IR-based Traceability Recovery Processes: an Empirical Comparison of “One-Shot” and Incremental Processes

Andrea De Lucia, Rocco Oliveto, Genoveffa Tortora
Department of Mathematics and Informatics, University of Salerno
84084 Fisciano (SA) - Italy
Emails: {adelucia, roliveto, tortora}@unisa.it

Abstract—We present the results of a controlled experiment aiming at analysing the role played by the approach adopted during an IR-based traceability recovery process. In particular, we compare the tracing performances achieved by subjects using the “one-shot” process, where the full ranked list of candidate links is proposed, and the incremental process, where a similarity threshold is used to cut the ranked list and the links are classified step-by-step. The analysis of the achieved results shows that, in general, the incremental process improves the tracing accuracy and reduces the effort to analyse the proposed links.

I. INTRODUCTION

The need to provide software engineer with methods and tools supporting traceability recovery has been widely recognised in the last years. Several researchers have applied Information Retrieval (IR) [1], [2] techniques for recovering traceability links between artefacts of different types [3], [4], [5], [6], [7], [8], [9], [10], [11].

An IR-based traceability recovery tool compares a set of source artefacts against another set of target artefacts and ranks the similarity of all possible pairs of artefacts. The conjecture is that artefacts having a high textual similarity probably share several concepts, so they are likely good candidates to be traced from one another. Generally, the list of retrieved links contains a higher density of correct traceability links in the upper part of the list and a much lower density of such links in the bottom part of the list. As a consequence, while IR-based approaches help in the identification of traceability links in the upper part of the ranked list, the ranking in the lower part of the returned list of candidate links give almost no help, due to the presence of too many links that are not correct (false positives). This means that the effort required to discard false positives becomes much higher than the effort to validate correct links. For this reason, most approaches use some method to cut (or filter) the ranked list (e.g., a threshold on the similarity value), thus presenting the software engineer only the subset of top links in the ranked list [3], [6], [8], [9], [10]. Of course, the lower the similarity threshold used, the higher the number of correct links as well as the number of false positives retrieved. For this reason, we propose to incrementally decrease the similarity threshold to give the software engineer the control on the number of validated correct links and the number of discarded false positives [6]. In this way, starting with a high threshold, the links suggested by the tool can be analysed and classified step-by-step and the process can be stopped when the effort to discard false positives is becoming much higher than the effort to identify new correct links.

Using the proposed traceability recovery process we observed that the threshold used to stop the process plays an important role [12]. In particular, when the software engineer stops the traceability recovery process using a low threshold he/she increases the number of correct links (better recall) with respect to software engineers stopping the process with a higher threshold [12]. This situation suggests that better recall could be achieved by providing the software engineer with the full ranked list of possible links ordered by decreasing similarity values (“one-shot” contrasted to the incremental process). However, the ranked list contains a high density of correct links in the upper part and a low density of such links in the lower part. According to these considerations, we expect that the full ranked list method might result in a better recall but in a worse precision (more tracing errors) with respect to the incremental process. Indeed, using an incremental process the software engineer tends to concentrate only on the upper part of the list and he/she is able to work with lists of candidate links of smaller size, thus making the recovery process less error prone.

In this paper we present the results of a controlled experiment carried out to analyse the effect of the traceability recovery process on the retrieval performances of the software engineer. The experimentation involved 20 master students of the University of Salerno, Italy. In the context of the experiment subjects had to use the traceability recovery tool of ADAMS [6] to perform two different traceability recovery tasks. For each task they used a different traceability recovery process, i.e., the “one-shot” or the incremental process. In this way, we can try to respond to the following questions:

1) does the “one-shot” process reduce the time spent by the software engineer to complete a traceability recovery task?
2) does the “one-shot” process increase the number of correct links classified by the software engineer (better recall)?
3) does the incremental process reduce the tracing errors made by the software engineer (i.e., the number of false
 Regarding the time and the accuracy, the results achieved by subjects were not influenced by the traceability recovery process. Instead, the process significantly affects the number of correct links identified and the tracing errors.

The rest of the paper is organised as follows. Section II discusses related work, while Section III presents the traceability recovery functionalities provided by ADAMS. Sections IV and V present the planning and the experiment results, respectively. Finally, Section VI gives concluding remarks and lessons learned.

II. RELATED WORK

Antoniol et al. [3] were the first to use IR methods for recovering traceability links between software artefacts. In particular, they apply and compare the probabilistic and vector space models [1] to recover traceability links between code and documentation in two case studies. The achieved results revealed that the two methods have similar performances when a preliminary morphological analysis (stemming) of the software artefacts is performed. Other authors have used enhancing strategies to improve the tracing performance of these basic models. Examples of enhancing strategies for the probabilistic model are hierarchical modelling [5], logical clustering of artefacts [5], [13], semi-automated pruning of the probabilistic network [5], and query term coverage and phrasing [14]. Instead, thesaurus [15], [11], key-phrases [15] and pivot normalisation weighting scores [11] have been proposed to improve the accuracy of the vector space model.

The retrieval accuracy of an IR-based traceability recovery tool can be improved by learning from an input training set of correct links [4], [7] or from user feedback provided during the classification of the candidate links [16], [8]. However, even though the retrieval performance generally improves with the use of feedback, IR-based approaches are still far from solving the problem of recovering all correct links with a low classification effort [16]. In a recent paper, Dekhtyar et al. [17] investigate the impact of different feedback strategies that human analyst may employ while IR methods generate candidate traceability links.

Marcus and Maletic [10] propose the use of Latent Semantic Indexing (LSI) [2], an extension of the VSM, for traceability recovery. They perform the same case studies as in [3] and compare the accuracy of LSI with respect to the vector space and probabilistic models. The achieved results show how the latter models require morphological analysis of text contained in source code and documentation to achieve the same accuracy as LSI. LSI was also used to reconstruct traceability links between high-level and low-level requirements [8], as well as among software artefacts of different types [6], [9].

Based on the traceability recovery methods proposed in the literature, several IR-based tools have also been proposed to support the software engineer during the traceability recovery process [6], [8], [9], [18], [19]. In some cases, the usefulness of such tools has also been assessed through user studies. In [3] the authors presented the results of a preliminary study where they compare the IR-based approaches against a "grep" brute force traceability link recovery demonstrating the benefits of a more sophisticated technology, like an Information Retrieval method.

De Lucia et al. [6] presented the results of a case study carried out to validate the traceability recovery tool of ADAMS. In the experimentation the tool was used by about 150 users in 17 software development projects. The students used ADAMS as artefact management system and exploited the traceability features of ADAMS for maintaining up-to-date the traceability information. In particular, they had the possibility to trace links manually or with the tool support. At the end of the experimentation the authors analysed the links traced by students and observed that almost all the traced links were traced with the tool support. Moreover, at the end of the experimentation, students evaluated ADAMS through a questionnaire. The analysis of the answers revealed that students found the tool useful during the traceability recovery process.

The main drawback of case studies is that comparing the usefulness of different methods and/or tools is very difficult [20]. In particular, it is difficult to analyse how the tracing accuracy of the software engineer is affected by the use of a traceability recovery tool. For this reason we conducted a controlled experiment aiming to achieve such a statistical evidence [12]. The experimentation involved first year master students, who had to perform two traceability recovery tasks (one with the support of ADAMS Re-Trace and another manually) on a software repository of a completed project. In this way we were able to analyse whether an IR-based traceability recovery tool improves the performance of the software engineer with respect to a manual approach. The results demonstrated that the use of a traceability recovery tool significantly improves the tracing accuracy of the software engineer, measured as the harmonic mean of his/her precision and recall [1]. In particular, it significantly reduces the percentage of false positives traced (better precision), although it does not significantly help to recover more correct links (better recall). Moreover, the tool significantly reduces the time spent by the software engineer to trace links.

III. TRACEABILITY RECOVERY IN ADAMS

ADAMS (ADvanced Artefact Management System) is a fine-grained artefact management system [21]. Traceability links in ADAMS are useful for impact analysis and change management during software evolution. ADAMS allows software engineers to manually create traceability links between artefacts. However, when the number of project artefacts is high, traceability link management tends to be a difficult task. For this reason, we have integrated in ADAMS a traceability recovery tool based on LSI, called ADAMS Re-Trace [6].

The traceability recovery functionality is activated across a three steps wizard, where the software engineer selects (i)
the source and target categories (e.g., use cases, code classes) and can filter on the names of artefacts, (ii) the artefacts (belonging to the selected categories) he/she is interested in, and (iii) the approach used to show the list of candidate links (see Figure 1). In particular, the tool supports both the “one-shot” and the incremental processes. The former is supported through the “Full ranked list” approach, where all the possible pairs of source and target artefacts are shown in a decreased order of similarity. The latter is supported through the “Threshold based” approach, where a threshold is used to cut the ranked list and to consider as candidate links only the links between artefacts with similarity above such a threshold. It is important to note that in the “Full ranked list” approach the similarity level of each pair of artefacts is not shown in order to avoid an implicit similarity filtering.

In order to better support the incremental process, the traceability recovery tool of ADAMS maintains for each pair of artefact types the lowest and the last thresholds used by the software engineer in previous traceability sessions. In particular, in the third step of the traceability recovery wizard the software engineer has the possibility to select one of the following thresholds to cut the ranked list:

- the default threshold, i.e. 95%;
- the lowest threshold used by the software engineer in previous sessions;
- the last threshold used in the previous session.

Once the source and target artefacts have been selected and the approach used to show the list of candidate links has been chosen, the software engineer can start the traceability recovery. The tool compares the links retrieved by using the proposed approach (i.e., full ranked list or threshold based) with the links traced by the software engineer. In this way the tool is able to show the software engineer only retrieved links that have not been traced yet.

Figure 2 shows the list of candidate links obtained using the “Threshold based” approach. The software engineer can analyse such a list, trace the correct links and discard the false positives. Moreover, the software engineer has also the
possibility to decrease the threshold and analyse new candidate links if he/she has the perception that the cost of discarding false positives is still acceptable compared with the benefits of identifying new correct links. For this reason, the tool also maintains information concerning the previous traceability recovery iterations (with a higher similarity threshold) in terms of number of suggested links with respect to the number of links classified as correct links or false positives.

IV. EXPERIMENT PLANNING

This section describes in detail the planning of the proposed experimentation following the guidelines by Wohlin et al. [20]. The goal of the experiment was to analyse the tracing performances of the software engineer, in terms of time spent and links traced, with the purpose of evaluating the impact of the traceability recovery process adopted. The quality focus was ensuring better tracing performances, while the perspective was both (i) of a researcher, that wants to evaluate how the traceability recovery process affects the recovery performances of the software engineer; and (ii) of a project manager, that wants to evaluate the possibility of adopting the process within his/her own organisation.

A. Context selection

The study was executed at the University of Salerno, Italy. The subjects of the experiment were 20 master students in Computer Science at the University of Salerno (Italy), attending the course of Advanced Software Engineering. All the subjects were from the same class with a comparable level of background, but different levels of ability. They had knowledge of both software development and software documentation, as well as of software artefact traceability.

In the context of the experiment, subjects had to perform two traceability recovery tasks over an artefact repository of a completed software project, called EasyClinic. The project aimed at developing a software system implementing all the operations required to manage a doctor’s office. The artefact repository was composed of 30 use cases, 20 interaction diagrams, 63 test cases, and 37 code classes. The traceability matrix provided by the original developers was used to define the number of links correctly and erroneously traced by each subject.

The experiment was performed in a controlled laboratory setting and the two traceability recovery tasks that students had to perform were:

- **T1** recovering traceability links between 30 use cases and 37 code classes (the number of all correct links is 93);
- **T2** recovering traceability links between 20 interaction diagrams and 63 test cases (the number of all correct links is 83).

The experiment was organised in two laboratory sessions where subjects used the traceability recovery tool of ADAMS to complete the traceability recovery task. In each session, subjects performed the traceability recovery task adopting the “one-shot” or the incremental process.

B. Hypothesis formulation

The main objective of our study was to analyse how the traceability recovery process affects the time spent and the tracing performances of subjects, in terms of correct links recovered (and traced), tracing errors, and accuracy of the link classification made by subjects, by tracing correct links and discarding false positives. Thus, we formulated the following null hypotheses:

- \( H_0 \) the “one-shot” process does not significantly reduce the time spent by the software engineer to trace the links;
- \( H_{0,cl} \) the “one-shot” process does not significantly increase the amount of correct links traced by the software engineer;
- \( H_{0,te} \) the incremental process does not significantly reduce the amount of links erroneously traced by the software engineer (false positives);
The approach adopted to perform the traceability recovery task: “one-shot” or the incremental process. The ability level of each involved subject: Low and High Ability. The time spent by a subject to complete a traceability task. The percentage of links correctly classified (as correct links or false positives) over the total number of classified links.

H₀�, the incremental process does not significantly improve the overall accuracy of the link classification made by the software engineer. Consequently, the alternative hypotheses are:

H₁�, the “one-shot” process significantly reduces the time spent by the software engineer to trace the links;

H₁₀�, the “one-shot” process significantly increases the amount of correct links traced by the software engineer;

H₂₀�, the incremental process significantly reduces the amount of links erroneously traced by the software engineer (false positives);

H₂₁�, the incremental process significantly improves the overall accuracy of the link classification made by the software engineer.

Our study was also devoted to investigating how subjects’ ability interacts with the traceability recovery process and affects the tracing performances. This required to formulate another null hypothesis:

H₀ₐ, subjects’ ability does not significantly interact with the traceability recovery process.

The related alternative hypothesis can be easily derived.

C. Variable selection

In the context of the experimentation, we identified the following independent variables (factors):

- **Process**: the main factor of our study, i.e., performing a traceability recovery task adopting the “one-shot” or the incremental process.
- **Task**: the two traceability recovery tasks to perform, indicated with T₁ and T₂ (see Section IV-A).
- **Lab**: the two laboratory sessions, indicated with Lab1 and Lab2.

Other than **Process**, the experimental hypotheses were defined in terms of another factor, namely **Ability**. A quantitative assessment of the ability level of each involved subject was obtained by considering the average grades obtained at the previous exams. Subjects with average grades below a fixed threshold, i.e., 24/30, were classified as *Low Ability* (Low), while the remaining ones as *High Ability* (High).

The main outcome observed in the study were the time spent and the tracing performances of subjects. In order to test the first null hypothesis, i.e. H₀₀, we considered as dependent variable the time spent by the subject to complete the traceability task. Concerning the null hypotheses H₀₁₀, H₀₁₁, and H₀₂₀, the tracing performances of the subjects were assessed analysing the classification of the proposed links. In the following, *correct* is the set of correct links (derived from the original traceability matrix), while *traced* and *discarded* are the sets of links classified by a subject as correct links and false positives, respectively. Analysing the link classification of the software engineer, we can define four sets:

- *correctlyTraced* = *traced* ∩ *correct*
- *incorrectlyDiscarded* = *discarded* ∩ *correct*
- *uncorrectlyTraced* = *traced* ∩ *incorrect*
- *incorrectlyDiscarded* = *discarded* ∩ *incorrect*

where A denotes the complementary set of A. Thus, it is possible to assess the performances of each subject in each task using three metrics widely used to evaluate classification results [1], [22]:

\[
\text{recall} = \frac{\text{|correctlyTraced|}}{\text{|correct|}} \quad \text{precision} = \frac{\text{|correctlyTraced|}}{\text{|traced|}} \quad \text{accuracy} = \frac{\text{|correctlyTraced| + correctlyDiscarded|}}{\text{|traced| + discarded|}}
\]

It is easy to note that recall measures the percentage of correct links traced by a software engineer, while precision measures the percentage of traced links that are actually correct. Moreover, accuracy measures the percentage of links correctly classified (as correct links or false positives) over the total number of classified links. For this reason, recall, precision, and accuracy are the dependent variables used to test the null hypotheses H₀₀₁, H₀₁₁, and H₀₂₀, respectively. Table I summarises the independent and the dependent variables, as well as the collected data.

D. Experiment design

The assignment given to each group of subjects in each experimental session (Lab 1 and Lab 2) followed the design in Table II. Such a design ensures that each subject works on different tasks in the two laboratory sessions, using each time a different method. Also, the design permits to consider different combinations of Task and Process in different order across laboratory sessions. Thus, the chosen design permitted the use of statistical tests, i.e., Two-Way Analysis of Variance (ANOVA) [23], to analyse the effects of multiple factors.

Subjects performed the tasks individually. In particular, we organised them in two sets taking into account the ability level and then randomly distributed subjects among the laboratory groups making sure that High and Low ability subjects were...
equally distributed across groups\(^2\). Before the experiments, we presented the subjects with the traceability recovery tool showing how to trace links using both the “one-shot” and the incremental processes. Moreover, we let subjects get confidence with the tool by performing some simple traceability recovery tasks on a software artefact repository not related to EasyClinic to avoid biasing the experiment. Finally, right before the experiments, we showed to the students a presentation with detailed instructions on the tasks to be performed.

It is important to note that students were not familiar with the EasyClinic project beforehand. Such a situation could influence our controlled experiment because it is very difficult to reach an agreement on what should be a link and what should not be a link when there is no domain knowledge. This is the reason why a month before the controlled experiment execution we gave to the students both the system documentation and the source code. Moreover, meetings made by students together with system experts were scheduled twice a week aiming at enriching the students’ domain knowledge.

During the controlled experiment each student was provided with the following material:

- handouts of the introductory presentation and the user guide of the traceability recovery tool;
- an electronic version of the software documentation and the source code of EasyClinic;
- the ADAMS system;
- a survey questionnaire (shown in Table III) to be filled-in after each laboratory session.

The survey questionnaire was composed of questions expecting closed answers according to the Likert scale [24] - from 1 (strongly agree) to 5 (strongly disagree) - to assess if the system’s domain and task were clear, if subjects had enough time, and other related questions (Q1 to Q5). In addition, at the end of the second experiment lab, subjects filled-in another questionnaire, composed of three questions (Q6 to Q8), aiming at investigating the usefulness and the usability of the tool.

For each Lab, subjects had two hours available to perform the required Task. After the task was completed, they returned us the filled-in survey questionnaire, while the links traced and discarded by subjects were automatically stored in the ADAMS database.

V. ANALYSIS AND INTERPRETATION OF RESULTS

After the controlled experiment execution, we analysed the links traced by each subject in each task and used the original traceability matrix of EasyClinic to compute the recall, precision, and accuracy of each subject. Table IV reports descriptive statistics of the dependent variables grouped by Process and Task. Moreover, the table also reports descriptive statistics of the number of links analysed by subjects.

The next sections report and discuss the results achieved in the controlled experiment, analysing the effect on the dependent variables of the main factor (i.e., Process) and of other factors. Finally, results from the analysis of survey questionnaires and the discussion of threats to validity are also reported.

A. Influence of Process

In order to test the first four null hypotheses, i.e. \(H_{0,t}, H_{0,cl}, H_{0,i}, \) and \(H_{0,a}\), we analysed the effect of Process on the three dependent variables. Since experiments were organised as longitudinal studies where each subject performed two different traceability recovery tasks, with the two possible treatments (adopting the “one-shot” and the incremental process) it was possible to use a paired Wilcoxon one-tailed test [25] to analyse the differences exhibited by each subject for the two treatments. The results were intended as statistically significant at \(\alpha = 0.05\) and are reported in Table V. The table also reports descriptive statistics of differences achieved by subjects, the effect size [26] and the percentage of positive differences (i.e., \% of Pos. Eff.).

Analysing the results we observed that \(H_{0,a}\) cannot be rejected. Indeed, analysing the descriptive statistics of differences achieved by subjects we observe that, in general, subjects spent the same time to complete the traceability recovery adopting or not the “one-shot” process (see Table V).

Concerning \(H_{0,i,a}\) and \(H_{0,i}\), the Wilcoxon tests revealed that our initial findings are correct. In particular, we can reject both the null hypotheses \(H_{0,i,a}\) and \(H_{0,i}\). Moreover, analysing the descriptive statistics of differences achieved by subjects we also observed that the effect of the incremental process on precision is higher than the effect of the “one-shot” process on recall. In particular, the analysis of the effect size highlighted only a small effect of Process on recall (effect size < 0.5) and a medium effect of the same factor on precision (effect size ≥ 0.5 and < 0.8) [26]. This suggested us a deeper analysis of the achieved results. Indeed, we analysed the lowest threshold used by subjects with the incremental process. The analysis revealed two outliers. In particular, on average the subjects stopped the traceability recovery process using a low threshold (i.e., 43.75%), while the two outliers stop the traceability recovery process with a higher threshold (i.e., 65%
for both the outliers). Thus, we decided to cut-off from the data set the results achieved by these two subjects and reanalysed the data. Concerning the second dependent variable, i.e., recall, we observed that, in this case, we cannot reject the null hypothesis \( p\text{-value} = 0.064 \). This means that when the incremental process is not stopped with a too high threshold, the number of correct links retrieved is comparable with the number of correct links retrieved adopting the “one-shot” process. On the other hand, also in this case we can reject \( H_{0_c} \) \( (p\text{-value} = 0.023) \), i.e., the incremental process still results in a better precision.

Finally, the analysis of the results also revealed that we cannot reject \( H_{0_c} \). Nevertheless, analysing the descriptive statistics of differences achieved by subjects we observe that subjects obtained a better accuracy (not statistically evident) using the incremental process (see Table V). Moreover, we also observed that subjects performing a traceability task adopting a “one-shot” process analysed a much larger number of links than subjects performing the same task but adopting the incremental process \( (p\text{-value} = 0.003) \).

B. Influence of other factors

In this section we focus the attention on the effect of other independent variables, i.e. Lab, Task, and Ability on the dependent variables. Concerning time, ANOVA revealed no significant effect of Lab \( (p\text{-value} = 0.635) \), Task \( (p\text{-value} = 0.323) \), and Ability \( (p\text{-value} = 0.292) \), as well as no significant interaction between factors.

Regarding recall, ANOVA indicated no significant effect of Lab \( (p\text{-value} = 0.430) \) as well as of Task \( (p\text{-value} = 0.393) \). Moreover, ANOVA did not reveal significant interaction between factors. Considering the effect of Ability, ANOVA revealed a significant effect of such a factor on recall \( (p\text{-value} = 5.137e-04) \) indicating that High ability subjects were able to trace more correct links than subjects with Low ability.

ANOVA also revealed no significant effect of Lab \( (p\text{-value} = 0.521) \) on precision, as well as no significant interaction between Process and Lab \( (p\text{-value} = 0.923) \). Moreover, the analysis revealed a significant effect of Task on the tracing errors made by the subjects \( (p\text{-value} = 0.012) \), while no significant interaction between Process and Task was revealed \( (p\text{-value} = 0.747) \). In particular, the Wilcoxon test revealed that performing task T2 subjects achieved better precision level than when performing task T1 \( (p\text{-value} = 0.015) \). No significant differences emerged between the two processes used to perform the traceability recovery task. Indeed, the different precision levels achieved by subjects performing the two tasks is due to the different retrieval performances of the IR method. In particular, the performances of the IR engine are better for task T2 (tracing interaction diagrams onto test cases) than for task T1 (tracing use cases onto code classes), as discussed in [6]. ANOVA also revealed no significant effect of Ability on precision \( (p\text{-value} = 0.097) \), as well as no significant effect of the interaction between Process and Ability \( (p\text{-value} = 0.549) \).

Finally, ANOVA revealed no significant effect of Lab \( (p\text{-value} = 0.593) \) on accuracy, as well as no significant interaction between Process and Lab \( (p\text{-value} = 0.151) \). No significant effect on accuracy of both Task \( (p\text{-value} = 0.650) \) and the interaction between Process and Lab \( (p\text{-value} = 0.905) \) was also revealed. As well as for recall, ANOVA revealed a significant effect of Ability on accuracy \( (p\text{-value} = 5.121e-05) \) indicating that High ability subjects were able to better classify proposed links than subjects with Low ability.

C. Survey questionnaire results

The feedback provided by subjects after each Lab were analysed aiming at better understanding the experimental results. Figure 3 shows the boxplots of the results grouped by Task. Statistical significance of differences has been tested using a paired Wilcoxon one-tailed test (to analyse the effect of Task) and a Mann-Whitney one-tailed test [25] (to analyse the effect of Ability and Process).

By looking at the more general questions, we noticed that, overall, subjects had enough time to perform the task \( (Q1) \). No significant differences emerged between tasks \( (p\text{-value} = 0.265) \) and different levels of Ability \( (p\text{-value} = 0.212) \), as well as between processes \( (p\text{-value} = 0.091) \). The objectives \( (Q3) \) and the laboratory tasks \( (Q4) \) were clear, and also in this case statistical tests revealed no significant effect of Task \( (p\text{-values equal to 0.960 and 0.242, respectively}) \), Ability \( (p\text{-values equal to 0.274 and 0.734, respectively}) \) and Process \( (p\text{-values equal to 0.546 for both the two questions}) \). Regarding the system domain knowledge \( (Q2) \), the results show that during the controlled experiment subjects had a perception of an acceptable domain knowledge and no significant
difference between different levels of **Ability** (p-value = 0.228) and between tasks (p-value = 0.395) emerged. No major difficulties were experienced by subjects during the laboratory sessions (Q3). Moreover, the analysis revealed no significant effect of **Task** (p-value = 0.389) and **Ability** (p-value = 0.721), but a significant effect of **Process** (p-value = 0.016). In particular, using the “one-shot” process subjects experienced more difficulties than using the incremental process.

Questions related to the use of the tool revealed that in general it was clear (Q6). Figure 4 shows the boxplots of the results grouped by **Ability**. No significant difference emerged between subjects (p-value = 0.518). Regarding the usefulness of the proposed links (Q7) subjects generally found the suggestions of the tool useful and no significant difference emerged between subjects (p-value = 0.367). Finally, subjects generally preferred to apply the incremental process in both tasks (Q8). Also in this case no significant difference emerged between subjects (p-value = 0.433).

**D. Threats to validity**

In this section we discuss the threats to validity that can affect our results. Concerning the conclusion validity, attention was paid to not violate assumptions made by statistical tests. Whenever conditions necessary to use parametric statistics did not hold (e.g., analysis of each experiment data), we used non-parametric tests, in particular Wilcoxon test for paired analyses. We dealt with random heterogeneity of subjects by introducing the **Ability** factor, and analysing their interaction with **Process**. Finally, survey questionnaires, mainly intended to get qualitative insights, were designed using standard ways and scales [24]. This allowed us to use statistical test, i.e., Mann-Whitney one-tailed test, for analysing differences in the feedback too.

The construct validity threats concern the relationship between theory and observation. In our study, recall, precision, and accuracy well reflect the traceability link identification performances of the software engineer. Interactions between different treatments were mitigated by the chosen experimental design. Regarding the levels of the **Ability**, we used only two levels, that is Low and High, discriminating students taking into account the average grades they obtained at the previous exams. Clearly, more levels than Low and High could have been used. Nevertheless, analyses performed with more levels did not yield any different or contrasting results. Another threat is related to the number of subjects with low and high abilities, as this number is not even. We mitigated such a threat randomly distributed subjects among the laboratory groups making sure that High and Low ability subjects were equally distributed across groups. In order to avoid social threats due to evaluation apprehension, students were not evaluated on their performances in the Lab. Moreover, subjects were not aware of the experimental hypotheses.

The internal validity concerns external factors that may affect an independent variable. An important threat is related to the learning effect experienced by subjects between labs. This is mitigated thanks to the experiment design: subjects worked, over the two labs, on different traceability tasks and using two different levels of the main factor (“one-shot” and incremental processes). Nevertheless, there is still the risk that,
during labs, subjects might have learned how to improve their tracing performances. We tried to limit this effect by means of a preliminary training phase. Moreover, the Lab factor has been accounted as a factor in the analysis of results and the ANOVA analysis showed no significance of Lab. Finally, there was no abandonment and everything was clear.

The external validity concerns the generalisation of the findings. This kind of threat is always present when experimenting with students. The selected subjects represent a population of students specifically trained on software development technologies and software engineering methods. Also, all students are master students who either had some professional experiences or worked on industrial projects during their Bachelor thesis. This makes these students comparable to industry junior developers. Moreover, before the controlled experiment execution meetings were organised with the aim of giving to the students an acceptable system domain knowledge. The artefact repository used in the controlled experiment is not comparable to industrial projects, but it is just used to evaluate IR methods for traceability recovery [6]. Moreover, the traceability tasks assigned to subjects involved different types of software artefacts. Nevertheless, the working pressure and the overall environment in industry is different, thus replicating the study in industry is highly desirable.

VI. CONCLUSION AND LESSONS LEARNED

In this paper we presented the results of a controlled experiment carried out to evaluate how the process affects the tracing performances of the software engineer when using an IR-based traceability recovery tool. In particular, we compared the tracing performances achieved by subjects performing two traceability recovery tasks using the “one-shot” and the incremental processes. The achieved results provided us with a number of lessons learned:

- the time spent by the software engineer to complete the traceability recovery task is not affected by the traceability recovery process. This means that the extra time for the threshold tuning required for the incremental process does not affect the total time required to complete a traceability task;
- in general, the incremental process improves the traceability link identification performances of the software engineer. The analysis of the achieved results initially revealed that the “one-shot” process helps to recover more correct links with respect to the incremental process. However, a deeper analysis of the achieved results revealed two subjects with Low ability that stopped the incremental process too soon (i.e., with a too high threshold). By removing these outliers we observed that the number of correct links retrieved with the two different processes is comparable. On the other hand, the incremental process significantly reduces tracing errors. As well as the recall, the accuracy of the link classification achieved by the subjects using the two processes is comparable. Finally, the analysis of the questionnaire showed that subjects preferred to apply the incremental process and they experienced more difficulties performing the task adopting the “one-shot” process;
- the incremental process reduces the effort required to complete a traceability recovery task. The number of
links analysed using the “one-shot” process is significantly larger than the number of links analysed adopting the incremental process. Such a result combined with the above results, suggests that the incremental process reduces the effort required to identify traceability links using an IR-based traceability recovery tool. In particular, analysing the lower part of the ranked list only results in a small improvement of the retrieval performances and does not help to identify all correct links. Indeed, while in the upper part of the ranked list the density of correct links is quite good, in the bottom part of the list such a density decreases in such a way that the prioritisation made by the IR method does not help anymore. Thus, IR-based traceability recovery tools might be used to retrieve as many correct links as possible (keeping low the number of false positives to discard) but they have to be necessarily combined with manual tracing activities to build a complete traceability matrix.

As is always the case with empirical studies, replications are the only way to corroborate our findings. Replicating this study with students/professionals having a different background would be extremely important to better understand how an IR-based traceability recovery tool and process influence the traceability link identification performances.

ACKNOWLEDGMENTS

We would like to thank the students who were involved in the experiment as subjects. The work described in this paper is supported by the project METAMORPHOS (METHODS AND TOOLS FOR MIGRATING SOFTWARE SYSTEMS TOWARDS WEB SERVICE ORIENTED ARCHITECTURES: EXPERIMENTAL EVALUATION, USABILITY, AND TECHNOLOGY TRANSFER), funded by MIUR (MINISTERO DELL’UNIVERSITÀ E DELLA RICERCA) under grant PRIN-2006-2006098097.

REFERENCES