CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Identifying relevant change sets to facilitate change impact analysis

Design Science Research

Bachelor of Science Thesis in Software Engineering and Management

Kristiyan Dimitrov
Marcus Nilsson

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Identifying relevant change sets to facilitate change impact analysis
Design science research

Kristiyan Dimitrov
Marcus Nilsson

Supervisor: Jan-Philipp Steghöfer & Salome Maro
Examiner: Francisco de Oliveira Neto

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Identifying relevant change sets to facilitate change impact analysis

Design Science Research

Kristiyan Dimitrov
Gothenburg University
Gothenburg, Sweden
gusdimkr@student.gu.se

Marcus Nilsson
Gothenburg University
Gothenburg, Sweden
gusnilmaed@student.gu.se

***Abstract*—conducting a change impact analysis may prove to be a difficult and costly endeavor. Estimating the potential effects a change would have on the system is essential in nowadays software development where the project budget and deadlines are paying an important role. An inaccurate estimation could lead to a potential failure in the project or delay in the release of the software. But the process of change impact analysis itself also costs resources to conduct. The study observes whether the process of change impact analysis can be improved in terms of time spent and errors made by using a traceability graphical representation and hiding irrelevant trace links on a system level, based on a cross-cutting approach proposed by [1].**

Keywords—Traceability, change impact analysis, software engineering.

I. Introduction

We live in a dynamically changing world, the area of software engineering in particular is in a constant state of change. Changes may occur in the user requirements, business goals or be caused by technological advancement. As such change management has become an essential part of producing high-quality software. Identifying the artifacts affected by a change within a system and estimating the cost is a key process to cost-efficient development [1].

To understand the problem at hand, an understanding of the impact of change needs to be made. Change Impact Analysis (CIA) is a process of identification and estimation. Identifying the consequences a change would have on the system and estimating the resource cost that would be incurred by conducting that change [2, 3]. For an accurate estimation to be made, the artifacts affected by a change must be correctly identified. A change could potentially affect large disjoints of the system, hence the resource cost, such as time or money, for conducting a change would raise significantly. Such a cost increase, which may be above the project budget, would result in a negative cost-to-budget ratio or unmet project deadlines [3]. As a result, a significant amount of time has to be spent on CIA. There are various ways of identifying affected artifacts by a change [4]. One such way of identifying the affected artifacts by a change is traceability. The process of identification, in the case of this paper, shall be treated as the act of using traceability to identify the affected artifacts [5]. Traceability is used for tracking the connections between the various artifacts within the system. It allows for a trace link between a source and a target artifact to be established and stored [15]. Once stored, the links can be graphically represented, hence giving a visual description of the links in the system to the analyst. Combined with CIA, traceability has become a vital part of the development process in software companies and the subject of other studies [7].

A. Motivation

Complex systems are composed of many interconnected artifacts. Traceability links of such systems result in complex trace-link trees due to the number of traces that cover these connections. As the complexity of such systems increases, conducting a change becomes more difficult. Despite the presence of several studies that present frameworks and approaches for conducting a CIA [4, 5], the process remains difficult due to the uncertainty involved in identifying the affected artifacts, unless traceability has been a part of the design process [6]. Through a case study it has been established that traceability links reduce the difficulty of identifying details which influence the change for programmers, but fail to do so for the higher level decision makers to understand the impact and make a wise decision [7]. The same conclusion has been reached about traceability on requirements, where measurements for impact analysis are made on the links between artifacts [8]. Other studies observed how much time is spend per developer on CIA. The end result varies significantly and the argumentation as of why from different senior developers, is that it depends on the complexity of the involved component while another says that it also depends on the structure of the documentation [2]. Due to this gap, we see the need to investigate how traceability visualizations can be improved to better facilitate CIA.

B. Purpose statement

The purpose of this design science research is to improve change impact analysis by hiding less relevant traceability links. For this process a software artifact shall be constructed and quantitative analysis methods will be used. To construct the artifact and fulfill the purpose the following research questions have been determined:

RQ1: How can traceability links be enriched to better support change set identification?

- RQ1.1: Would hiding less relevant trace links reduce the time spent per developer on CIA?
- RQ1.2: Would hiding less relevant trace links reduce the number of errors made during the process of identification of affected artifacts?

II. Related Work

The body of literature for traceability and CIA is considerably big. Some studies focused on the fundamentals for traceability and its overall body [15], while others focused on designing traceability as for it to be incorporated within the system [6]. Likewise for CIA in software systems there can be found studies that focus on the overall body [18], while others focused on presenting different approaches as to conduct CIA [4, 5].

A. State of the Art

Nowadays trace links can be created manually, semi automatically or fully automatically [17]. Studies have defined different methods for creating traces between artifacts such as trace links between documentation and source [9], model-based traceability [14] and requirements traceability [7, 15]. Today, in the case of Capra, all trace-links are created without displaying the relevance of the change sets. A change set is a set of artifacts that are connected through some form of connection, i.e. dependency, inheritance, shared variable, etc. A relevant change set is the set of artifacts that will have an impact when making a change to a part of a software system.

B. Potential for Improvement

Different solutions have been observed from various points of view. One such point of view is observing a single component from traceability and designing a new artifact out of its base. By observing traceability strategies and their usability, a model-driven approach towards traceability was established [14]. Another study focused on a goal-centric approach towards requirements traceability, aims at long-term maintenance [16]. Goal-Centric Traceability provides developers with a means of handling functional changes within non-functional requirements. This approach retrieves traceability links from the non-functional requirements but is highly dependent on the human factor [17]. Without a human to filter out non-relevant traces when conducting a change, this approach suffers from a great deal of imprecision.

Both of these approaches are heavily focused on requirements traceability and are subject to imprecision without the human factor to filter out the relevant change sets. While these approaches retrieve the traces from documentation, the relevance of the trace in regards to the change observed in the CIA is unreliable, thus increasing the time spent per developer on CIA.

III. Background

In this section background information about traceability and the tool that shall be used to test the solution shall be given.

A. Traceability

Traceability is the potential for traces to be established and used. A trace is a triplet of elements: source artifact, target artifact and a trace link [15]. For traceability to be effectively used, every trace needs to be created, represented and stored. For the creation, representation and storing of the traces, the traceability tool Capra shall be used. Throughout the paper the term ‘change set’ shall be used to represent all affected artifacts by a change.

B. Capra

Capra is a configurable and extendable traceability management tool based on the Eclipse Modeling Framework (EMF). It is an Eclipse plug-in and provides the ability to create, visualize and maintain trace links between arbitrary software artifacts like requirement documents, UML models, source code, etc. with a simple drag-and-drop function [10]. Various organizations use diverse traceability methods, Capra copes with such conditions by allowing the option of creating Artifact Handlers which support the type of artifacts which the user requires to be linked. Capra on its own provides handlers for Java methods, classes, properties, etc. and C functions, classes, properties etc., as well as Hudson builds and EMF based models like the Papyrus UML models.

IV. Methodology

A. The Artifact

For traceability to be effectively used it must be part of the design process of the system [6, 1]. Without the necessary architectural structure that supports tracing, traceability may prove to be an extensively difficult task. For the design of the artifact several architectural assumption had to be made as to both assert which traceability links are less relevant and to automate the process of hiding such links and reducing the complexity of the trace link trees.

1) Architectural Assumptions

As stated by Antoniol et al. [9] traceability is a process of information retrieval, when the links are created automatically, and only a human analyst can make the final decision of whether the information retrieved is actually relevant. This leads us to the conclusion that absolute calculation of relevance in any possible scenario is unfeasible. This on the other hand does not necessarily mean that the information retrieved is not relevant. Several studies confirm that traceability needs to be essentially part of the architectural design process [1, 6], as to allow more efficent tracing. Similarly, other studies focus on tracing between a set of arbitrary artifact, i.e. between code and documentation as to improve the process of tracing [9]. The artifact that is constructed as an addition to Capra structures all of those arbitrary artifacts under a top-down architecture, following a parent-child approach that accomodates the source-target artifact methodology of traceability, explained in more detail in the example below. This allows for several assumptions to be made as to allow calculation of relevance between the affected artifact and the other artifacts that compose the system. The solution is inspired and follows similar rules to those of van de Berg et al. [1], namely the fact that the parent of the changed artifact is considered but not the children of the parent.

On figure 1 we can observe a traceability graphical representation generated by Capra. In the example, AAA.java is the parent node of BBB.java and CCC.java, CCC.java is the parent node of DDD.java, EEE.java and GGG.java, etc. If a change were to occur on artifact CCC.java, the relevant artifacts would be the children and distant children (FFF.java) of CCC.java, the parent of CCC.java (AAA.java) but not BBB.java. Since we change CCC.java, then the relation of {AAA.java, CCC.java}, {CCC.java, DDD.java}, {CCC.java, EEE.java}, {CCC.java, GGG.java} and {EEE.java, FFF.java} would be affected by the change but the relation {AAA.java, BBB.java} would not change. Hence a change in CCC.java would result in every artifact being affect except BBB.java.

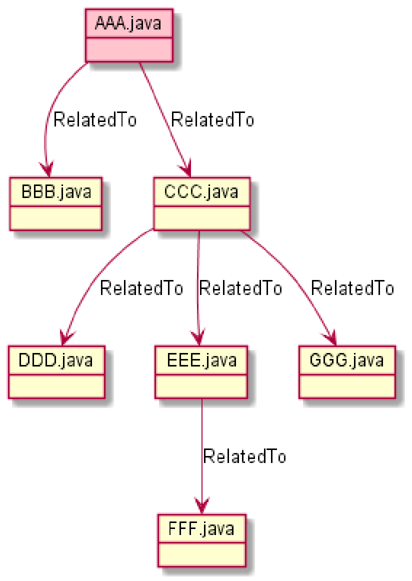


Figure 1. Capra generated traceability model

On Figure 2 a realistic system view example can be seen.

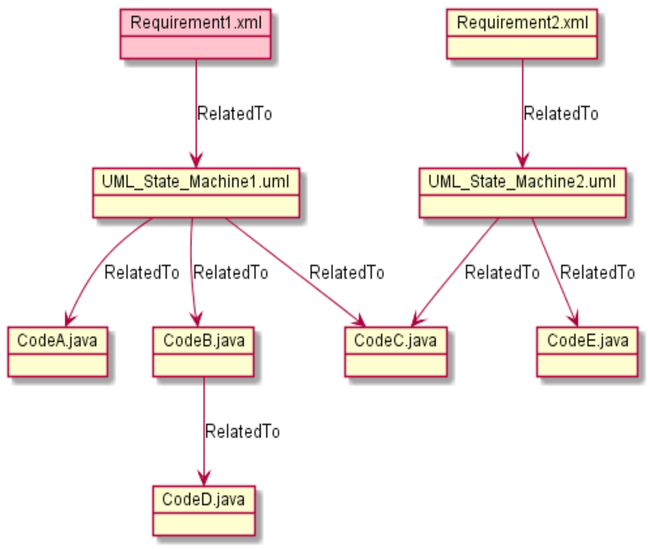


Figure 2. Capra generated traceability model on a system level / Unfiltered

Assuming a change happened in the ‘Requirement1.xml’ file and we wish to see only the relevant to that file components of the system. Applying the filter would give us a new graphical representation which now lacks the files that do not affect the ‘Requirement1.xml’ as it can be seen on Figure 3.

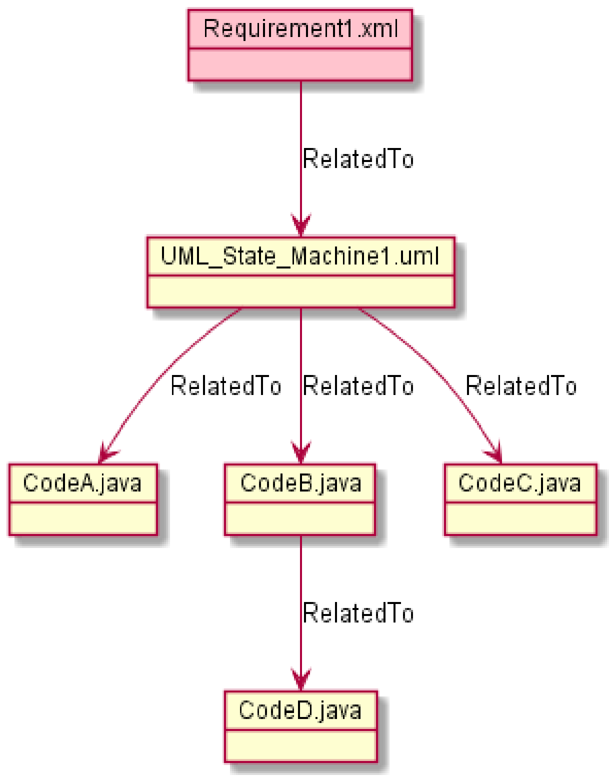


Figure 3. System level / Filtered

2) *Rules*

The system must be designed in a way as to follow the rules below:

- The traces must be directional, i.e. a source artifact cannot be a target artifact in relation to its target artifact. Vice-versa a target artifact cannot be a source artifact in relation to its source artifact. This rule is necessary because the approach cannot work unless the root, top-most parent, is detectable.
- A change of an artifact would result in a change of all of its children and distant children as well as the parent and distant parent of the changed artifact but not the children of the parent / distant parent. The rule is based on the approach and rules stated by van den Berg et al. [1] as to create the architectural assumptions necessary for the artifact to work.

B. *Technical Details*

The algorithmic approach used is a Depth First Search (DFS) to get all the child nodes of the selected artifact [13]. The decision was inspired from van den Berg et al. [1] and the explanation they provide of which parts of a software system is affected by a change of a certain node in a certain layer.

The DFS will search for a new child along a path from the selected node until there are no more child nodes before considering the next path. This opposed to another alternative that was considered, the breadth first search, which considers all outgoing paths from a node (all child nodes owned by this node) before continuing, saves memory and allows the algorithm to run on larger systems.

When all child nodes have been discovered the tree is searched top – down until it finds a connection leading to the selected node (the node suffering a change) and marking this as the new selected one. After this the algorithm recursively loops back and search for a connection to the new selected node.

Following this approach we end up with a complete tree consisting of the node that was originally identified as the source of the change, all the child nodes and all the direct parent nodes but no new child of any of the parent nodes.

C. *Implementation*

An experiment has been conducted to assert if the findings of this study can be used to make CIA efforts faster and more accurate. This has been realized through the possibility of hiding irrelevant trace links through the constructed artifact. For the experiment, students and developers from field of software development were used as sample groups. More in information about the experiment is presented in Section IV.

D. *Evaluation*

To evaluate the artifact hypothesis testing will be used. First the statistical data gathered from the experiment will be plotted in a box-plot to receive visual confirmation of the data and define outliers [11]. If outliers are discovered, they will be investigated in person separately and deemed whether they should remain or be omitted from the data. This is due to the fact that the outlier may be the result of an outside factor unimportant to the study or is actually important to the end result. Once the outliers have been handled, the data will be ran through a normality test as to determine the type of statistical test to be used. If the data is normally distributed, it will be ran through a parametric test to ensure power [11], if the data is not normally distributed, it will be ran through a non-parametric test. Further information on the hypothesis is presented in Section III.

V. EXPERIMENT DESIGN AND PREPARATION

A. *Subjects*

The subjects selected are developers with hand-on practical experience as well as students from Gothenburg University bachelor level software engineering as well as from master programs within the field of software engineering. None of the participants have experience in conducting a change impact analysis. The total sample consists of 12 subjects, divided randomly between a control group and a treatment group. The control group consists of 7 subjects, whereas the treatment group consists of 5 subjects. The majority is in the control group as a strong baseline for comparison was necessary that is close to the population.

B. Role of Participants

The participants will act as developers that are presented with an unknown system to them. The participants will be presented with a scenario of a change within one or more of the components of the system. The objective is to conduct a change impact analysis on the system and identify all affected artifacts by the change using the Eclipse traceability tool Capra.

C. Variables and Instruments

1) Variables:

- Independent Variables – as an independent variable we have one factor which is the Capra installation. For that that we have two levels: the standard Capra or the modified Capra.
- Dependent Variables – as dependent variables we have time spent per developer on CIA and the number of false negatives and false positives made while conducting CIA.
- Controlled Variables – As a controlled variable we have the graphical traceability representation of the system which is provided to us by the supervisor as to ensure a real-life example of a system.

2) Instruments:

- The participants are provided with a personal computer on which the experiment will be conducted.
- A standard out of the box Capra tool.
- A modified Capra installation that allows hiding less relevant trace links depending on the selected artifact.
- A set of instructions consisting of general information regarding the process of the experiment, traceability, change impact analysis and the task to be undertaken during the experiment.
- A Capra generated traceability model.
- A UML state diagram of the system [12].
- A file with C code related to the traceability model via a traceability link to the UML diagram [12].
- A questionnaire about the general skill set that the person possess relevant to the task. The questionnaire (see appendix Questionnaire) was used in order to determine that all participants had close to equal experience with CIA and the tools they were going to use.

D. Hypothesis

The general hypothesis of the experiment is that reducing the size and complexity of the trace link trees using the modified Capra (MC) would both improve the time spent on CIA and reduce the number of errors done while conducting a CIA. The main null hypothesis and the alternative hypothesis are stated as follows:

(i) $H_{0\ TPD}$: Time (SC) = Time (MC).
(i) $H_{1\ TPD}$: Time (SC) ≠ Time (MC).
(ii) $H_{0\ NFN}$: NumErrors (SC) = NumErrors (MC).
(ii) $H_{1\ NFN}$: NumErrors (SC) ≠ NumErrors (MC).
(iii) $H_{0\ NFP}$: NumErrors (SC) = NumErrors (MC).
(iii) $H_{1\ NFP}$: NumErrors (SC) ≠ NumErrors (MC).

In the first set of hypothesis (i) the null hypothesis states that the time per developer (TPD) while using the standard Capra (SC) installation to conduct change impact analysis is equal to the time per developer when conducting a CIA with the modified Capra (MC) installation. The alternative hypothesis in return rejects the null hypothesis by stating that the time per developer on CIA with the standard Capra is different from the time spent per developer on CIA with the modified Capra.

The second set of hypothesis (ii), is aimed at the number of false negatives (NFN) while conducting the CIA. A false negative is an artifact not included by the individual conducting the analysis but is actually part of the change. The null hypothesis states that the number of false negatives made while conducting a CIA with the standard Capra installation is equal to the number of false negatives made while conducting CIA with the modified Capra. The alternative hypothesis rejects the null hypothesis by stating that the number of false negatives made during CIA with the standard Capra is different from the number of false negatives made with the modified Capra.

Lastly, the third set of hypothesis (iii) is aimed at the number of false positives (NFP), following the same principle of hypothesis testing as the false negatives. A false positive is an artifact included by the individual to be part of the change set but should be excluded. Initially the number of errors were looked as a whole but after a discussion the decision was made to consider the NFN and NFP separately as to determine whether there is a difference on a more specific level of error making, hence leading to a more solid answer to RQ 1.2.

E. Design

The experiment follows a standard design of one independent variable with two values. The sample is randomly split into two groups. The control group used the standard out of the box Capra installation which serves as a baseline comparison, whereas the treatment group was given the modified Capra with the extra functionality of hiding less relevant trace links. All participants were provided with a set of instructions on how the experiment will be executed and a lecture on CIA to ensure experience equality.

The system that the subjects analyzed was the Emergency Braking and Evading System (EBEAS). The traceability tree for the system was reduced in size and did only contain 2 requirements. The requirements that were available contained information about emergency braking and object detection. In the requirements document there are also behaviors specified for evading in the case that there is no time to brake.

F. Validity Threats

We have defined three types of validity threats which were relevant and important for the results of the study: conclusion, internal and external validity. While running the four pilot tests to confirm that most validity threats were defined, several new ones became apparent. Below we will be discussing the threats to validity identified both before and after the four pilot tests that were conducted.

Most of the threats lie within the external validity as to if the findings are possible to generalize sufficiently due to the low experience levels within the control and treatment groups together with the small sizes of the groups. In order to equalize the experience levels we provided the groups with a set of instructions on what traceability and CIA is, what it is used for and an explanation of how to navigate the different artifacts to be considered in Capra.

All participants had as much time as they wanted to read the document and understand it as well as to ask questions to ensure they understood what they read. The system they were analyzing was a scaled down version of a real-life system which concern us with the fact that it may be too small thus rendering the algorithmic filter redundant.

The biggest concern is the threat to conclusion as the sample size is so small. If a larger sample size was used there is a possibility that the conclusion would be different than presented and we would find that there is a significant difference in the conclusion.

Concerning the threat to internal validity we have the set of instructions each participant was presented with. The instruction document consist of a set of explanations in both text and images to ensure the participant has numerous ways of understanding what is presented. Not all information may be presented in the most optimal way and the lengthy nature of the document provide a challenge for readers to remember all information within. Each participant had the option to go back to the instructions at any time in case they had forgotten or wanted to refresh their memory on a certain part of the instruction. Scaling down the instruction document was impossible as the system had to be understood and when conducting a CIA effort, documentation of the system is usually present. Other threats to internal validity include the environment and time of day the participant performed the analysis in the experiment. If an individual is affected by stress from any outside factor or noisy environment this could cause them to lack focus and perform worse than average and therefore give inaccurate results. To mitigate this all participants were asked to choose a time and place to perform the experiment so that the environment was of best nature for them to perform. Individuals who participated remotely had the comfort of the home and outside of workhours to be able to fully concentrate on the task at hand. Others chose times where they had nothing else to focus on and a quiet place such as a library. Another threat that was identified is related to Eclipse and the technical aspects of the experiment. When opening certain files Eclipse has a tendency to crash or take an immense amount of time to open the file which could result in corrupt data if the program needs to restart due to a crash or freeze. This threat could not be mitigated as Capra is an Eclipse plug-in and we are unaware of the cause of the crashes and freezes and this was observed on different computers.

VI. Statistical Results

A. Statistical data

The dependent variables observed are the time spent per developer in minutes (TPD), average number of false negatives made (NFN) and average number of false positives made (NFP).

In figure 4 we observe the TPD and the following data:

- Mean(SC) - 27.71429
- Mean(MC) – 30
- Variance(SC) - 89.90476
- Variance(MC) - 91.33333
- Standard Deviation(SC) - 9.481812
- Standard Deviation(MC) - 9.556847

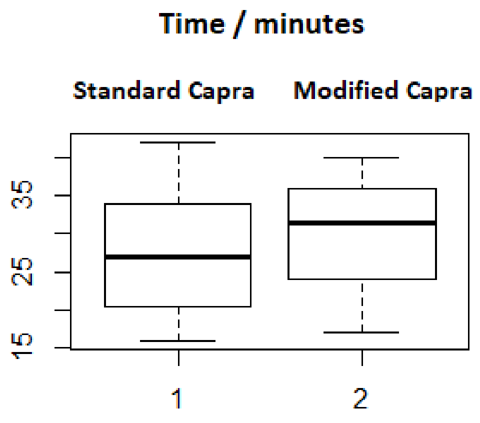


Figure 4. TDP

In figure 5 we observe the NFN and the following data:

- Mean(SC) - 4.857143
- Mean(MC) – 5
- Variance(SC) - 5.47619
- Variance(MC) – 2
- Standard Deviation(SC) - 2.340126
- Standard Deviation(MC) - 1.414214

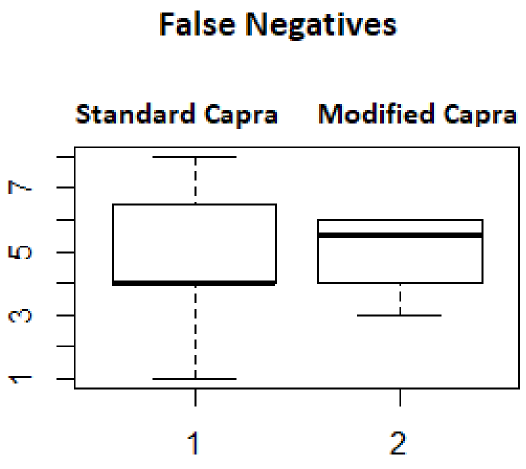


Figure 5. NFN

In figure 6 we observe the NFP and the following data:

- Mean(SC) – 2
- Mean(MC) - 1.25
- Variance(SC) - 2.333333
- Variance(MC) - 2.25
- Standard Deviation(SC) - 1.527525
- Standard Deviation(MC) - 1.5

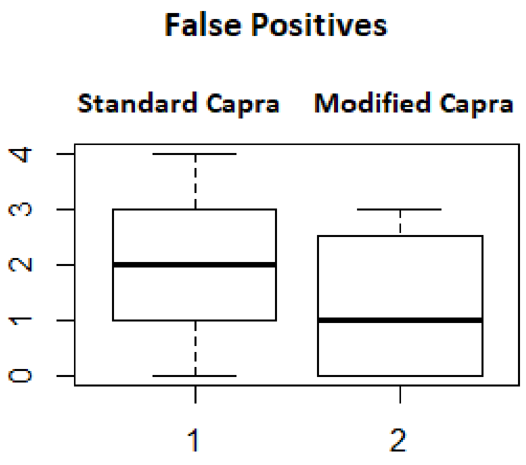


Figure 6. NFP

B. Outlier handling

In figure 7 two outliers can be seen on the false negatives boxplot. Both outliers were investigated and it was noted that the first outlier on the bottom which performed exceptionally well was due to special focus on the instruction document and familiarization with the system, which we deeply encouraged and gave sufficient time for. Due to this reason the decision was made to keep the lower outlier as a valuable data point.

The second outlier that did significantly worse than the rest had a reason for that as well. The subject misunderstood how the traceability graphical representation worked, subsequently leading to incorrect results. Throughout the 4 pilot tests and all other subjects, this is the only case of misunderstanding the representation, hence the data point has been removed.

After the removal of the second outlier the first outlier disappeared as it can be seen on figure 3.

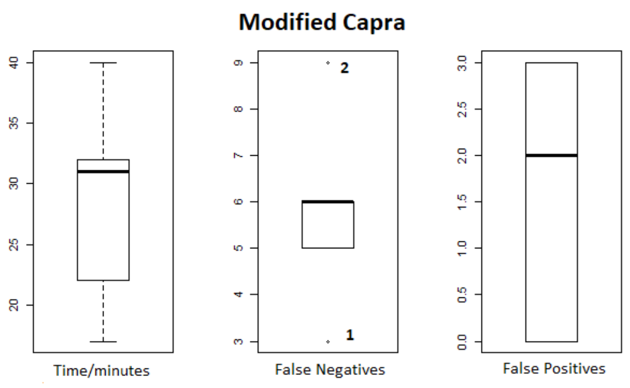


Figure 7. Outliers

C. Statistical results

Due to all data sets lacking a normal distribution, a non-parametric test would give more accurate results. Additionally, the sample size used is too small for it to have sufficient power for a parametric test to give accurate results. To confirm the distribution of the data, a normal QQ plot was made together with a line of the distribution. The Man-Whitney U test allows us to decide whether the population distributions are identical without assuming normality. The Man-Whitney U test itself complies with the hypothesis as it tests for whether two dependent groups have identical distributions, which is what we wish to discover. Additionally non-parametric tests are more reliable than parametric tests when the sample size is below 20. In the case of this study the sample size is too low for a parametric test, hence the conclusion was reached to use a non-parametric test. The only downside with the Man-Whitney U test is that if there are more ties in the ranks than acceptable, the accuracy of the test would diminish.

D. Wilcox test results

a) TPD results

- P-value: 0.9273
- Level of significance: < 0.05

b) NFN results

- P-value: 0.3641
- Level of significance: < 0.05

c) NFP results

- P-value: 0.49
- Level of significance: < 0.05

Since the TPD p-value is not less than 0.05 we do not reject the null hypothesis. Likewise, both the NFN and NFP due to the p-value being bigger than 0.05 we do not reject the null hypothesis. The TPD, NFN and NFP did not show a statistically significant difference between the Standard Capra and the Modified Capra with the filtering of the less relevant trace links.

VII. Discussion

The results of the experiment turned out not as expected and the null hypothesis was not rejected. Overall the results were quite similar but there is a considerable amount of room for interpretation. Namely all subjects had no experience in conducting a CIA effort, this could potentially have influenced the results. The decision was made to use inexperienced in CIA developers as to ensure experience equality and less varying data. A relative level of experience equality could have been achieved if all subjects had a set amount of time (in experience) in conducting a CIA but we were unable to access such people. Without experience in CIA the subjects had no set approach to follow besides the information we provided regarding traceability, hence leading to the subjects being 'lost', unable to properly make a connection between the traceability graphical representation and the other artifacts (the UML diagrams and C code). This situation occurred due to the participants exploring the system (Capra) beyond the necessary components despite being instructed otherwise in the instruction document. This is proven further as the subjects that spent their time wisely and properly examined the instructions document with all the necessary information, which we encouraged and gave sufficient time for, showed overall better results. Furthermore, the experiment had low power due to the limited sample size, with a sample of 20 or greater we believe that the gap of the false positives and false negatives would reduce significantly and encourage further studies in the field of traceability and change impact analysis. Nevertheless to answer RQ1, enriching traceability links with the ability to hide less relevant trace links did not improve the process of change set identification.

a) Control Group – 7 participants

ID	Minutes	False Negative	False Positive
1	35	4	0
2	18	4	2
3	42	8	3
4	23	6	4
5	33	1	2
6	16	7	0
7	27	4	3

b) Treatment Group without outliers – 4 participants

ID	Minutes	False Negative	False Positive
1	40	6	3
2	17	5	2
3	32	3	0
4	31	6	0

B. *Time Per Developer*

The results show that the sample that used the standard Capra installation did overall better time-wise. We believe that using inexperienced analysts impacted the TDP results the most. When having no experience with change impact analysis or Capra, the issue regarding became quite apparent early on while observing the pilot tests. While the subjects had the luxury of hiding trace links, they were at the same time burdened with handling more functionality. While the system shows only the relevant trace links, the participants still had to discover the parent and they spent a significant amount of time doubting over which component would be the parent of the change. This lead us to believe that we should make it 'crystal clear' where the change is happening and at the same time we didn't not wish to make it too simple. The scenarios were changed and the instruction document contained more system information but that did surprisingly not solve the issue, the participants still had doubts, in the sense of whether an artifact is part of the change set or not, and were exposed to a new to them system which they had to use to clear to those doubts, ascertain whether an artifact is part of the change set or not. We believe this to have caused the results regarding time to have been influenced up to a diminishing point where the modified Capra performed worse. To answer RQ 1.1, hiding less relevant trace links did not reduce the time spent per developer on CIA.

C. *False Negatives and False Positive*

While the null hypothesis was not rejected yet again for both dependent variable, we believe that a repeated study with better sample size would give different results. The results show a small decline in errors and we believe this decline to increase as the sample would increase. A sufficient sample of 20 people should have enough power to show potentially better results. Aside from that a further impact on these results is the issue mentioned above that affected the time per developer. As stated earlier the subjects that spent their time wisely and paid sufficient attention and respect to the instruction document that contained system information performed better than the rest, this is due to the rest focusing too much on the burden of the extra functionality and limiting their view to the graphical representation, dismissing the instruction document. Indeed documentation is not the most compelling part when doing analysis and the bigger the document the more repulsive it is and may have discouraged the subjects from paying sufficient time to it, but it is an essential part of any analysis activity and could not be reduced. We also noted that the subjects did not always use the filter in the way they were instructed to do. As opposed to selecting the artifact in the artifact wrappers view and using this view to find the related artifacts that could be affected by the change they went into many different artifacts and made guesses depending on the names of the artifacts. This was a more common occurrence in the second scenario where the subjects were analyzing an artifact further down in the traceability tree. To answer RQ 1.2, hiding less relevant trace links did not reduce the amount of errors made during the process of identification of artifacts.

VIII. Conclusion and Future Work

Even though the results far from what was expected on a sample of inexperienced analysts, we believe that the approach could potentially reduce the time spent on CIA on the condition that the system is sizeable enough and the analysts using the system are experienced. Both the false negatives and false positives had no statistically significant difference on the results but we believe that the small decline we see in the results would prove significant with a sample of experienced analysts that provides sufficient power to represent the actual population. This is leading us to believe that there is merit in further studies on how reducing the links observed would improve change impact analysis.

A further study could be made that covers the gaps discovered in this study and use the identified drawbacks to provide more reliable results. Further studies in this area should pay special attention to the sample picked as to represent the actual population and possibly provide training in using Capra as to reduce the level of unfamiliarity that causes constant doubt in the subjects. While experienced analysts are aware of how to approach the system that they have to analyze, the feeling of unfamiliarity could even then prove to be a considerable obstacle.

Acknowledgment

We would like to thank our supervisors Salome Maro and Jan-Philipp Steghöfer for the consistent feedback and additional support in setting up and learning Capra.

References

[1] van den Berg, K.G., 2006. *Change Impact Analysis of Crosscutting in Software Architectural Design.* In: Workshop on Architecture-Centric Evolution (ACE 2006), 3-7 July 2006, Nantes, France. pp. 1-15.

[2] Borg, M., de la Vara, J.L. and Wnuk, K., 2016, September. Practitioners' Perspectives on Change Impact Analysis for Safety-Critical Software–A Preliminary Analysis. In *International Conference on Computer Safety, Reliability, and Security* (pp. 346-358). Springer International Publishing.

[3] Ghosh, S.M., Sharma, H.R. and Mohabay, V., 2011. Study of Impact Analysis of Software Requirement Change in SAP ERP. *International Journal of Advanced Science and Technology*, *33*, pp.95-100.

[4] Lehnert, S., 2011, September. A taxonomy for software change impact analysis. In Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution (pp. 41-50). ACM.

[5] Arnold, R.S. and Bohner, S.A., 1993, September. Impact analysis-towards a framework for comparison. In *Software Maintenance, 1993. CSM-93, Proceedings., Conference on* (pp. 292-301). IEEE.

[6] Hughes, T. and Martin, C., 1998, March. Design traceability of complex systems. In *Human Interaction with Complex Systems, 1998. Proceedings., Fourth Annual Symposium on* (pp. 37-41). IEEE.

[7] Li, Y., Li, J., Yang, Y. and Li, M., 2008, May. Requirement-centric traceability for change impact analysis: a case study. In *International conference on software process* (pp. 100-111). Springer Berlin Heidelberg.

[8] Z. Jianjun, H. Yang, L. Xiang, and B. Xu. "Change impact analysis to support architectural evolution." Journal of software maintenance and evolution: research and practice, vol. 14, no 5, 2002, pp. 317-333.

[9] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A. and Merlo, E., 2002. Recovering traceability links between code and documentation. *IEEE transactions on software engineering*, *28*(10), pp.970-983.

[10] Maro, S. and Steghöfer, J.P., 2016, September. Capra: A Configurable and Extendable Traceability Management Tool. In *Requirements Engineering Conference (RE), 2016 IEEE 24th International* (pp. 407-408). IEEE.

[11] Wohlin, C. and Höst, M., 2001. Special section: controlled experiments in software engineering.

[12] Becker, S., Dziwok, S., Gerking, C., Heinzemann, C., Thiele, S., Schäfer, W., Meyer, M., Pohlmann, U., Priesterjahn, C. and Tichy, M., 2014. The MechatronicUML design method-process and language for platform-independent modeling. *Software Engineering Group, Heinz Nixdorf Institute, University of Paderborn*, *134*.

[13] Tarjan, R., 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing*, *1*(2), pp.146-160.

[14] Cleland-Huang, J., Hayes, J.H. and Domel, J.M., 2009, May. Model-based traceability. In *Traceability in Emerging Forms of Software Engineering, 2009. TEFSE'09. ICSE Workshop on* (pp. 6-10). IEEE.

[15] Gotel, O., Cleland-Huang, J., Hayes, J.H., Zisman, A., Egyed, A., Grünbacher, P., Dekhtyar, A., Antoniol, G., Maletic, J. and Mäder, P., 2012. Traceability fundamentals. In *Software and Systems Traceability* (pp. 3-22). Springer London.

[16] Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E. and Christina, S., 2005, May. Goal-centric traceability for managing non-functional requirements. In *Proceedings of the 27th international conference on Software engineering* (pp. 362-371). ACM.

[17] Hayes, J.H. and Dekhtyar, A., 2005, November. Humans in the traceability loop: can't live with'em, can't live without'em. In *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering* (pp. 20-23). ACM.

[18] Bohner, S.A. and Arnold, R.S., 1996. An introduction to software change impact analysis. *Software change impact analysis*, pp.1-26.

Appendix I

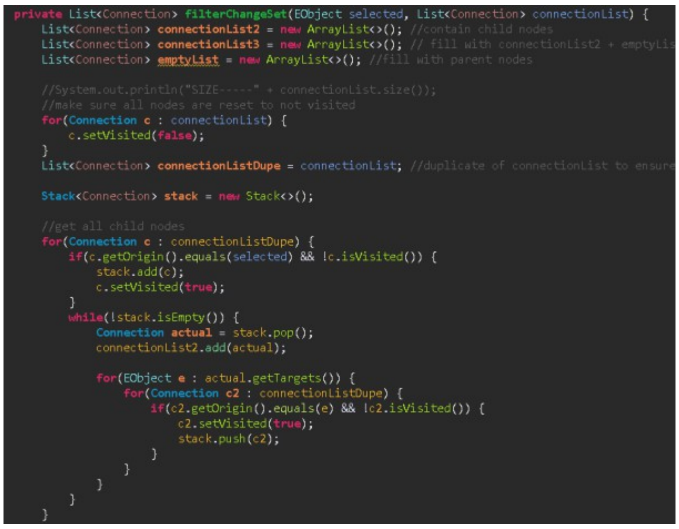
Instructions

Appendix II

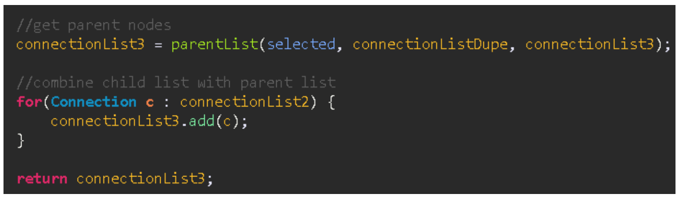
Questionnaire

Appendix III

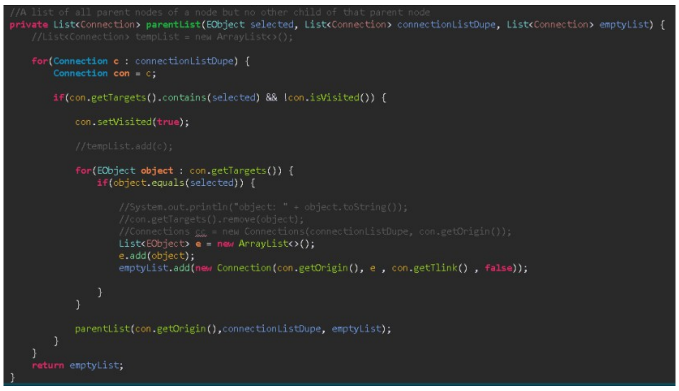
Filtering algorithm



```java
private List<Connection> filterChangeSet(EObject selected, List<Connection> connectionList) {
    List<Connection> connectionList2 = new ArrayList<>(); //contain child nodes
    List<Connection> connectionList3 = new ArrayList<>(); // fill with connectionList2 + emptyLis
    List<Connection> emptyList = new ArrayList<>(); //fill with parent nodes

    //System.out.println("SIZE-----" + connectionList.size());
    //make sure all nodes are reset to not visited
    for(Connection c : connectionList) {
        c.setVisited(false);
    }
    List<Connection> connectionListDupe = connectionList; //duplicate of connectionList to ensure

    Stack<Connection> stack = new Stack<>();

    //get all child nodes
    for(Connection c : connectionListDupe) {
        if(c.getOrigin().equals(selected) && !c.isVisited()) {
            stack.add(c);
            c.setVisited(true);
        }
        while(!stack.isEmpty()) {
            Connection actual = stack.pop();
            connectionList2.add(actual);

            for(EObject e : actual.getTargets()) {
                for(Connection c2 : connectionListDupe) {
                    if(c2.getOrigin().equals(e) && !c2.isVisited()) {
                        c2.setVisited(true);
                        stack.push(c2);
                    }
                }
            }
        }
    }
```



```java
//get parent nodes
connectionList3 = parentList(selected, connectionListDupe, connectionList3);

//combine child list with parent list
for(Connection c : connectionList2) {
    connectionList3.add(c);
}

return connectionList3;
```



```java
//A list of all parent nodes of a node but no other child of that parent node
private List<Connection> parentList(EObject selected, List<Connection> connectionListDupe, List<Connection> emptyList) {
    //List<Connection> tempList = new ArrayList<>();

    for(Connection c : connectionListDupe) {
        Connection con = c;

        if(con.getTargets().contains(selected) && !con.isVisited()) {

            con.setVisited(true);

            //tempList.add(c);

            for(EObject object : con.getTargets()) {
                if(object.equals(selected)) {

                    //System.out.println("object: " + object.toString());
                    //con.getTargets().remove(object);
                    //Connections cc = new Connections(connectionListDupe, con.getOrigin());
                    List<EObject> e = new ArrayList<>();
                    e.add(object);
                    emptyList.add(new Connection(con.getOrigin(), e , con.getTlink() , false));

                }
            }

            parentList(con.getOrigin(),connectionListDupe, emptyList);
        }
    }
    return emptyList;
}
```

APPENDIX I

User manual:
Supervisors – Marcus Nilsson & Kristiyan Dimitrov
The process will be recorded using screencaption software!
If you have any questions do not hesitate to ask. There will be no trick questions in the experiment.

Abbreviation list & terminology explanations:
Artifact – Items/components/classes/diagrams within the project
Changeset – all artifacts affected
Child – A child is a node directly below another node. A -> B -> C. B is a child of A and C is a child of B.
CIA – Change Impact Analysis
EBEAS– Emergency Braking and Evading Assistance
Node – An item in the graph
Parent – A parent is a node directly above another node. A -> B -> C. A is the parent of B and B is the parent of C.
SA – Situational Analysis
Trace / tracelink – the arrow between two nodes.
Transitive – Meaning any node that has any connection to a specific node.
V2V – Vehicle to Vehicle

Change Impact Analysis (CIA):
This is the process of analysing what artifacts are affected by a proposed change. CIA is used to help understand and estimate how much effort will have to be put into a change. This change could be that the customer wants to change the appearance of a certain part of a website or have an entirely new feature added or have something removed as well.
This helps a programmer or management understand which parts of the project are going to be affected and have to be considered when making this change or completely ignore the change as the impact would be too costly in both time and funds.

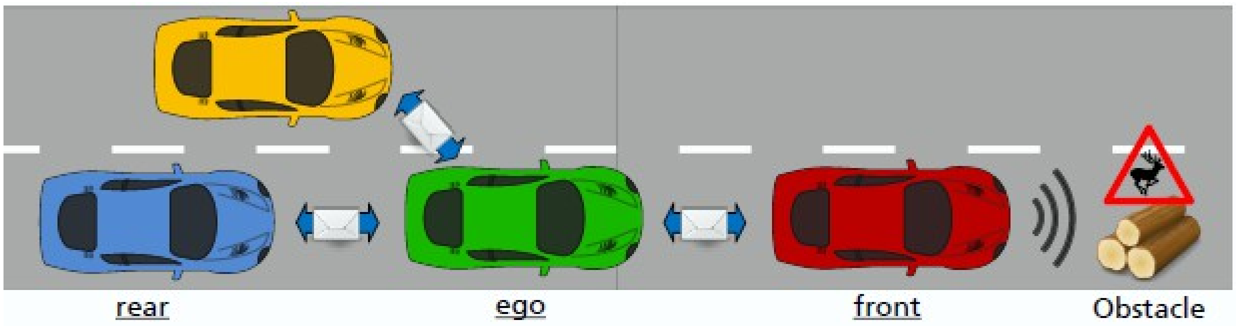
Traceability:
In software systems we can use traceability to help with CIA. A trace link is a connection between two or more nodes or artifacts indicating that they have a relationship. Examples of artifacts are:

- Requirements
- Models
- Code
- Tests

The traceability links are used to help understand from where an artifact originates. For example a piece of code that has a traceability link from a requirement we know that this piece of code will realize a

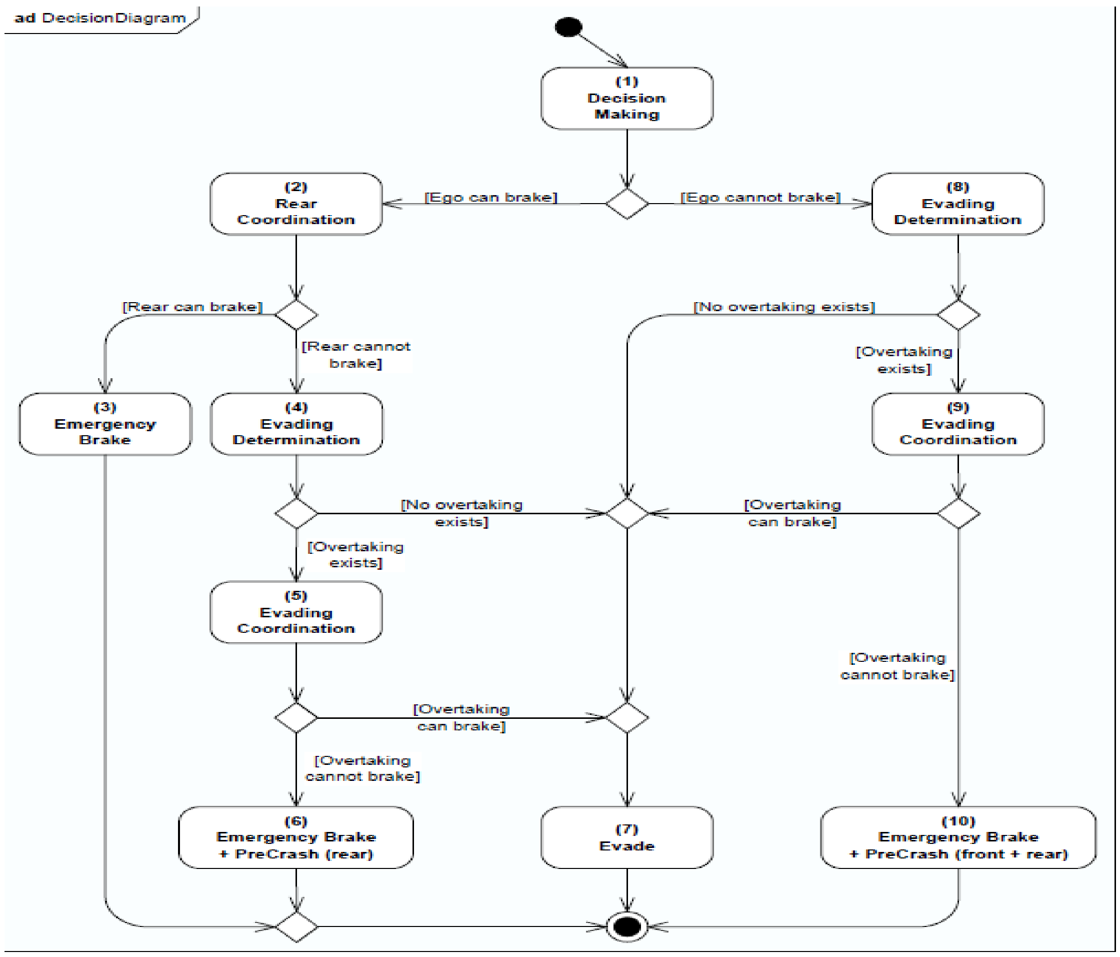
requirement listed in that artifact. The requirement artifact can for instance be a PDF document handling layout. Then we know that the piece of code will have something to do with layout.
Traceability can also be used outside of software projects. An example could be within the food industry where traceability can be used to track the origin of certain products and who handled it in the process of reaching the consumer.

EBEAS brief explanation:



The system you are to analyse handles emergency braking and evading.
Ego – this car. (your car).
Front – the car in front.
Rear – the car behind you.

If an obstacle is detected the car will check if it is safe to brake by sending messages backwards to see if the distance and condition will be ok to perform a brake. This is a V2V message. The Ego car will check its own speed etc to determine if it's safe for the car in front to brake and send back a reply. That car will also do the same backwards and see if it can indeed brake as well as a result of the car in front braking. If it cannot brake safely the car will see if it is possible to evade instead as the distance for evading is shorter than braking. If nothing is safe there's a precrash system kicking in which raises seats and tighten seatbelts etc.
Below you will see the decision making for the ego car in case you feel this may be helpful for your understanding.



(decision making for the ego car)

Experiment information:

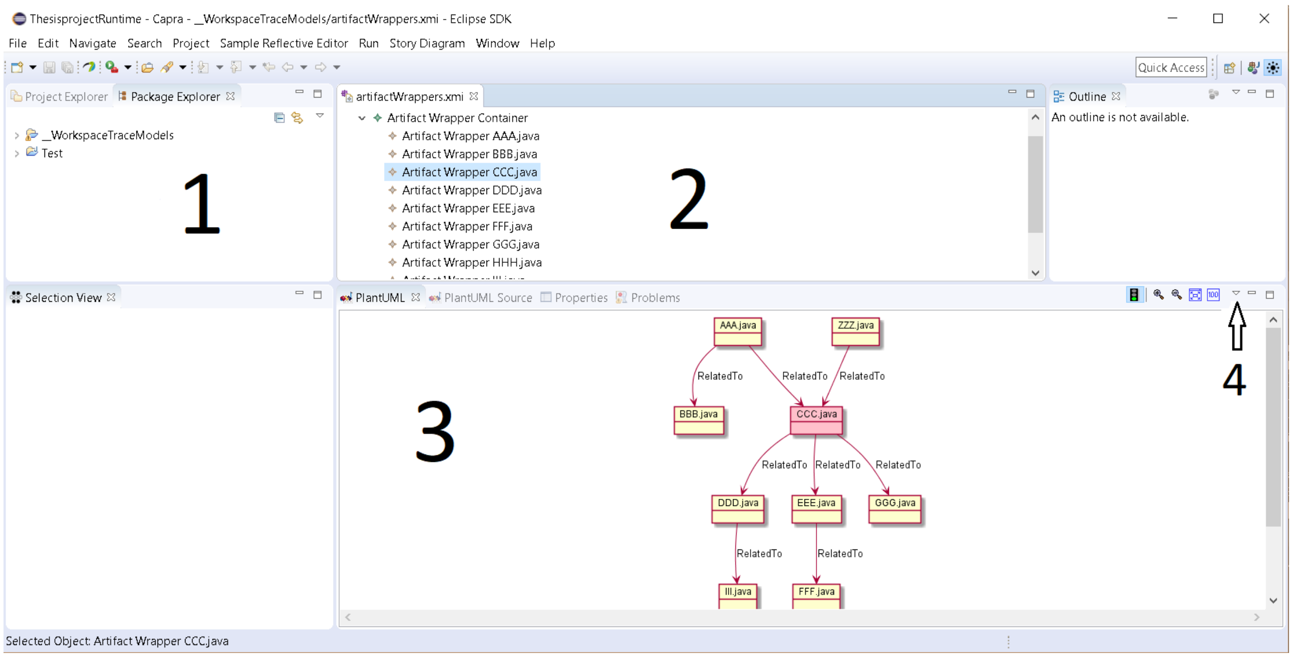
You will be presented with a set of instructions that a change has to be made to a part of the system. Your task will be to find the artifact handling this scenario (see view 2 below), select this item and then in view 3 find the related artifacts by following the tracelinks from the selected item (pink).
When you have found an item you think will be affected you shall note the name of the item on a piece of paper.
(If you are doing this remotely either note down in a text document or pause the next trace until the supervisor have noted down the name of the item)

There will be two parts of the program you need to use. Artifact wrappers if there is a change to anything related to SA.
And under examples the main folders used will be <Package>ObstacleDetection and <Package>EmergencyBraking.

Under these folders there will be different files etc. It is in your task to find the correct files based on the naming and what we ask for in the questions.

<u>Capra instructions:</u>

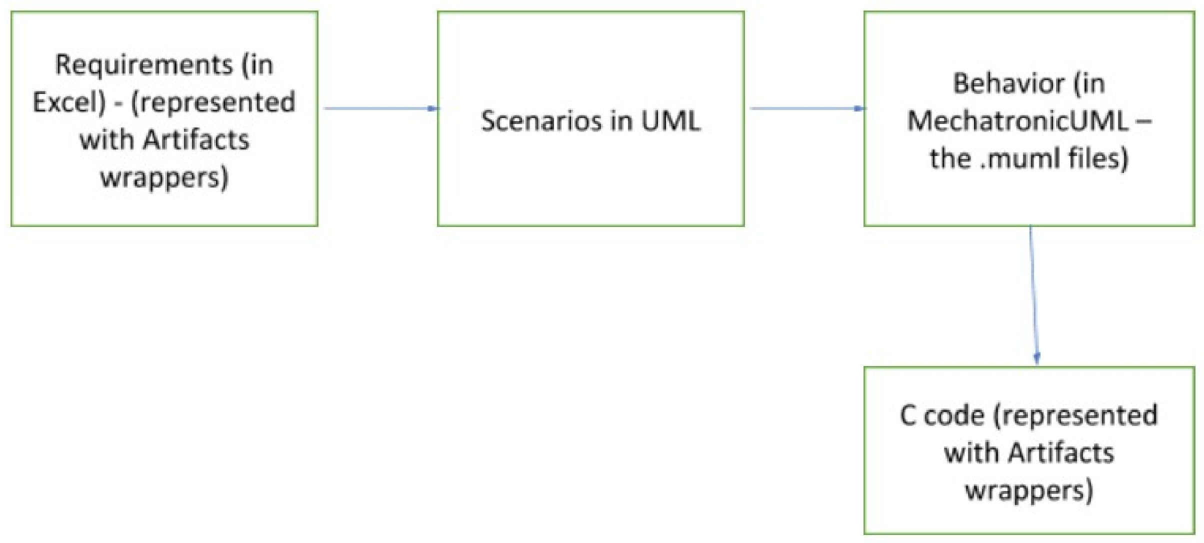
Layout:


ThesisprojectRuntime - Capra - _WorkspaceTraceModels/artifactWrappers.xmi - Eclipse SDK
File Edit Navigate Search Project Sample Reflective Editor Run Story Diagram Window Help
Quick Access
Project Explorer
Package Explorer
_WorkspaceTraceModels
Test
1
artifactWrappers.xmi
Artifact Wrapper Container
Artifact Wrapper AAA.java
Artifact Wrapper BBB.java
Artifact Wrapper CCC.java
Artifact Wrapper DDD.java
Artifact Wrapper EEE.java
Artifact Wrapper FFF.java
Artifact Wrapper GGG.java
Artifact Wrapper HHH.java
2
Outline
An outline is not available.
Selection View
PlantUML
PlantUML Source
Properties
Problems
3
AAA.java
ZZZ.java
RelatedTo
BBB.java
CCC.java
DDD.java
EEE.java
GGG.java
III.java
FFF.java
4
Selected Object: Artifact Wrapper CCC.java

1 – This is the project that is going to be analyzed
2 – All artifacts in the project. Here you will select an item to show the connections associated with this item in window 3. This is where you will look for an item we ask you to perform a change impact analysis on.
3 – Traceability tree with directional connections. Pink is a selected item.
4 – The arrow button will allow you to change between transitive and filtered view. Depending on the scenario you are presented with you will use either one. If you have the transitive scenario you should not use this function.
If you are changing between transitive view and filtered view you will have to click on the artifact again to update the diagram accordingly. If you wish to display only the direct connected nodes simply turn off transitive view and click the node again in view 2.

An arrow in the diagram goes from a parent to a child. The node the arrow is pointing to is the child.

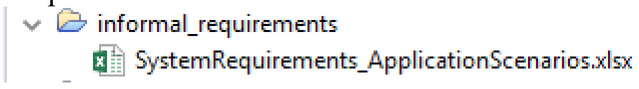
Rules:
You will write down any node names that you think will be part of the changeset. Use the diagram to see what is connected together with the code to determine whether or not this file will actually be connected. There are also statecharts which you may use to determine or verify behaviour in the system.
Tip:
Check the diagram first to find every artifact. Then check code to determine if the artifact will be changed or not.
Tabelle1\::2 means table 1 row 2 in the requirements file.
Tabelle1\::3 means table 1 row 3 in the requirements file.


Requirements (in Excel) - (represented with Artifacts wrappers)
Scenarios in UML
Behavior (in MechatronicUML – the .muml files)
C code (represented with Artifacts wrappers)

Scenarios realize requirements, behaviors realize scenarios and C code realize behaviors.

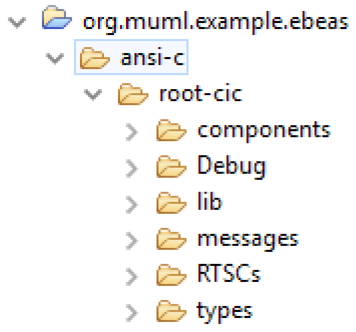
SEE NEXT PAGE FOR THE EXPERIMENT QUESTIONS:

Requirements can be found here:



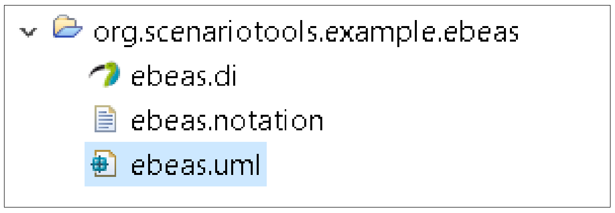
```
informal_requirements
    SystemRequirements_ApplicationScenarios.xlsx
```

Code can be found here:



```
org.muml.example.ebeas
    ansi-c
        root-cic
            components
            Debug
            lib
            messages
            RTSCs
            types
```

Models can be found here:



```
org.scenariotools.example.ebeas
    ebeas.di
    ebeas.notation
    ebeas.uml
```

Note that there are also statecharts under "realtimestatechart" and a component diagram under "component" which may be of use to you.
(All are found in view 1)

<u>Scenario:</u>

- Tabelle1\::2 (requirement handling obstacle detection) has to change. This handles a situation where the object is too close to the last point of brake so that there is no time to negotiate if braking is safe. The change is that the leading vehicle should no longer warn the following vehicle about the emergency braking that it's about to perform. Analyse which artifacts would be affected.
  [changeset 1]

- When making the decision of emergency braking we want to consider the outdoor temperature in order to be able to change the threshold of the time before the decision goes into evading instead of braking. If it's cold the road might be slippery and braking distance increase etc. Analyse which artifacts will be affected if we add this component in the SituationAnalysisDecisionsSA_Decisions_PortStateChart_ProcessStep.
  [changeset 2]

Appendix II

Change Impact Analysis Experiment

Full Name:_______________________ **Signature:**________________

Please cross the correct answer on a scale of 1 to 5.

1. How much experience do you have in programming?

	1	2	3	4	5	
Poor	●	●	●	●	●	**Excellent**

2. How much experience do you have with UML diagrams?

Poor	●	●	●	●	●	**Excellent**

3. How much experience do you have in conducting change impact analysis?

Poor	●	●	●	●	●	**Excellent**

4. How much experience do you have in using traceability for software systems?

Poor	●	●	●	●	●	**Excellent**

5. How much experience do you have in using the Eclipse IDE?

Poor ● ● ● ● ● **Excellent**

6. Have you ever used Eclipse Capra before?

Yes **No**

Appendix III

Code can be found here:

https://www.dropbox.com/sh/tz7ijctp6qdt7y0/AADMk8yjPhG-3ID-poQSnhRwa?dl=0