recovery is independent of the way in which a tool would visualize the traceability links, but a threat to validity remains that our results cannot be 1:1 translated to all existing tools and visualizations.

6.2.3 Construct Validity

- The comparison of two differently sized software systems in the experiments involves various aspects, for example, application domain, support features, coding styles, system structures and comments in the source code. We tried to mitigate these risks by selecting the differently sized industrial software systems from the same domain that are highly comparable. In addition, the same questionnaire is used for the selected systems. Therefore, we argue that the size variable plays a major role in the observed differences. This threat, however, cannot totally be ignored.
- Another potential threat to validity is the number of measures used to evaluate the quantity and quality of retrieved elements. In our case we only used standard information retrieval metrics, in particular, recall, precision, and f-measure, to measure the quantity of correctly and incorrectly retrieved elements, and their overall quality, respectively. This does not allow for cross-checking the results with different measures.

6.2.4 Conclusion Validity

- A threat to validity might result from the interpretation of the impact evaluation activities because impact of these activities consists of a list of system elements (e.g., architectural components, source code classes). We mitigated this risk by calculating the standard information retrieval metrics for retrieved elements from all impact evaluation activities. We argue that information retrieval measures allow analysts to objectively evaluate the correctness of impact evaluation activities rather than intuitive or ad-hoc human measures. We conclude that this potential threat is mitigated to large degree.
- Finally, the violation of assumptions made by statistical tests could distort the results of the experiments. In the analysis of the experiments, the Shapiro-Wilk normality test and Wilcoxon Rank-Sum test are used. First, Shapiro-Wilk normality test is used to find out whether equal variances of the level of correctness can be assumed. Second, as a consequence of non-normal distributions, the corresponding non-parametric statistical test, the Wilcoxon Rank-Sum test, is used to test the significance of the found results. Note that the results of the tests were interpreted as statistically significant at $\alpha = 0.05$ (i.e., the level of confidence is 95%). Thus, this factor is not seen as a threat to validity.

6.2.5 Lessons Learned

The analysis of the differently sized systems showed that traceability is more important in larger software systems. This result seemed to be generally applicable for all the benefits of traceability, such as software reuse, maintenance, evolution and quality control. In the absence of traceability information, the developers and architects have to investigate and understand the larger part of the software system. This is even more cumbersome and costly in the large-sized systems. Most importantly, the size factor makes it harder to achieve and maintain the compliance between artefacts produced in the different activities of the development process, such as requirements, architecture design, detailed design, implementation, and testing.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we describe the results of two controlled experiments that were conducted with differently sized ESB systems, in particular, UltraESB Version 2.3.0 and PetalsESB Version 4.2.0, to find out if traceability links are specifically beneficial for larger systems. Three aspects were specifically taken into consideration: the quantity of correctly and incorrectly retrieved elements, and their overall quality. The evaluation of the experiments shows

that using traceability links leads to slight difference in the quantity and quality of retrieved elements for a 133.71% larger software system. This difference in the absence of traceability links, however, significantly increases the quantity of incorrect elements and reduces the overall quality of elements retrieved in the larger software system, while no conclusive evidence concerning the quantity of missing elements was found. It is also interesting to note that the achieved results were statistically significant for lower quantity of missing and incorrect elements, and overall, a higher quality of the retrieved elements for both UltraESB Version 2.3.0 and PetalsESB Version 4.2.0 when analysed individually.

As it is usual for empirical studies, replications in different contexts, with different objects and participants, are good ways to corroborate our findings. Comparing the results of the different automation levels of traceability links is part of our future work agenda. Another direction for future work is to replicate the experiments with our trace link tool that is currently under development.

8. ACKNOWLEDGEMENTS

This work is supported by the Austrian Science Fund (FWF), under project P24345-N23. We also thank to all the participants for taking part in the experiments.

9. REFERENCES

[1] B. Boehm, H. D. Rombach, and M. V. Zelkowitz. *Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[2] T. Cook and D. Campbell. *Quasi-experimentation: design & analysis issues for field settings*. Rand McNally College, 1979.

[3] B. Cornelissen, A. Zaidman, and A. van Deursen. A controlled experiment for program comprehension through trace visualization. *IEEE Trans. Softw. Eng.*, 37(3):341–355, May 2011.

[4] D. Cuddeback, A. Dekhtyar, and J. Hayes. Automated requirements traceability: The study of human analysts. In *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, RE '10, pages 231–240. IEEE Computer Society, 2010.

[5] A. De Lucia, R. Oliveto, and G. Tortora. Assessing ir-based traceability recovery tools through controlled experiments. *Empirical Softw. Engg.*, 14(1):57–92, Feb. 2009.

[6] A. De Lucia, R. Oliveto, F. Zurolo, and M. Di Penta. Improving comprehensibility of source code via traceability information: A controlled experiment. In *Proceedings of the 14th IEEE International Conference on Program Comprehension*, ICPC '06, pages 317–326. IEEE Computer Society, 2006.

[7] A. Dekhtyar, O. Dekhtyar, J. Holden, J. Hayes, D. Cuddeback, and W.-K. Kong. On human analyst performance in assisted requirements tracing: Statistical analysis. In *Proceedings of the 2011 19th IEEE International Requirements Engineering Conference*, RE '11, pages 111–120. IEEE Computer Society, Aug 2011.

[8] O. Gotel and A. Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of the 1994 First IEEE International Requirements Engineering Conference*, pages 94–101. IEEE Computer Society, Apr 1994.

[9] J. Hayes, A. Dekhtyar, S. Sundaram, E. Holbrook, S. Vadlamudi, and A. April. Requirements tracing on target (retro): improving software maintenance through traceability recovery. *Innovations in Systems and Software Engineering*, 3(3):193–202, 2007.

[10] M. A. Javed and U. Zdun. The supportive effect of traceability links in architecture-level software understanding: Two controlled experiments. In *Proceedings of the 11th Working IEEE/IFIP Conference on Software Architecture*, WICSA 2014, pages 215–224. IEEE, 2014.

[11] M. A. Javed and U. Zdun. A systematic literature review of traceability approaches between software architecture and source code. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 16:1–16:10. ACM, 2014.

[12] M. A. Javed and U. Zdun. The supportive effect of traceability links in change impact analysis for evolving architectures – two controlled experiments. In *14th International Conference on Software Reuse*. Springer, January 2015.

[13] A. Jedlitschka, D. Hamann, T. Göhlert, and A. Schröder. Adapting profes for use in an agile process: An industry experience report. In *PROFES*, pages 502–516, 2005.

[14] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *Software Engineering, IEEE Transactions on*, 28(8):721–734, Aug. 2002.

[15] P. Mader and A. Egyed. Assessing the effect of requirements traceability for software maintenance. In *Proceedings of the 2012 28th IEEE International Conference on Software Maintenance*, ICSM '12, pages 171–180, 2012.

[16] H. Mann and D. Whitney. *On a Test of Whether One of Two Random Variables is Stochastically Larger Than the Other*. Institute of Mathematical Statistics, 1947.

[17] T. I. of Electrical and E. Engineers. Ieee standard glossary of software engineering terminology. IEEE Standard, September 1990.

[18] B. Ramesh and M. Jarke. Toward reference models for requirements traceability. *IEEE Trans. Softw. Eng.*, 27(1):58–93, Jan. 2001.

[19] M. Shahin, P. Liang, and Z. Li. Architectural design decision visualization for architecture design: preliminary results of a controlled experiment. In *Proceedings of the 5th European Conference on Software Architecture: Companion Volume*, ECSA '11, pages 2:1–2:8. ACM, 2011.

[20] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):pp. 591–611, 1965.

[21] C. Wohlin. *Experimentation in Software Engineering: An Introduction: An Introduction*. The Kluwer International Series in Software Engineering. Kluwer Academic, 2000.