CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Evaluating trace link visualizations

Master's thesis in Computer science and engineering

Fredrik Johansson
Themistoklis Ntoukolis

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Evaluating trace link visualizations

Fredrik Johansson
Themistoklis Ntoukolis

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Evaluating trace link visualizations

Fredrik Johansson
Themistoklis Ntoukolis

Supervisor: Jan-Philipp Steghöfer, Department of Computer Science and Engineering
Examiner: Regina Hebig, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Evaluating trace link visualizations

Fredrik Johansson
Themistoklis Ntoukolis
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Traceability has become a very important part of the software development lifecycle and therefor a lot of research has been done and is being done about it. Our aim with this paper is to explore and evaluate the effect that different traceability visualizations have on the workload and find the most optimal visualizations that users will benefit more by using in their projects. We used design science research in this paper and started with a survey of industry professionals so that we could get an overview of traceability and visualizations use in real life. This information was then compared with the results of the experiment that was conducted as a second step. We created prototype visualizations that were used for the experiment in which 16 students took part. These two iterations provided us with qualitative and quantitative results that showed us there is no difference in workload between the visualizations and that perception and familiarity have an influence in which visualization is preferred. Furthermore the results indicated that no matter the visualization the information should be provided by them is the same.

Keywords: Traceability, Eclipse Capra, traceability visualizations, computer science, engineering, project, thesis, requirements, change impact analysis, matrix, sunburst.

Acknowledgements

We would like to thank deeply our supervisor professor Jan-Philip Steghöfer. He was always available for us with his thoughts and feedback about our work. More importantly for being extremely patient and understanding with us even in times when our work was not going as it should. Also we would like to thank all the industry engineers that gave their time to answer our questionnaire and the students that took time from their schedule to conduct the experiment, their help is deeply appreciated. Lastly but not least we would like to thank our examiner professor Regina Hebig and our opponents Arvid Björklund and Johannes Edenholm for their feedback and criticism of our work and making sure that our study is of a proper standard.

A big thank you goes to my family for keeping up with me this whole time. (TN)

Fredrik Johansson & Themistoklis Ntoukolis, Gothenburg, July 2020

Contents

List of Figures

List of Tables

1
Introduction

1.1 Context

Over the last few years, traceability has gained a lot of importance since it supports a lot of different software engineering tasks. According to Gotel and Finkelstein [1], requirements traceability is the ability to "describe and follow the life of a requirement in both a forwards and a backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use), and through periods of ongoing refinement and iteration in any of these phases". This is a very specific definition, a more general definition is that traceability is the ability to relate artifacts created during the development of a software system to describe the system from different perspectives and levels of abstraction with each-other, the stakeholders that have contributed to the creation of the artifacts and the rationale that explains the form of the artifacts [2].

Traceability is important because it supports and is a key functionality for a lot of software engineering activities including regression testing, requirements validation, change impact analysis, program comprehension, etc. For example:
1. Change impact analysis is an activity that focuses on identifying the potential consequences of change or estimating what documentation adjustments are needed to facilitate a specific change. Based on the definition of the activity we can derive that what change impact analysis needs from traceability is the capability to support top-down and bottom-up tracing from a components perspective. Also, an approach that has a rich set of coarse and fine-grained relationships within and across different levels of software models is needed [3]. As far as visualization of traces is concerned, what change impact analysis needs is to be able to show the relationships and keep track of all the ones that are relevant to a specific change in a very connected and cluttered image.
2. Software evolution is another area of software development where traceability is used. The purpose of software evolution is the repeated update of the software and ensuring the relevance, reliability, and flexibility of the system. In this case, traceability must contain all the traces linking the various artifacts, enabling the forward and backward tracking of them. Also it needs to be able to follow those links throughout the different versions (repositories) of the artifacts.
3. One of the main activities that use traceability is program comprehension. This is a necessary activity for every developer in order to understand the project and the code as a whole [4]. This means a user should be able to understand and comprehend the system, even if he is not familiar or has not worked with it before. Knowing this, a traceability visualization must allow moving around the hierarchy of the links without losing the whole view and be able to bring the code to a more comprehensible form so that it can support comprehension [5].

As can be seen by the examples above a lot of different activities tend to use traceability, that is why traceability can be achieved by using different techniques like cross-

referencing documentations, record-keeping using Excel spreadsheet, etc. All the above techniques are mainly used manually which together with the wide variety of different documentation formats make tracing links prone to errors, also manually maintaining the traceability becomes cumbersome when a project grows. Big projects with thousands of trace links can become tedious to understand and over time the maintenance and update of the links and the project as a whole will become difficult, resulting in inaccurate relations. Therefore, a lot of traceability tools for creating and maintaining trace links are proposed like RETRO [6], MultiVisio Trace [7], D3TraceView [8], Eclipse Capra [9], etc.

These tools come with some kind of visualization ability to help users with their tasks. But even though these visualizations are there to help the engineers, mostly when big projects are involved they are not used at all or are used very little [10]. This is because trying to fit hundreds of links in a limited image space makes it crowded and hard to pinpoint the necessary things needed by a user and as a result traceability visualization transforms to more of a hindrance rather than a help. Our research will focus on understanding the problems of visualizing trace links and improving the way visualization assists the user in its projects by proposing a prototype that does that and evaluating it.

1.2 Goal

As mentioned in the previous section, visualizing projects with a lot of trace links is difficult. This makes it easier for errors and mistakes to happen and the visualization does not have the beneficial purpose it should have. So to help engineers with their tasks and to help them better comprehend the project they are working on, a good visualization is needed. Visualizations of the traceability links should fulfill the engineers' needs.

The main goal, therefore, is to create traceability visualizations for projects that users will use more often as they can get more benefits from them. Considering that traceability is a very important activity of a software development lifecycle it becomes important that engineers and analysts spend effort on it and the time spent is as efficient as possible. So by improving their interaction with traces, can influence the rest of the development activities positively. If the analyst can spend less time analyzing, trying to understand and upkeeping the traceability links, it will allow him to focus more efficiently on the rest of the development processes. From this goal the two research questions that are the basis for this study and that we will try to answer emerge:

RQ1: What graphical information is useful for an analyst when using a traceability tool?

RQ2: What impact do different visualizations have on work efficiency?

These questions are analyzed in more depth in Section 3.

1.3 Purpose

The purpose of this study is to explore and evaluate the effect that different traceability link visualizations have on human error and on the workload needed to create and maintain traceability links. The focus will be especially in working with projects that have a high number of links. Visualizing these projects will impact comprehension of traceability links and improve the analysts' work effectiveness. Since this research will handle traceability, the main use focus is on software engineering activities that use traceability. Visualizing the links can also help engineers and analysts with other engineering practices for example by making it easier to check the evolution and change of a project, help in the comprehension of software or to do a project's change impact analysis.

1.4 Method

In this paper, we start the inquiry by creating and conducting a survey with field professionals. The main purpose of this survey is to help us figure out which visualizations practitioners think are helpful and to some extent give us an overview of how traceability is handled in real life. After the results of the survey are analyzed, the prototype visualizations are created and the controlled experiments that evaluate their performance are carried out.

1.5 Outline

The thesis has the following outline: Chapter 2 will provide an overview of the related work and research that has been done around traceability and Chapter 3 describes the research methodology used in this thesis. The results are described and evaluated in Chapter 4 while Chapter 5 we discuss the results gotten and future work that can be done in the field.

2

Background & Related Work

Before continuing with the rest of the thesis, some concepts need to be explained and we need to give an overview of the state of traceability research. First, software traceability is recognized as an important part of a well-engineered software system and is a required component of the approval and certification process in most safety-critical systems [1, 11]. As mentioned in the previous chapter its goal is to allow the engineer to trace work items across the development lifecycle (both for managerial or regulatory purposes). This is done by linking two or more artifacts in application development, thus showing that there is a dependency or relation between them. Despite its importance, traceability is perhaps one of the most difficult aspects of the software development process. The cost and effort needed to create and maintain trace links in a rapidly evolving system can be high. Its benefits often go unrealized in practice because of badly defined traceability processes, poor user training or a lack of effective tooling [12]. This wide spectrum is noticed by Cleland-Huang et al. [13] where on one end most of the practitioners did not even understand traceability and on the other end of the spectrum resided the practitioners that worked in regulated domains with traceability standards. Researchers have focused on specific areas of traceability developing more sophisticated tools [7, 8], applying information retrieval techniques [6, 9], developing new trace query languages [14] and creating new visualization techniques [15] that use trace links.

A lot of different visualization representations like linear, tabular, sunburst, graph-based, tree-based, [10] etc. are common forms used to illustrate traceability information in different tools, with hyperlinks used for traversing. Traceability visualization is the presentation of trace link information visually by using graphs, lists, images, etc, to help with the understanding of complex information [15, 16, 17]. Here, traceability information includes the links, the set of properties of those links and the related artifacts that they trace. As mentioned in the introduction, at least one visualization technique is almost always used in traceability. But the studies point out that these forms of visualization have a lot of limitations in the information they show if they are used individually [7]. So with a lot of options and none of them being above the rest, the question arises "which of them has the better trade-off between the effort needed to capture complex information and the value of this information?" [11, 18]. To measure this, we should have in mind that traceability visualization involves three main concepts: what to visualize [9], how to visualize, and when to visualize. According to Marcus et al. [19] a traceability tool should contain different views where each best fits a specific task. Maleej et al. [10] showed that whether a visualization is suitable for a particular task is not a straight yes/no answer and that users cannot always choose the most suitable visualization but instead the choices come from user familiarity with specific visualizations and user-perceived usefulness.

Many of these visualization techniques are referenced and used a lot in the literature and in a big variety of tools [19, 15]. Below we are describing some of them and how they achieve their goals.

1) The graph view is the most common technique. Graphs allow the representation of relationships by representing artifacts as nodes and the relations as edges. The edge will be there only if a relationship between the artifacts exists. This allows the visualization of relationships that are multidimensional.
2) Matrix (see Figure 2.1) is the next technique, which is mentioned almost as often as the graph view. Requirements traceability matrix (RTM) shows the relationships between the artifacts as a two dimensional-table. The columns and the rows represent different artifacts, usually requirements and code, while the relation between them is represented as a colored or marked cell. Meaning that if cell a_{ij} is marked or colored, there exists a relation between the artifact of row i and the artifact of column j.

	RE-114	RE-120	RE-681	RE-689	RE-691	RE-693	RE-695	RE-698
AFControlsComponent.java			X					
AFEmergencyComponent.java			X		X	X		
AFFollowBar.java							X	
AFInfoBox.java		X	X		X	X		
AFInfoPanel.java		X	X		X	X		
AFMainLayout.java	X	X	X			X	X	
AFMapComponent.java	X	X		X		X	X	
AFMapViewOperations.java						X		
Configuration.java		X						
FlightInfoCreator.java		X						
FlightManagerService.java					X			
FlightManagerServiceInstance.java					X			
FlightManagerServiceRemoteFacade.java					X			
FlightPlan.java					X			
FlightZoneManager2.java					X			
FRMainLayout.java		X		X				
FRTableDisplay.java		X						
IFlightDirector.java					X			
IFlightManagerRemoteService.java					X			
IFlightManagerServiceInstance.java					X			
IFlightPlan.java					X			
ManagedDrone.java					X			
MapDrawingUtil.java		X						
MapMarkerUtilities.java		X		X				
MyUI.java		X	X			X		
NavigationBar.java			X					
SimpleTakeoffFlightPlan.java					X			
SoloDirector.java					X			
TakeoffAltitudeWindow.java					X			

Figure 2.1: Matrix view example

3) Table (list) is a typical visualization technique that presents a simple list of information, where each traceability link is represented in one entry. The sequential list of links is ranked in order of similarity and contains information about source artifact, target artifact, and other relevant data. The disadvantage of the table is that it does not show an overview of the dataset or the hierarchy of the traceability relations.
4) The tree view (see Figure 2.2) is a graphical representation that displays the artifacts and their relations hierarchically. Each artifact is known as a node or branch and its relationships with other artifacts are shown as sub-elements. Often the tree view is visualized as an indentation list, it is intuitive and allows the elements to be shown without the need to scroll, by expanding a node to show its sub-elements or collapsing it to hide them. The problem is that the tree view grows vertically a lot.

5) Sunburst (see Figure 2.3), is another hierarchical visualization, but instead of indenting the elements it displays them as adjacent rings. Each child of a node (artifact) with

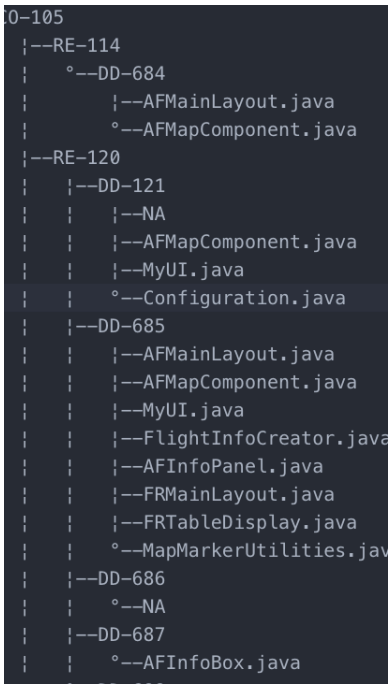


Figure 2.2: Tree view example

depth n is represented in the ring n+1 on the same radian space as its parent, this is the portrayal of the relationships between the artifacts. The nodes are distributed in all directions. Sunburst view is more interactive and performs better on large amounts of information.

6) A not so very common technique is Reingold-Tilford (see Figure 2.4) view. It is a radial representation that combines sunburst with tree view. Unlike sunburst this technique does not use colors in order to show information about the relationships.

The tool that we will use for our prototype is Eclipse Capra[1]. Eclipse Capra [9] is a traceability management tool. It allows the creation of trace links between arbitrary artifacts, provides features to edit them and keep them consistent, and visualizes the relationships between them. Eclipse Capra is a product of the Amalthea4public project, whose purpose was to create a traceability management tool that is embedded in the development environment (Eclipse in this case) and accessible throughout the entire development lifecycle thus easing the work needed to create and manage trace links and enabling other advantages. The tool is highly extensible. The meta-model used for the traceability links can easily be adapted to specific end user's needs. Eclipse Capra's modular architecture allows exchanging the persistence, the visualization, and the management modules easily. New adapters for additional artifacts can easily be added without re-compilation. This allows end-users to customize almost every aspect of the tool if needed. At the same time, sensible defaults that will allow most of the users to use Capra out of the box without extensive configuration are provided. The above reasons are why we selected Eclipse Capra.

As can be seen a lot of research in the field of traceability and traceability visualization has already been done in a wide extent. A lot of different tools and visualization techniques have been proposed and developed. However very little research was done in comparing

[1]https://projects.eclipse.org/projects/modeling.capra

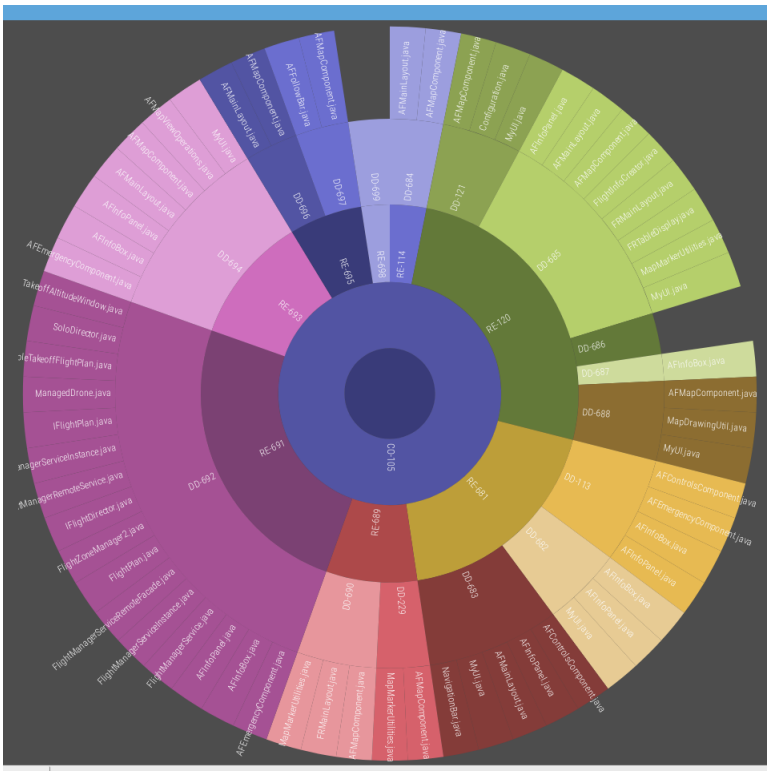


Figure 2.3: Sunburst example

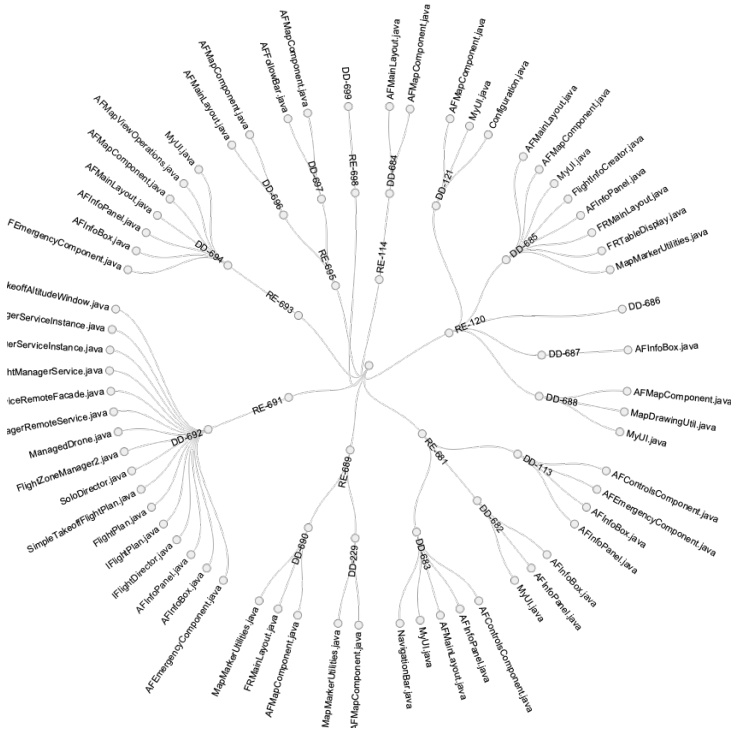


Figure 2.4: Reingold-Tilford example

and evaluating them against each other. The main differences of our paper to the others mentioned is that we search for a better visual way to interactively create, maintain, analyze and use traceability information by testing real life scenarios, comparing and evaluating their results and getting the feedback of two different groups on the proposed visualizations. Contrasting the reaction of two particular groups (industry vs academic) is the key difference of our paper from the others, where the sample group is mixed. Also in comparison to Li and Maalej [10] our thesis has two iterations a questionnaire with field engineers and an experiment with academic people. Furthermore, another difference is that the visualizations used in the experiment are not selected by us but by the answers given to the questionnaire, giving a more unbiased selection. The bigger size of the dummy projects we use for the experiment and questionnaire, together with the paper having a more general focus rather than a task oriented one are also distinctions worth mentioned.

3
Research Method

In the introduction section, we specified some of the different techniques we used in order to answer the questions we proposed. These techniques are part of the design science research method that we use.

3.1 Research Questions

Firstly we start by defining the research questions we are going to answer:

RQ1: What graphical information is useful for an analyst when using a traceability tool?

The purpose of this question is mainly to understand the domain of our research. Also, another thing that this question deals with is what the current state of traceability visualizations and what are the techniques that analyst and engineers use in their day to day projects. This question, as mentioned in the second chapter, is based on one of the concepts of traceability visualization which is: what should be visualized. The data gathered for this question are mostly quantitative.

RQ2: What impact do different visualizations have on work efficiency?

Just like the first research question this one is also based on one of the concepts of traceability visualization, in this case being the concept of how to visualize. This is the main question of our thesis, it helps us understand and evaluate the solutions we propose. This is done by observing the effects that the selected visualizations that are implemented to Eclipse Capra have to the different tasks done by engineers or analysts. For this question the data gathered are qualitative and quantitative.

3.2 Methodology

As mentioned in the beginning, we used design science research method in this paper. In design science, we iterate over two activities: designing an artifact that improves something for the stakeholders, in this case traceability visulizations and empirically investigating the performance of an artifact in a context[20, 21]. In other words the purpose of this method is to investigate a problem while implementing a suitable solution to solve the problem and evaluating the effectiveness of the solution. Looking at the steps of our research methodology structure we can clearly see the design science stages in it. The survey is used to investigate the problem, the prototype implementation can be seen as the designing of the artifact and lastly the experiment is the evaluation of the prototype. The data collection methods used, as mentioned, are a survey of engineers and an experiment of the developed prototype with students. These collection methods are also the main parts of each of the iterations we conducted. During the research done in understanding the area of traceability and finding related works that deal with visualization of traces, we found a lot of different tools and techniques that researchers proposed and developed. This helped

in devising the majority of the questions of the survey and gave us a better idea of the state of the traceability research. The sample group that the survey was conducted on was selected from a pool of field engineers provided to us by our advisor.

3.2.1 Analysis techniques

The data collected by the survey and the experiment are both quantitative and qualitative. In order to analyse the results and come to a conclusion we used a one-way ANOVA on the quantitative data collected. The idea is that we want to find whether any of the groups created by the levels of the independent variable are statistically different. Also, another reason that we decided on one-way ANOVA is that it uses the F-test for statistical significance which allows for the comparison of multiple means at once.

While for the qualitative data thematic analysis was used. The main tool was deductive approach where the reflected data were analyzed with some preconceived themes based on the theory or existing knowledge. Also, the semantic approach was used to analyze the explicit content of the data. There were five steps to do thematic analysis, which were familiarization of the data, coding the data, generate themes, review the themes, and define the themes. The qualitative responses were transcribed, then categorized to themes so that they could provide us with meaningful results. The themes were then analyzed so that an answer to the research question could be produced.

3.2.2 Datasets

An important part of the methodology is the dataset on which the study will be conducted. In our case we decided to use two different datasets for each iteration of our design science. Both of the datasets selected were part of group of projects provided to us by our supervisor.

For the survey iteration we used the Dronology[1] dataset. This project is created at the University of Notre Dame within the Department of Computer Science and Engineering and was launched in January 2017. It is an open-source Unmanned Aerial System (UAS), that focuses on coordinating UAS in tasks where they either work collaboratively to achieve goals or perform independent tasks in the same airspace. We decided to use this one because the dataset consists of a big amount of artifacts like components, requirements, design definitions, tasks, and source codes. All of them were connected with one another either explicitly or by ID tags creating a project with more than 500 relations. Also the format of all this information was a json file, which was easier to use in order to create the example visualization images for the survey.

As for the experiment, we decided to go with another project called Car Headlight. Car Headlight is simple example of product line for car headlights and contains feature models, requirements, design models and code files. We did not use the Dronology project for this iteration because it was available only as a json file which limited our ability to change and adapt the project to our needs. Plus this format can not be properly imported to Eclipse Capra. Also another reason we used Car Headlight is because it presented us with a good amount and variety of artifacts like requirements, code files, UML models etc., and it gave us the freedom to create and structure the trace links in our own way. Furthermore, it provided us with a way to add some more artifacts that we needed like test files, more requirements, a few UML models, etc. This allowed us to create a notably big project with 85 unique artifacts and with around 100 trace links connecting them.

[1] https://dronology.info/datasets/

3.3 Survey

The first technique used, which helped us understand the problem, was a survey. The survey was conducted through a questionnaire and was answered by engineers familiar with traceability. These engineers constituted our sample group, whose size was 23 people. Their contact information was provided to us by our supervisor.

The information we wanted to get by using this questionnaire is what techniques or tools engineers and analysts use in their projects. Also, we needed to know what benefits the respondents get from traceability, what problems they encounter and if visualization has or can have a part in this whole process.

Additionally, in the questionnaire, there were included questions that provided some suggested improvements and some comparisons between different visualization techniques. These questions were motivated by the papers we read during the related work research. The aim of these questions was to see how much of the research done was adopted and used in real-life projects. By answering these questions we hoped we could get a better understanding as to which solutions, provided by the research done in the field, work.

The questionnaire was divided into four sections with 33 questions. They were a combination of multiple choice and open questions. More concretely, the questionnaire sections and structure were:
1. General questions, whose aim was to help us get an understanding of the background and the relation that the respondent has with traceability.
2. The second section had questions that were geared specifically towards traceability. The purpose of this section was to understand what techniques related to traceability each respondent used in his work and how traceability is approached by the companies where the respondents work. This was achieved by using open questions combined with Likert scale based questions asking them to rate different statements.
3. The next section was composed of questions that were inspired by previous research papers and required the respondents to rank and score different trace link visualizations, according to their perceived usefulness and understandability. The idea is that the selected visualizations can be considered improvements that will be implemented in the tool. The examples were comprised of a variety of diverse trace visualizations of the project called Dronology. Since the purpose of this section was to rank visualizations, we mostly used the Likert scale.
4. The last section can also be considered as an extension of the third one since its aim was to understand what kind of navigation the engineers and analysts would prefer to be combined with the proposed visualizations.

After the survey, the first iteration of our design science was creating prototype visualizations according to the answers of the survey. The second step, after the prototype creation was conducting an experiment to evaluate the effectiveness of our prototype.

3.4 Experiment & followup

As mentioned the second iteration of our design science method was the experiment, which was conducted with students. We made this decision because we wanted to see how people with little working experience would manage and use the visualizations. And as mentioned before to contrast this with the responses that we received from the sample group of the survey. After all the information gathered from the survey was analyzed, a selection of the visualizations that would be implemented was done and the prototypes were created. As mentioned in the chapters before, the visualizations were implemented in Eclipse Capra,

which also was the environment that was used for the experiment. The experiment was followed up by a small survey in order to get a more in depth qualitative feedback from participants.

The experiment was structured to be a contrast between the different visualizations that were implemented so that we could get an understanding of the positives and drawbacks of each of them and see how usable they are. Firstly the participant got a small introduction about our thesis and about traceability. After that an overview of Eclipse Capra was given in order to make the participant familiar with the experiment environment. The participant received three different tasks, which were selected according to the answers given in the survey that was conducted in the first iteration. Those tasks tried to simulate real-life tasks done in different areas of software development. The participants were instructed to try and complete them by using the visualization assigned to them. The assignment of which visualization each participant would use was done randomly and also the experiment was performed individually by each participant with only him and us present at the specific time. During the experiment we measured the number of clicks and the time it took for the participants to complete all the tasks and we combined them with the number of the correct answers given. The number of clicks was measured using Click Counter[2] in the MacOS computer and Mouse Clickr[3] for the Windows computer, while the time was measured using a standard timer. Also as a back up we screen recorded all of the participants interactions. These three are the metrics we collected directly from the experiment.

The second part of the experiment consisted of a small survey, which was completed immediately after the end of the experiment. The survey consisted of two sections. The first one had general questions about the participant and some small evaluation questions. Its purpose was the same as in the main questionnaire, to gather information about the participants background, what relation they had with traceability, what they thought about the experiment and mainly get their evaluation about the visualizations. The second section consisted of questions taken from the SUS (System Usability Scale) tool. SUS is a reliable tool for measuring usability and it consists of 10 questions with five response options; from Strongly agree to strongly disagree. It can be used to evaluate a wide variety of of products and services. We selected the SUS questionnaire because it is very easy to administer to participants, can be used on small sample sizes with reliable results and most importantly it can effectively differentiate between usable and unusable systems. This helped us in quantifying the opinions of the participants about the visualization that they used. The goal was to combine the SUS score and the direct measured metrics in order to get a more definite answer to our research questions.

[2]http://www.murgaa.com/mac-mouse-click-counter/
[3]https://www.softpedia.com/get/System/System-Info/Mouse-Clickr.shtml

4

Results & Evaluation

In this chapter, we present the results of each of the data gathering techniques together with the main results of our paper and try to evaluate and understand them.

4.1 Survey Results

The questionnaire was created in Google Forms and was sent to the participants through email. From the 23 people that the questionnaire was sent only 11 answered, meaning that we got a response rate of around 48%. Something that is worth mentioning about the participants, by looking at the data, is that we have a good variety of software engineering related job positions with different years of experiences all around (see Figure 4.1). Seeing that everyone has over five years of experience in software engineering it gives us confidence in their answers.

Solution Architect
Software Architect
Technical leader
Senior System Architect
CTO
System / Software Architect
System Test Engineer
Principal Scientist
System Architect
Consultant
Assistant Professor

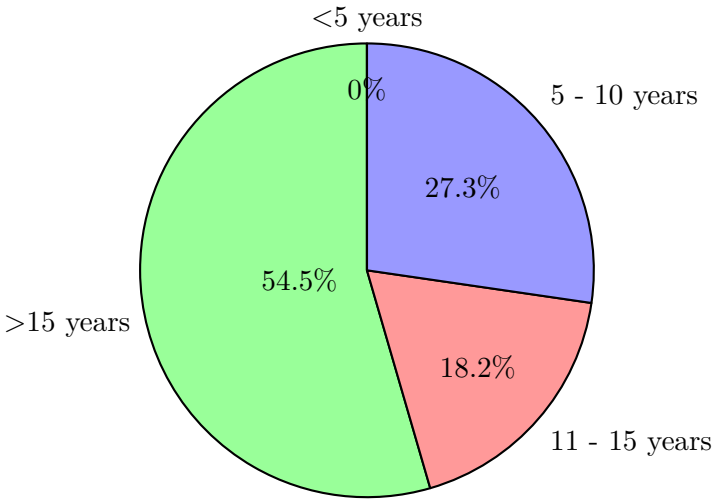


Figure 4.1: Participants' work position and their work experience

The first thing that becomes clear by looking at the results of the questions “How important is traceability in your company?” and “In which area of software engineering have you used traceability?” (see Figure 4.2, Figure 4.3), is that traceability is an important part of software development in modern companies since 10 of the respondents said that it was important (4) or very important (5). Also, any improvement or enhancement that can be done to traceability can have a big impact on the whole company and the way it works. This is supported by the big extent of activities that traceability is used in, as it is seen in the result table (Figure 4.3), where the most popular are requirement validation and change impact analysis.

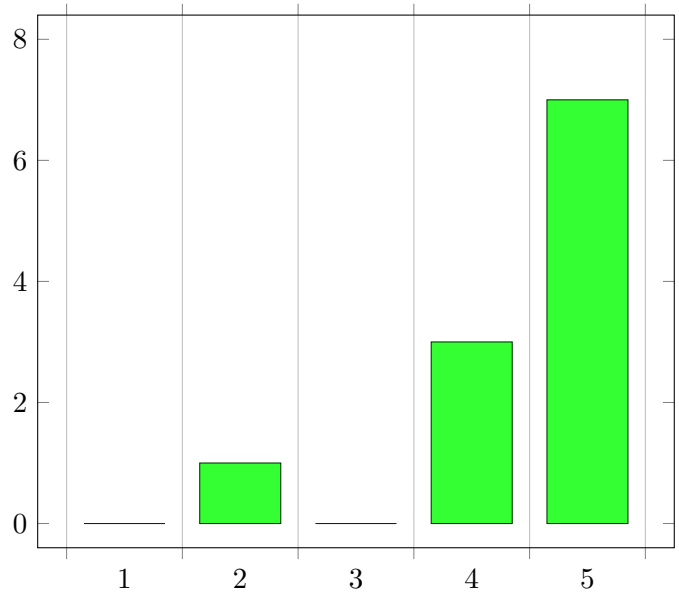


Figure 4.2: The importance of traceability (1-not important, 5-very important)

Seeing as traceability is an important activity in software engineering the next information needed is, how are traces kept and managed by the participants. The answers to this question as can be seen in the data gathered (see Table 4.1), tell us that all the respondents use some kind of tool to keep track of their projects’ trace links. These tools can range from IBM’s DOORS, which was the most mentioned, to in house created ones. But almost in every answer, there was more than one tool. This means that no one tool is the best, but a combination of tools is needed for the engineers to do their tasks and achieve their goals.

Having seen and analyzed the information we had gotten so far, we wanted to know what the participants thought about visualizing traces and what type of different visualization techniques they had encountered before. So firstly we looked at the answers of the question “How helpful is visualization of traceability links for you?” (see Figure 4.4) and as expected visualization of traces is considered a very helpful and influential aid in creating and managing traces since five answered very important (5) and three said important (4).

Figure 4.5 shows the answers to the question, which visualization technique the respondents are more familiar with. As seen, the traceability matrix and the table view are

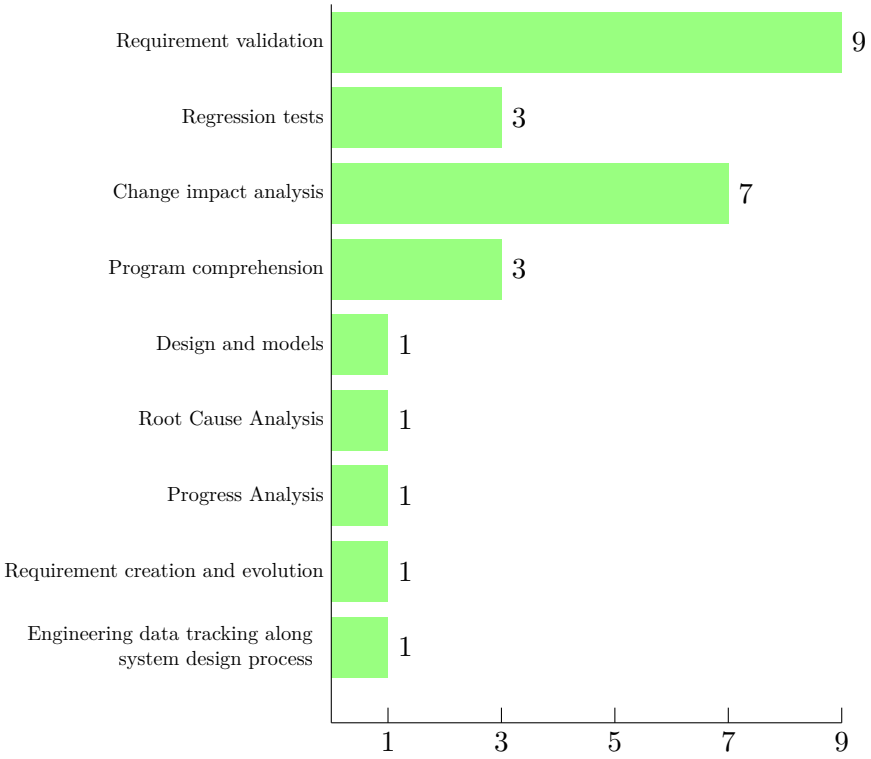


Figure 4.3: The areas of development that the respondents use traceability

DOORS (5 answers)
Polarion (1 answer)
IBM ALM Tool Suite (1 answer)
Atlassian tools (1 answer)
Git (1 answer)
MS Excel (1 answer)
Internal tools (2 answers)
Jira (1 answer)
ReqM2 (1 answer)
TomSawyer Perspectives (1 answer)

Table 4.1: The tools used for trace keeping and managing

the most selected ones. From these results, the information we got while reading related papers is reinforced, in that the traceability matrix and graph view are some of the most popular visualization techniques. But the thing that stands out more is that even though in the research papers we read sunburst is mentioned and used almost as much as the rest of the techniques, it is one of the least selected by the respondents.

The results were also strengthened even more by the respondents' answers to the third section of the questionnaire. Where, as mentioned, before different visualization examples of the same project were compared to one another. Before analyzing the results we need to clarify that in this part of the questionnaire we had some problems since one participant could not see two of the visualizations, sunburst and tree view, so in those cases instead of 11, we have 10 answers.

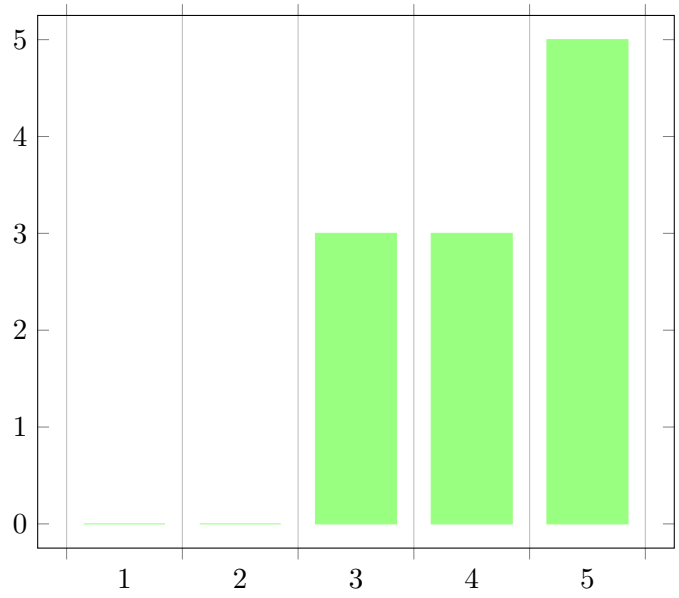


Figure 4.4: The importance of visualizing trace links (1-not important, 5-very important)

	Sunburst	Traceability Matrix	Graph View	Tree View	Weighted Graph	Reingold-Tilford
Understandability	2.9 ± 1.22	4 ± 0.95	3.5 ± 0.66	3.3 ± 0.64	2.5 ± 0.89	2.5 ± 0.99
Usefulness	2.7 ± 1	3.6 ± 1.15	3.3 ± 0.75	3 ± 0.89	2.5 ± 0.99	2.4 ± 1.07

Table 4.2: The average score and standard deviation of each visualization technique

The values for each visualization shown in the table above (see Table 4.2) represent the average score the visualization got from the respondents, according to the participants perceived understandability and usefulness of each example visualization. The second value represents the standard deviation of the scores. Looking just at the averages our before-mentioned results seem to uphold. Traceability matrix, graph, and tree view are the most voted, meaning people feel more positive towards them even though all of them were cluttered with links and the information was the same. But looking only at the averages does not give us a complete picture; therefore, we should take into consideration the standard distribution of the votes.

The first thing that becomes clear, by evaluating the averages together with the standard deviation, is that the participants have a stronger positive opinion for graph and tree view. This is because their standard deviation is the lowest and they are ranked the highest by average votes. The second thing we deduce is that the respondents have a strong negative opinion for the weighted graph and Reingold-Tilford, the opposite to the graph and tree view. The most interesting thing that comes out of these numbers is that even though traceability matrix has the highest average scores for both understandability and usefulness, its high standard distribution tells us that the respondents are split in their scores. The same arguments can be used for sunburst; even though it has low average scores, it has a high standard distribution. This analysis helped us to decide which of the techniques we should implement in our prototype. Our thought process was to select graph and tree view, as they were the ones that the participants felt most strongly positive about, and together with them implement also traceability matrix (RTM) and sunburst as the visualizations that the participants had more mixed opinions about. Also we considered sunburst as the novelty that we wanted introduce in the second iteration of our design

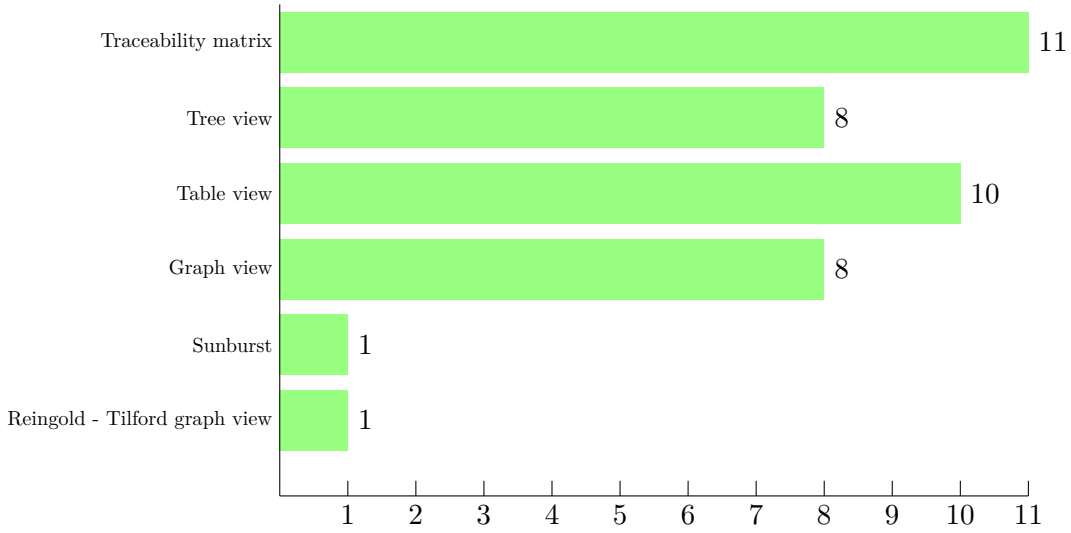


Figure 4.5: The visualization techniques most familiar to respondents

science methodology.

	Description	Meta-data	Expand the item	More navigation options
Average Score	4.36 ± 0.77	3.36 ± 1.37	2 ± 0.85	2.18 ± 1.11

Table 4.3: Average score and standard deviation of the hover navigation options

	Zoom in/Zoom out	Go to the artifact	Expand/Collapse links	More navigation options
Average Score	2.82 ± 1.03	4 ± 1.21	3.45 ± 1.16	3.73 ± 0.96

Table 4.4: Average score and standard deviation of the click navigation options

The main idea of implementing those visualizations in the prototype is that we want to understand, why there is a disparity between the visualizations engineers prefer and use in real life traceability tasks and the visualizations that some papers we read about traceability refer and use, mainly sunburst. Also we want to figure out and test which of these visualizations are better at handling projects with a high number of trace links. These visualizations will be paired with some basic navigation according to the preferences of the respondents. The results of the questions "When I hover over an item, I want" (see Figure 4.3) and "When I click an item, I want" (see Figure 4.4) are shown in the images below.

According to the participants' responses, we decided to show description and some meta-data when a user hovers above an item since they are the most selected. While for clicking we will implement go-to functionality in the graph view, RTM (traceability matrix) and tree view. As for sunburst for clicking we will implement zoom in/zoom out. Also for the tree view the expand/collapse navigation option will be implemented.

Last, but not least we present the raw results (see Table 4.5 & Table 4.6) to the questions of what information do the participants think is important to be shown in a traceability visualization and what type of problems they have faced, regarding traceability, during their working experience. The evaluation, categorization and the reasoning of

this data will be done to the next chapter, since it is tied more to the research questions posed in the beginning.

I think search/filtering is the key - understanding a large graph is difficult otherwise
It is usually not very important to visualize the traceability in the complete system in one diagram. It is more important that you have tools to find loose ends and missing
trace matrices are not very helpfull to trace source code and implementation artefacts
Graphical complexity management, level of detail, usability of layout algorithms, filtering in conjunction with hierarchical breakdowns, scalability of visualization approaches (some might work with smaller datasets, but don't scale well visually with bigger datasets), responsiveness and performance
to much elements to display
Having a clear overview for large scale
The visualizations often times are cluttered with information I do not need. I should be able to prune and select the artifacts I want to see in my view.
Depending on the amount of data, views may get very confusing.
Complexity of graphs, cycles in trees, readability in tables
scalability, when the number of trace links grow visualization quality is negatively affected

Table 4.5: Problems encountered while using trace link visualizations

Would like to see the related elements, normally a modelled element and referenced requirement.
For a given element: all other linked elements and their types (requirement, architecture component, source code, test case etc.)
Perhaps it could be useful to find illegal connections. However, I believe this could be found automatically by the tools instead of humans looking at diagrams
main relationships between specific elements under analysis
Comprehensive high-level overviews combined with drill-in capabilities for detail views without getting lost in too much "clutter" (or irrelevant information - which is hard to define based on different usecases/required perspectives). Would like to learn more about possibilities "beyond boxes and lines", e.g. Sunburst, etc.
overview, identification of missing links
Hierarchical relationships and Missing Traces
Missing links
e.g. coverage
Responsibility, dependencies, cycles
missing trec links, links to the actual artifacts (mapping to the underlying classes, requirement specification so on), changed trace links across versions

Table 4.6: The type of information the respondents want from a visualization

4.2 Prototype creation & Experiment set-up

As mentioned in the previous section, the four visualizations that were selected to be implemented were the graph view, the tree view, the RTM (traceability matrix) and the sunburst, together with the specified navigation options for each one as voted by the participants.

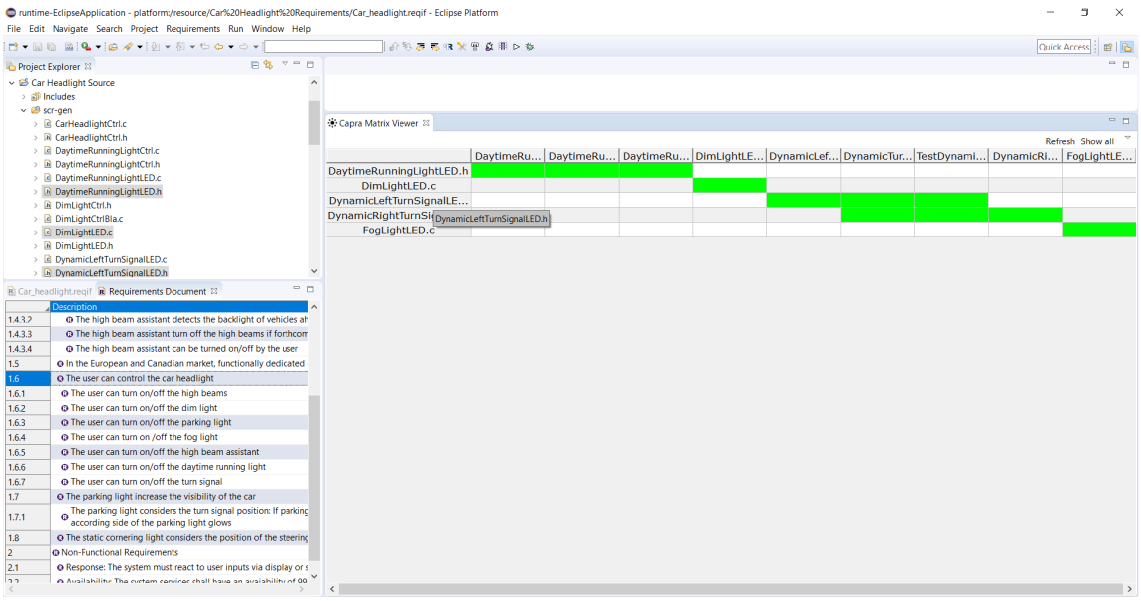


Figure 4.6: Eclipse Capra with matrix view

Also following the results of the survey, the tasks that the participants had to simulate were change impact analysis, program comprehension and test coverage (as part of regression tests). These three tasks were the most voted ones in the questionnaire (Figure 4.3), after requirement validation which was the number one task. Even though requirement validation is the most voted one we opted not to select it due to time constraints, because we wanted to keep the experiment at around 30 minutes time limit due to restricted student time. Implementing a requirement validation task would have meant passing that limit since the participant would have had to spend a considerable amount of time reviewing and understanding each requirement and their correctness in regard with the business goals and not using the visualization a lot. This would have given us little information about the visualization of the traces and would mostly gone away from our scope.
Specifically the tasks were:
1) First was the change impact analysis task. For this task the participants received the name of one requirement artifact and one code file that presumably were changed. The task given to the participants was to use the visualization assigned to them and try to find the code files connected to the given artifacts that will be affected by this change.
2) The second task had to do with program comprehension, in this one the participants got the list of all the requirements and their goal was to find and mark which of these requirements were not implemented. Meaning they had to use the visualization to get an overview of the project and its traces in order to help them view and understand the requirements and their connections.
3) Lastly the third task as mentioned before is about test coverage. In this task the participants received again a list of all the requirements but this time their task was to find out and mark which of these requirements were not test covered or partially test covered.

The idea is that a requirement is indirectly connected with a test code file, meaning that in order for the participants to find if the connection existed they had to traverse the trace links and navigate into more depth than in the first task. Consequently this exhaustive dive would make the visualization complex and clump a lot of traces in a limited space, letting us check and test how well the visualization will handle the clustering.

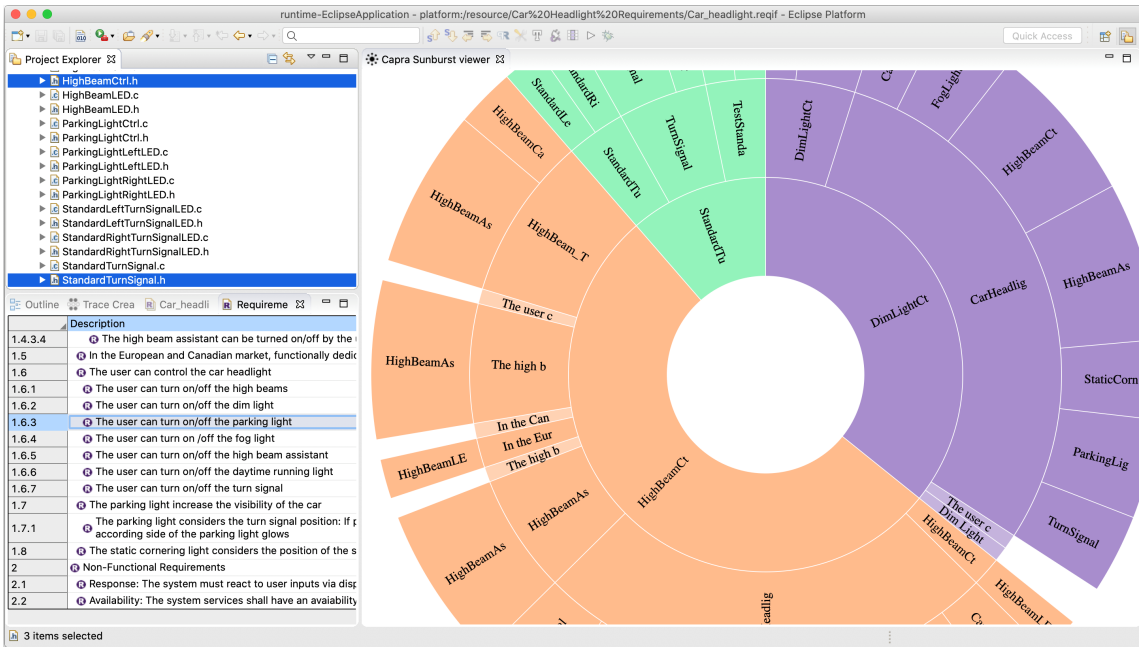


Figure 4.7: Eclipse Capra with sunburst view

4.3 Experiment results

Having clarified and reasoned about our specific task selection, the specific task structure and how our prototype program looks; we can start looking at the results of the experiment.

First thing that pops up from looking at our post-experiment survey data is that the work experience of the experiment participants in software engineering is small compared to experience of the survey participants. This is not a surprise since as mentioned in the previous chapter we decided beforehand that the sample group should be comprised only of students. The total number of the participants in the experiment is 16, which means that each implemented visualization was tested by 4 different people. As can be seen in the figure 5.2 below, the majority of the students were studying Software Engineering (9), while the rest of them were studying other programs like Computer Systems and Networks or were doing their PhD. The answer to the question "Do you have any work experience related to software engineering" showed us that 13 of them answered yes to this question, where only 2 of them had relatively long working experience (more than 5 years). The next important survey result worth mentioning it has to do with the familiarity and the knowledge that each participant had with traceability. By looking at the data we can see that only 11 of the participants had worked or had some extensive knowledge about traceability. This makes sense and ties in with the results from the first question about the study program that each participant attends, since Software Engineering students have a better grasp of the technique due to their degree.

The main and most important questions of the post experiment survey, had to do with the score that the participants gave to the traceability visualization that they used.

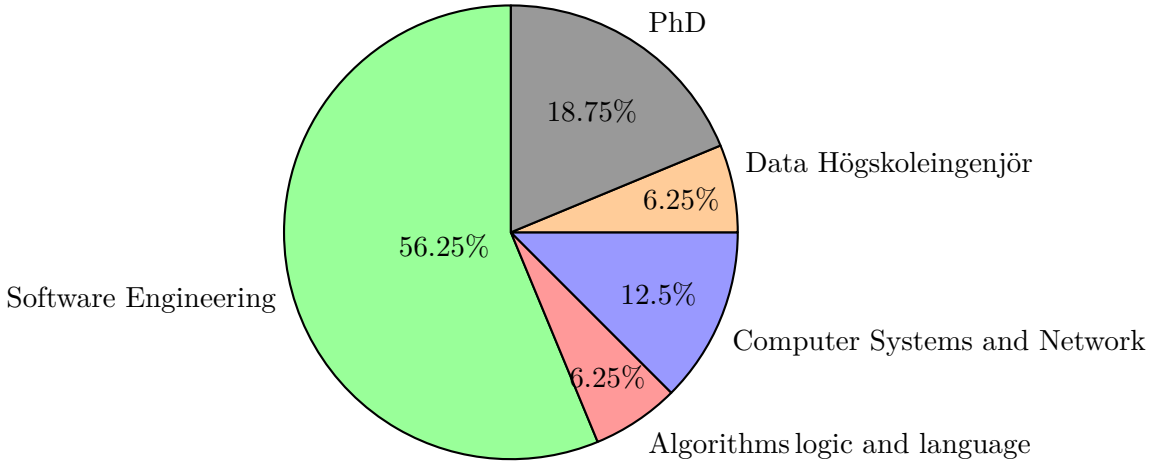


Figure 4.8: The study programs of the experiment participants

In order to be consistent we asked them to rate the visualization's usefulness and understandability, the same as in the survey. The average score of all of the responses to the these questions are shown in Table 4.7 below. The main thing that we notice is that average score given by the participants about the usefulness of the visualizations, results in sunburst being valued as having the same usefulness as matrix and even better than the graph view. The same pattern can be seen even in the rating of the visualisation's understandability, where sunburst again is valued as the best, but this time graph view is ranked higher than matrix view. Another thing that is constant in both the metrics is that the tree view is the least scored one. This is in contrast with the results that we got from the survey where sunburst was not valued so high compared to matrix or graph view.

	Graph	Sunburst	Matrix	Tree
Understandability	4.0	4.0	3.5	3.75
Usefulness	4.0	4.5	4.5	3.5
SUS	82.5	80.63	80.63	77.5

Table 4.7: Usefulness, understandability and SUS results

In Table 4.7 we can also see the result of the SUS (System Usability Scale) questionnaire, which is another metric that we use in combination with the other ones. As explained before we used this scale in order to measure the ease of use of the visualizations. The results of SUS shows us that as far as usability is concerned the best one is graph view, with sunburst and matrix having the same score and tree view last again. Trying to understand this table and the results in it, we can say that there is a clear division between the tree view and the other visualizations. But the same can not be said about the other three visualizations. According to the participants scores they are quite similar, one visualization is better in one regard while another is better in an other regard. This means that we have to look at the other metrics collected during the experiment in order to have a more better idea about the visualizations' effeciveness.

The other metrics were collected directly from the experiment, rather than using a questionnaire. The first one, whose results are shown at the table 4.8, is the time it took for the participant to finish all the experiment tasks using the given traceability visualization. If we analyze the data gathered we see that the individual times for each visualization

Usage Time	Sunburst	Matrix	Graph	Tree
	17:20	22:20	16:20	25:25
	15:00	14:15	20:35	19:55
	15:10	15:40	30:50	21:34
	13:19	16:33	21:29	18:56
Avg(mm:ss)	15:12	17:12	22:19	21:28

Table 4.8: All times for completing all tasks in minutes and seconds.

technique are very spread out and there is not a clear pattern. In every visualization there are people that finished the tasks quickly, but there are also participants that took quite some time for their tasks. By averaging out the times for each visualization technique we get a better insight about the visualizations which helps us to better compare them. By this it becomes evident that sunburst and matrix are better, in this regard, than graph and tree view. Also the times hover around the 20 min limit that we projected the experiment to take due to participant time limitations.

The second metric we measured during the experiment was the number of times a mouse button was pressed, the results of which are shown in the Table 4.9. Again in order to get a better picture about all the visualizations and their performance according to this metric, we are using the average of all the results. By considering these data we notice that sunburst has performed quite better than the other visualization techniques. While matrix is the last one in line with the results.

Number of Clicks	Sunburst	Matrix	Graph	Tree
	74	135	128	120
	138	127	97	147
	71	145	110	88
	79	117	95	81
Avg	90.5	131	107.5	109

Table 4.9: Number of clicks to complete all tasks.

Correct answers	Sunburst	Matrix	Graph	Tree
	48/48	48/48	48/48	48/48
	46/48	48/48	48/48	48/48
	48/48	48/48	43/48	48/48
	48/48	47/48	44/48	46/48
Avg	47.5	47.75	45.75	47.5

Table 4.10: Number of correct answers given.

The last directly measured metric had to do with the amount of correct answers given by the participants to the tasks they were asked to perform using the specified visualization technique, the results of this can be seen at table 4.10. The results show that as far as the correctness metric is concerned, all of them performed quite similarly. All the participants answered all the questions correctly with a few exceptions. This may have been a result of the type of tasks the participants were asked to perform. The tasks were designed to resemble real life but due to the time constraint, as specified before several times, they

lacked in regard to complexity and this may have influenced the results of this metric. In the overall view, the information that we get from Table 4.10 does not help us in better understanding the performance of the visualization techniques so we will not examine and use it further. In the next chapter, all the data gathered in these iterations will help us in answering the research questions proposed.

5
Discussion

Firstly in this chapter we will discuss the results that we got from both the survey and the experiment in regard to the research questions mentioned in Chapter 3. Furthermore we will look at our threats of validity and how those influence our results and what we did to limit their effect.

5.1 Research questions

The results of all the iterations of our design science research method are presented and analyzed in the previous chapter. That data directly or indirectly answer the research questions we proposed.

RQ1: What graphical information is useful for an analyst when using a traceability tool?

The first design science iteration (survey) is the one that, among other information also provides the answer to the question. As part of the survey there was a question that asked the participants to indicate what information they want the visualization they use to show. The results of this question are shown in Table 4.6. As can be seen these data are qualitative, therefor, as mentioned in the methodology section, we used thematic analysis to interpret them. The thematic analysis helps us to categorize the answers into themes that could be defined as meaningful results. The thematic analysis on the qualitative data yielded four main themes as shown in Table 5.1. The main thing that pops out by looking at the categorized answers is that the main request is for the visualization to be able to show an overview of the whole artifacts and their links, while the second most requested feature is to be able to detect or display missing links between the artifacts.

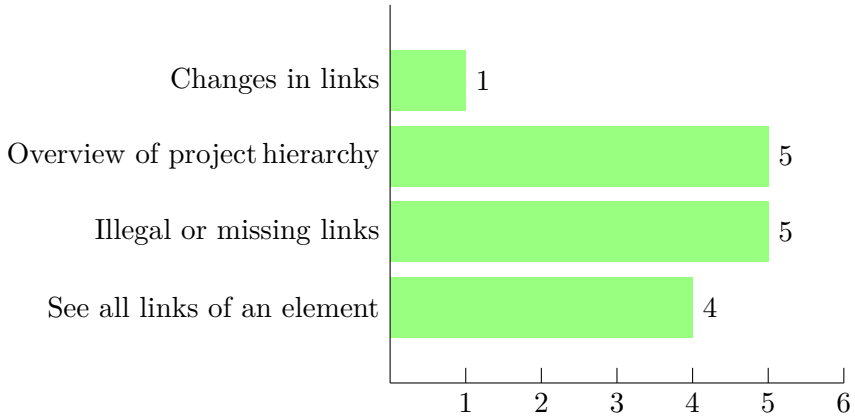


Figure 5.1: The summary of the results about the information that should be shown in the visualization

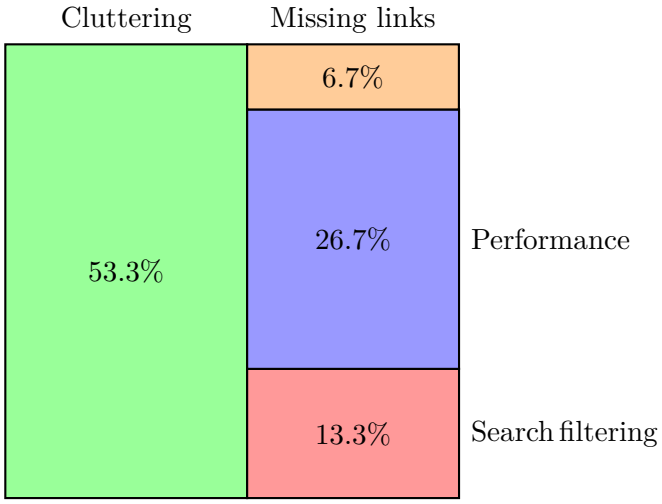


Figure 5.2: The summary of the results about the problems encountered while using traceability visualizations

These responses are also backed up by the scores that the survey participants gave to the example visualizations presented to them, where according to the results they valued much higher those visualizations that gave them a better overview of the hierarchical relationships between the artifacts. Meaning that this a must feature for every traceability visualization technique in order for it to be used and help the engineers in their tasks. But there was a discrepancy in the results, where even though some of the visualizations fulfilled the requirements proposed by the participants, their scores did not reflect this. Indirectly the survey participants also emphasized another trait that the visualization should have. Looking at the results of the question that had to do with the problems that they encounter by using traceability visualizations table 4.5, cluttering is the most mentioned one. There were different reasons given for cluttered visualizations like scaling of the project or higher complexity selection. These two characteristics, project overview and uncluttered visualization, seem a bit contradictory towards each other so having both of them in a single visualization has to do with finding the right size of the hierarchical overview so that the visualization does not become overwhelming and incomprehensible by the high amount of elements in it.

RQ2: What impact do different visualizations have on work efficiency?

As for the second research question, the experiment is the one that produces the metrics that help us give an answer to it. The first thing we needed to do is to decide how we were going to measure work efficiency. Knowing the structure of the experiment and what is considered efficient in a work environment, we decided that the metrics that we were going to measure were the time, number of clicks and the number of right answers for each participant (see tables 4.8, 4.9, 4.10).

If we do an examination of both time and number of clicks together there are a number of things that need to be pointed out. The first thing that becomes evident by looking at the collected data is that sunburst is always at the top in both the metrics. While on the other hand tree view is always in the bottom part of the metrics. Lastly there is a disparity with matrix view. Even though the average usage time of the view is shorter than both tree and graph visualizations, the number of clicks counted is the highest among all the visualizations. This can be interpreted as matrix view being more straightforward to use but needing a lot more effort than the others.

	Sunburst	Matrix	Graph	Tree
Avg time	912.25	1032	1338.5	1287.5
Avg number of clicks	90.5	131	107.5	109
Avg number of correct answers	47.5	47.75	45.75	47.5

Table 5.1: Summary of the metrics for all of the tasks

As stated in the previous chapter just looking at the average values for each metric we notice some differences between the values that point us towards a specific ranking of the visualizations. But something that can be important to consider is that the individual measurements for each metric and visualization have quite diverse values, for example one participant using matrix view took more time to complete the tasks than another participant using tree view, etc. So in order to be certain about the effects that each of the visualizations have to the metrics we performed a statistical analysis (one way ANOVA) on them.

	df	Sum of Squares	Mean Square	F	p-value
Between groups	3	498665.19	166221.73	3.04	0.0707
Within groups	12	656752.75	54729.4		
Total	15	1155417.94	77027.86		

Table 5.2: ANOVA analysis of time

	df	Sum of Squares	Mean Square	F	p-value
Between groups	3	3310	1103.33	1.91	0.1826
Within groups	12	6947	579		
Total	15	10258	683.87		

Table 5.3: ANOVA analysis of number of clicks

	df	Sum of Squares	Mean Square	F	p-value
Between groups	3	10.25	3.42	1.49	0.2669
Within groups	12	27.5	2.29		
Total	15	37.75	2.52		

Table 5.4: ANOVA analysis of correct answers

The average number of correct answers is the one metric that even with a naked eye seems to get the least affected and after we perform the analysis we are sure that the difference between the averages is not statistically significant. Analysing the next metric which is the average number of clicks we also get a not statistically significant difference in their values. The same result was obtained even when we statistically analyzed the average time for each visualization, though in the case of time we see that the p-value is very close to alpha value of 0.05. These results are as interesting as they can be considering our small sample size since they clash with the perception that the participants had about each visualization and the way it affected their task fulfillment which is reflected in the

scores they gave for each visualization's usefulness, understandability and SUS metric. Although it should be emphasized that these results can change if a bigger sample size is used for this experiment.

As a result the answer to the research question is that the different traceability visualizations have no statistically significant effect on work efficiency. In this case the difference in the values could be due to the unfamiliarity with the Eclipse Capra tool, or even due to the difference in knowledge about traceability in general and how the experimented tasks should be approached.

5.2 Threats to validity

As in any other research, even in our we had some threats and delimitations. We divided the threats mainly to external, internal and other as discussed below.

Limitations: The main thing that should be mentioned is that the the sample sizes for both of the iterations of our design science study were not as big as we wanted to. Firstly our survey had a a very low response rate, where we expected all of the 23 engineers to answer but this was not the case. However since it was a survey we prepared for this by having a lot of questions in it thus providing us we quite a proper amount of data. As for the experiment sample size, our first plans were to have at least 28 participants, seven for each of the visualizations, thus giving us a better chance to have a meaningful statistical analysis. But due to the whole research taking more time than it should we had to stop waiting for more participants to take part in the experiment, so our sample size ended up being 16 people. This small number of participants both in the first iteration (survey) and the second iteration (experiment) are not significant enough to allow us to make generalized claims for the whole population.

If we knew that the participation rate in the experiment would have been so low there were a couple of things that we could have done to compensate for it. First off we could have had interviews with the participants after the experiment thus providing us with more information and data. The other thing we could have done would have been to change the structure of the experiment in a way that it would have also provided us better results and data.

Threats to internal validity: During the selection of the visualizations which will be implemented there can be a selection bias. Therefore together with the small literature inspection there will be a survey done to a small sample group. This way the list of visualizations will be general and impartial and have a reduced influence by our own ideas. Since a survey will be conducted the risk of maturation is possible. In order to avoid this the survey group selected will be different from the ones that will participate in the experimental phase. Likewise another threat to the validity is the experiment introduction phase for each participant. Due to the experiment being conducted individually by each member there is a possibility that the introduction, even though it was pre-planned, might have been delivered differently to each of them. The use of two different machines, one running MacOS and one running Windows, for the experiments can be considered as internal threat because the functional differences between the systems might have affected the results. Furthermore another important delimitation that influenced our experiment structure a lot was time. As mentioned before since the participants of the experiment were students we had to consider the amount of time they were willing to dedicate to our experiment and plan around it.

Threats to external validity: Using mostly students can affect the generezability of the results, since they might not have much "real" world experience. To control for this

the post experiment questionnaire included questions about their traceability experience, background and coding experience so that this information can be taken into account. Also the number of participants both in the first iteration (survey) and the second iteration (experiment) are not significant enough to make generalized claims for the whole population.

Other threats: An important validity is the construct one and in order to address this one we decided to take some measures. Firstly we tried to address the mono-method bias, which happens when only one type of method is used for measurement. In order to reduce this threat we used both a survey getting more qualitative data and an experiment which provided us with quantitative information. Moreover the experiment was also accompanied by a usability test and a post-test survey. This allowed us to be able to check the results of the methods.

Another important construct validity that we tried to reduce its effect is the learning and experience gaining, which could happen in the experiment. That is why we decided to divide the experimentation sample into four groups consisting of 4 people; where each group will try to solve specific but similar tasks using the unique visualization assigned to each group. Lastly we reviewed the screen recordings of the participants so that we could check that the participants followed the instructions given.

6

Conclusion & Future Work

In this thesis we tried evaluating the different visualizations used in software traceability. More specifically we tried to see if there is a difference in performance and perception. The results of the survey conducted on professionals of the field showed that there are visualizations (graph, matrix) that are used more than others and that are perceived as more profitable to use. Furthermore the survey indicated that no matter the visualization, the information that must be presented by them should be almost the same no matter the visualization. However the experiment that we conducted on students supported the survey's results to some extent. It showed that as far as preference or perception is concerned there were visualizations that scored slightly better than others, but not to the same degree as in the survey and not the same visualizations as in the survey. Although, when it came to analyzing their performance all visualizations included in the experiment had no statistically significant difference in their results, but as mentioned before this should be taken with a grain of salt since our small sample size does not give us the necessary confidence.

This does not mean that just using any visualization will be fine, because as seen by the individual measurements, their values vary a lot. This means that there are some factors that somehow effect the user-visualization interaction. One of this factors might be the familiarity of the user with a specific visualization. This was observed slightly in this research, where familiarity with a specific technique would influence its usage and perception even though it was no better than any other. Another factor that could have affected the individual measurements might be the type of task required to be fulfilled. This is not something that we can safely confirm but is something that came up a lot during the discussions with the experiment participants. These two elements are something that more research can be conducted upon. Moreover traceability and its visualization are a human-oriented process and our small sample size does not allow us to generalize the results, but it encourages for more research to be conducted in order to definitely identify the best technique for every task. Also, our prototype and its visualizations are far from perfect and more features and functionality can be added to make its use more effective.

Bibliography

[1] O. Gotel and C. Finkelstein, "An analysis of the requirements traceability problem," pp. 94–101, 2002.

[2] G. Spanoudakis and A. Zisman, "Software traceability: A roadmap," *Handbook of Software Engineering and Knowledge Engineering*, vol. 3, 08 2005.

[3] S. Ibrahim, M. Munro, and A. Deraman, "A requirements traceability to support change impact analysis," *Asian J Inform Technol*, vol. 4, 01 2005.

[4] P. Mäder and A. Egyed, "Assessing the effect of requirements traceability for software maintenance," in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, Sep. 2012, pp. 171–180.

[5] J. Krüger, G. Çalıklı, T. Berger, T. Leich, and G. Saake, "Effects of explicit feature traceability on program comprehension," 08 2019, pp. 338–349.

[6] J. H. Hayes, A. Dekhtyar, S. K. Sundaram, E. A. Holbrook, S. Vadlamudi, and A. April, "REquirements TRacing On target (RETRO): Improving software maintenance through traceability recovery," *Innovations in Systems and Software Engineering*, vol. 3, no. 3, pp. 193–202, 2007.

[7] A. Rodrigues, M. Lencastre, and G. A. De Cysneiros Filho, "Multi-VisioTrace: Traceability visualization tool," *Proceedings - 2016 10th International Conference on the Quality of Information and Communications Technology, QUATIC 2016*, pp. 61–66, 2017.

[8] G. C. Filho and A. Zisman, "D3TraceView: A Traceability Visualization Tool," *Proceedings of the 29th International Conference on Software Engineering and Knowledge Engineering*, vol. 2017, pp. 590–595, 2017.

[9] S. Maro, J. P. Steghofer, J. Hayes, J. Cleland-Huang, and M. Staron, "Vetting automatically generated trace links: What information is useful to human analysts?" *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018*, pp. 52–63, 2018.

[10] Y. Li and W. Maalej, "Which Traceability Visualization Is Suitable in This Context? A Comparative Study," pp. 194–210, 2012.

[11] M. Ramesh, B. & Jark, "Towards Reference Models for Requirements Traceability," *IEEE Transactions on Software Engineering*, vol. 27, no. 1, pp. 58–93, 2001. [Online]. Available: http://www.springerlink.com/index/6q31q7556n440211.pdf

[12] P. Arkley and S. Riddle, "Overcoming the traceability benefit problem," pp. 385–389, 2005.

[13] J. Cleland-Huang, O. C. Z. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman, "Software traceability: trends and future directions," pp. 55–69, 2014.

[14] P. Mäder and J. Cleland-Huang, "A visual language for modeling and executing traceability queries," *Software Systems Modeling*, vol. 12, 07 2013.

[15] T. Merten, D. Jüppner, and A. Delater, "Improved representation of traceability links in requirements engineering knowledge using Sunburst and Netmap visualizations,"

2011 4th International Workshop on Managing Requirements Knowledge, MaRK'11 - Part of the 19th IEEE International Requirements Engineering Conference, RE'11, no. August 2011, pp. 17–21, 2011.

[16] K. E. Wiegers and J. Beatty, *Software Requirements 3.* Redmond, WA, USA: Microsoft Press, 2013.

[17] X. Zhou, Z. Huo, Y. Huang, and J. Xu, "Facilitating software traceability understanding with envision," pp. 295–302, July 2008.

[18] P. Mäder, O. Gotel, and I. Philippow, "Getting back to basics: Promoting the use of a traceability information model in practice," *Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, TEFSE 2009*, pp. 21–25, 2009.

[19] A. Marcus, X. Xie, and D. Poshyvanyk, "When and how to visualize traceability links?" *Proceeding of the 3rd International Workshop on Traceability in emerging forms of software engineering, TEFSE 2005*, pp. 56–61, 2005.

[20] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction.* Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[21] R. Wieringa, *Design science methodology for information systems and software engineering.* Springer, 2014, 10.1007/978-3-662-43839-8.

A

Appendix 1

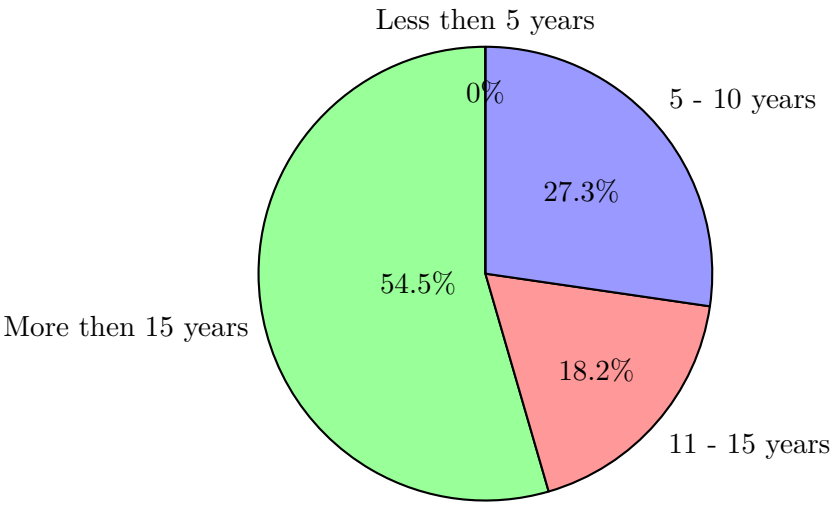
Traceability visualization questionnaire

A.1 General Questions

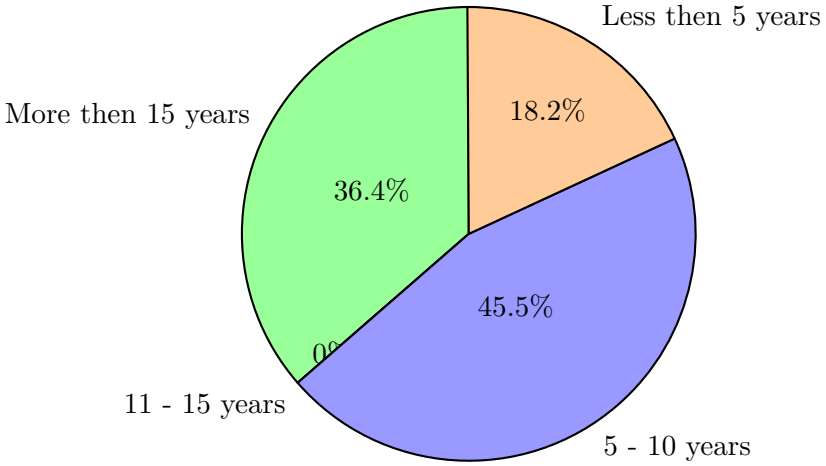
1. What is your job position?

Solution Architect
Software Architect
Technical leader
Senior System Architect
CTO
System / Software Architect
System Test Engineer
Principal Scientist
System Architect
Consultant
Assistant Professor

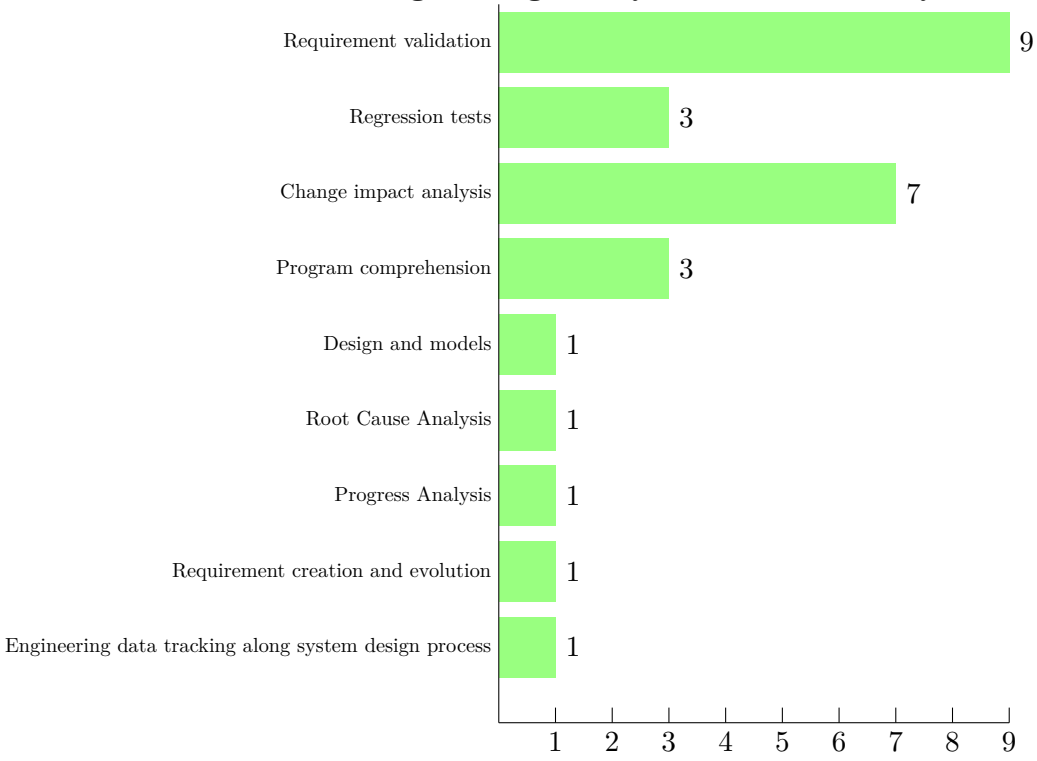
2. How many years of software engineering experience do you have?



3. How many of those years are related with traceability?

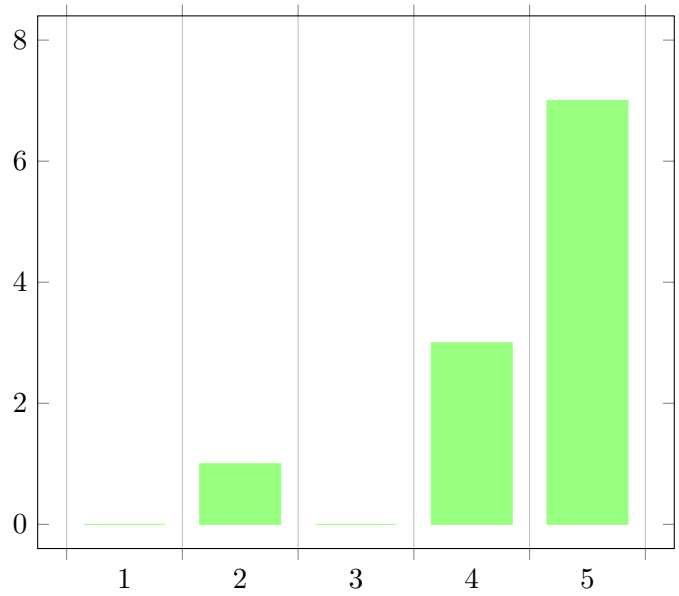


4. In which area of software engineering have you used traceability?



A.2 Traceability focused questions

1. How important is traceability in your company?



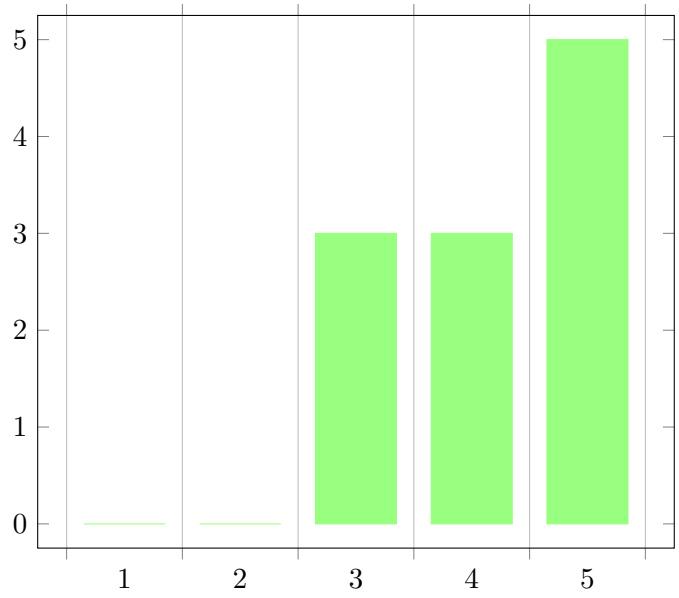
2. What techniques do you use for traceability management in the company?

Unique IDs of elements (e.g. Requirements, Components) + traceability links
Different depending on the tools. Either the tools supports it or we use unique IDs or checksums.
mainly trace tables
UML Tools, EMF based custom developments
Our own tracing tool
Linking between System Artifacts/Assets
Traceability matrices
From manual tracking to tool supported/forced tracking.
Change management, impact analysis
Excel sheets, Jira, TIM, etc

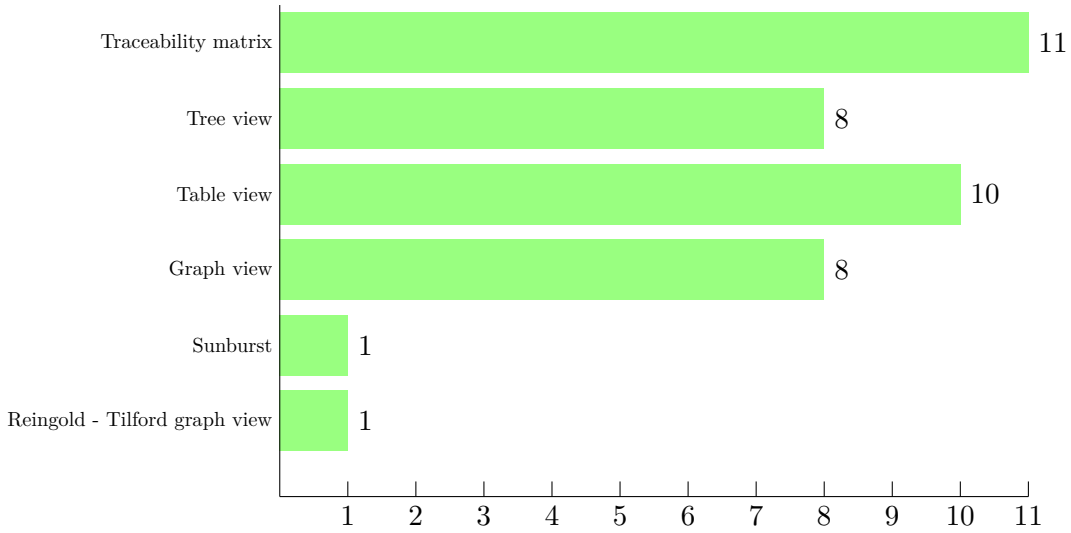
3. If you are assisted by some tool, can you please name it and describe it?

DOORS (5 answers)
Polarion (1 answer)
IBM ALM Tool Suite (1 answer)
Atlassian tools (1 answer)
Git (1 answer)
MS Excel (1 answer)
Internal tools (2 answers)
Jira (1 answer)
ReqM2 (1 answer)
TomSawyer Perspectives (1 answer)

4. How helpful is the visualization of the traceability links for you?



5. What visualization techniques are you familiar with?



6. Can you explain some problems that you have had with the visualization techniques that you have used?

I think search/filtering is the key - understanding a large graph is difficult otherwise
It is usually not very important to visualize the traceability in the complete system in one diagram. It is more important that you have tools to find loose ends and missing
trace matrices are not very helpfull to trace source code and implementation artefacts
Graphical complexity management, level of detail, usability of layout algorithms, filtering in conjunction with hierarchical breakdowns, scalability of visualization approaches (some might work with smaller datasets, but don't scale well visually with bigger datasets), responsiveness and performance
to much elements to display
Having a clear overview for large scale
The visualizations often times are cluttered with information I do not need. I should be able to prune and select the artifacts I want to see in my view.
Depending on the amount of data, views may get very confusing.
Complexity of graphs, cycles in trees, readability in tables
scalability, when the number of trace links grow visualization quality is negatively affected

7. What kind of information would you like to get just by looking at a traceability visualization?

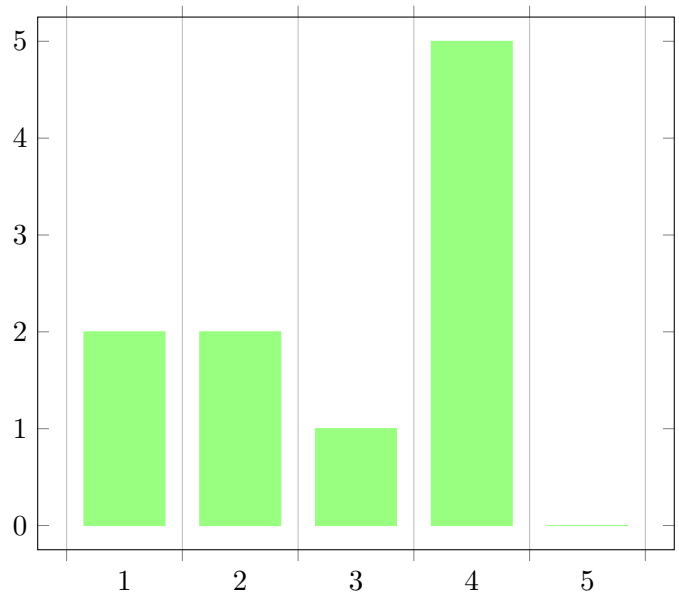
Would like to see the related elements, normally a modelled element and referenced requirement.
For a given element: all other linked elements and their types (requirement, architecture component, source code, test case etc.)
Perhaps it could be useful to find illegal connections. However, I believe this could be found automatically by the tools instead of humans looking at diagrams
main relationships between specific elements under analysis
Comprehensive high-level overviews combined with drill-in capabilities for detail views without getting lost in too much "clutter" (or irrelevant information - which is hard to define based on different usecases/required perspectives). Would like to learn more about possibilities "beyond boxes and lines", e.g. Sunburst, etc.
overview, identification of missing links
Hierarchical relationships and Missing Traces
Missing links
e.g. coverage
Responsibility, dependencies, cycles
missing trec links, links to the actual artifacts (mapping to the underlying classes, requirement specification so on), changed trace links across versions

A.3 Research focused questions

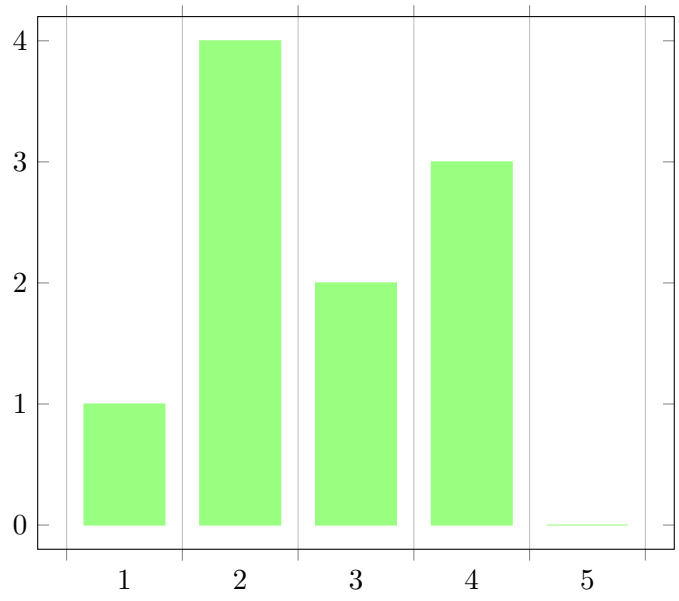
Sunburst

One person wasn't able to see the sunburst diagram in his/hers browser so that is why it is only 10 answers instead of 11 in this one

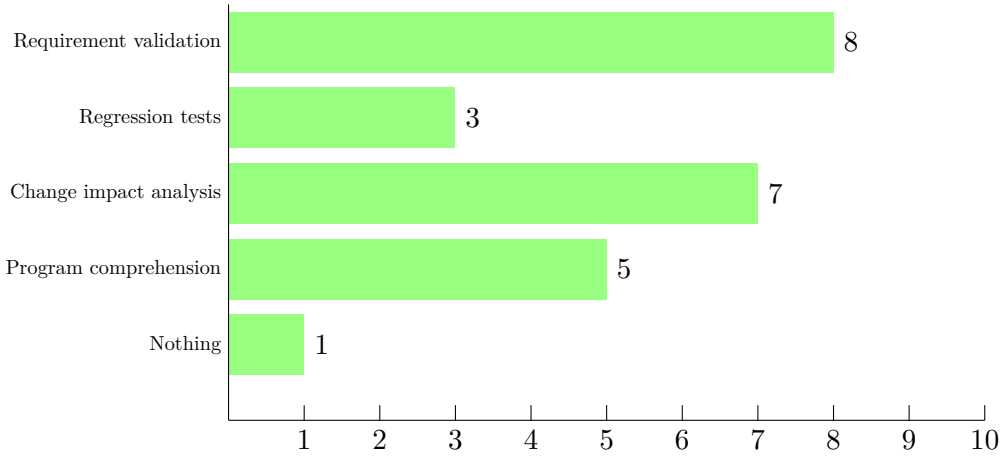
1. How would you rate the understandability of the above picture?



2. What score would you give to the visualization's usefulness?



3. For which traceability purpose do you think this visualization can be useful?

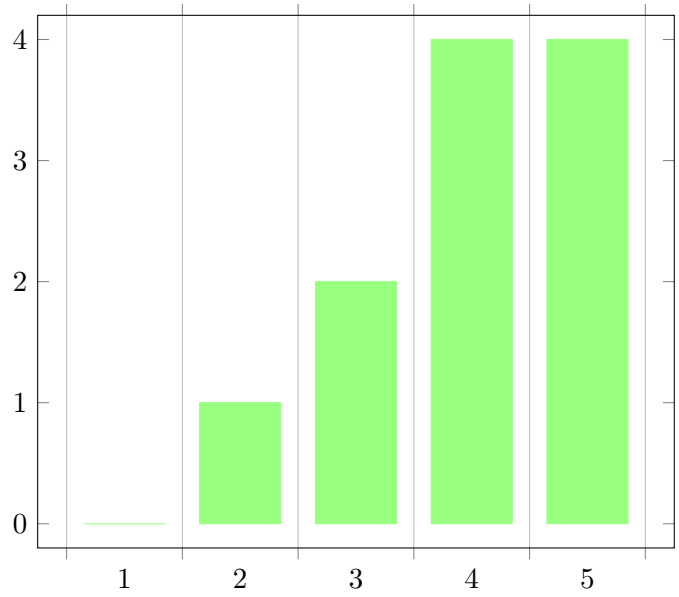


Other answers we got:

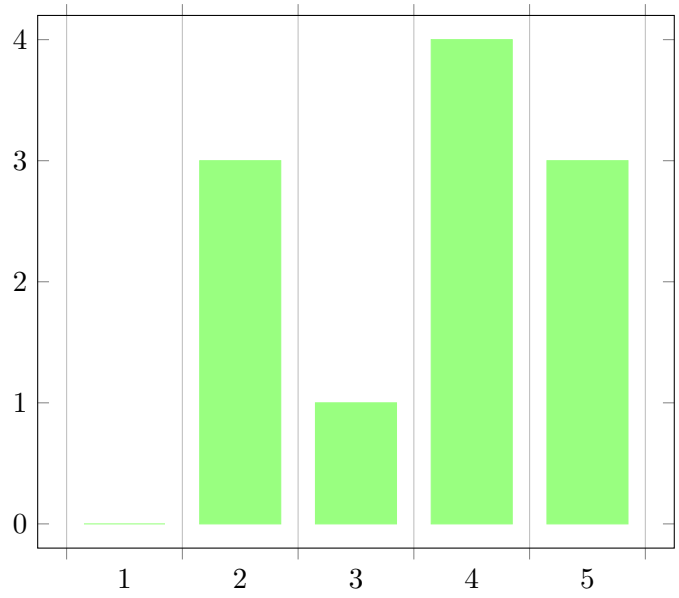
- To much information on a single picture is not useful.

Requirements Traceability Matrix (RTM)

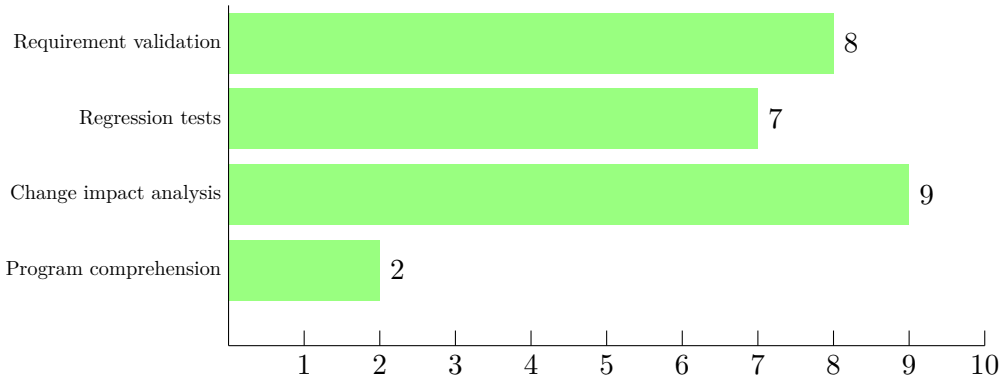
1. How would you rate the understandability of the above picture?



2. What score would you give to the visualization's usefulness?



3. For which traceability purpose do you think this visualization can be useful?

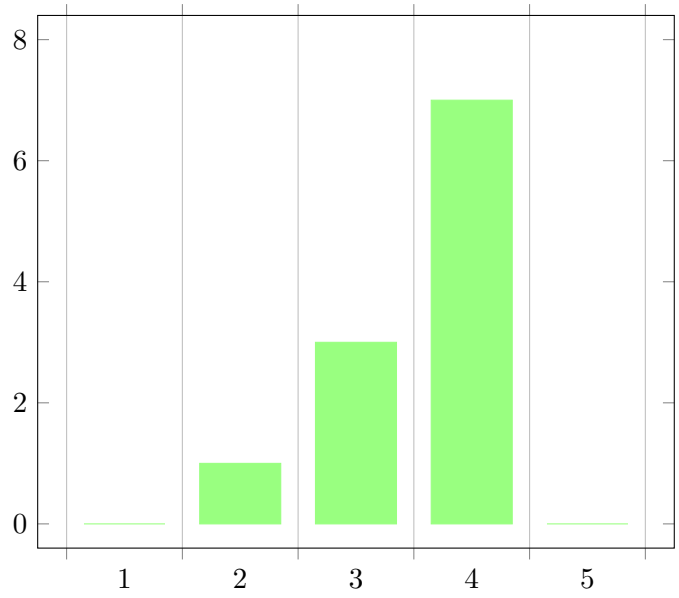


Other answers we got:

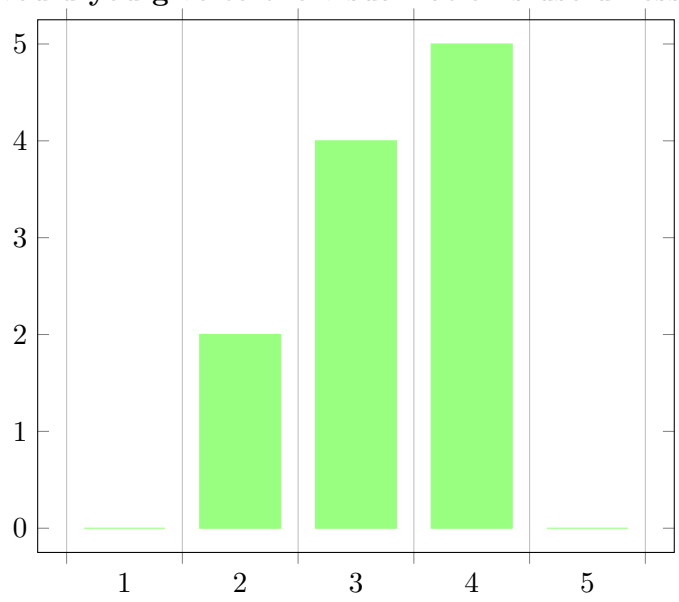
- The table shows one-to-many relations more clearly than trees or graphs; these can exist both ways (e.g. feature realized in many files, or a file realizing many features), but a hierarchy-based visualization such as a tree emphasizes just one direction.

Graph

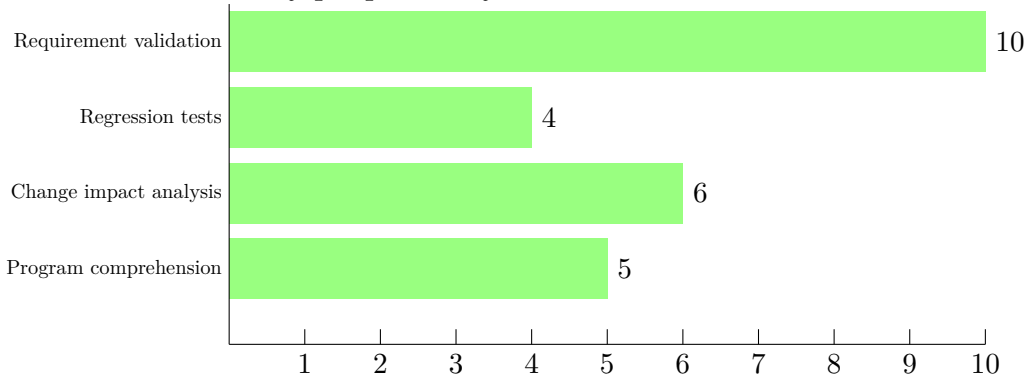
1. How would you rate the understandability of the above picture?



2. What score would you give to the visualization's usefulness?



3. For which traceability purpose do you think this visualization can be useful?



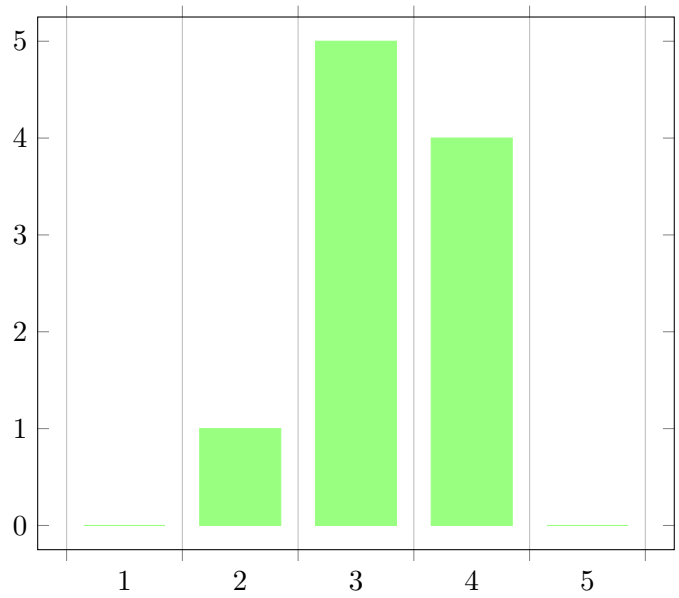
Other answers we got:

- To much information on a single picture is not useful.

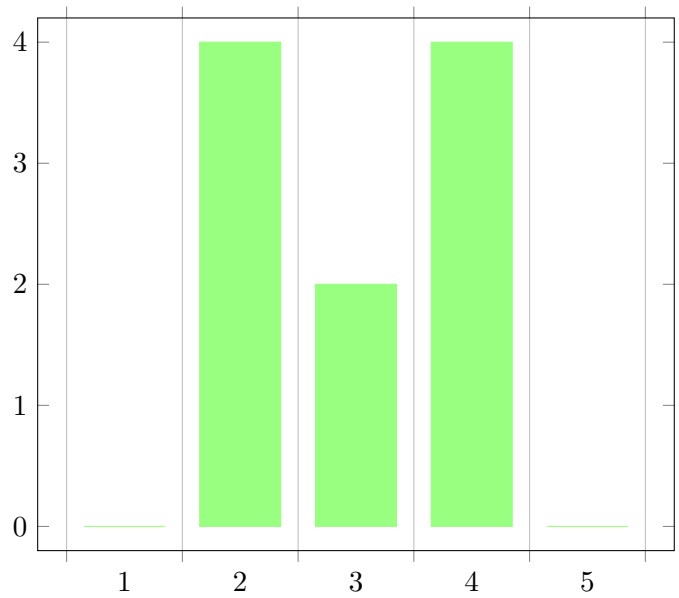
Tree view

One person wasn't able to see the tree view in his/hers browser so that is why it is only 10 answers instead of 11 in this one

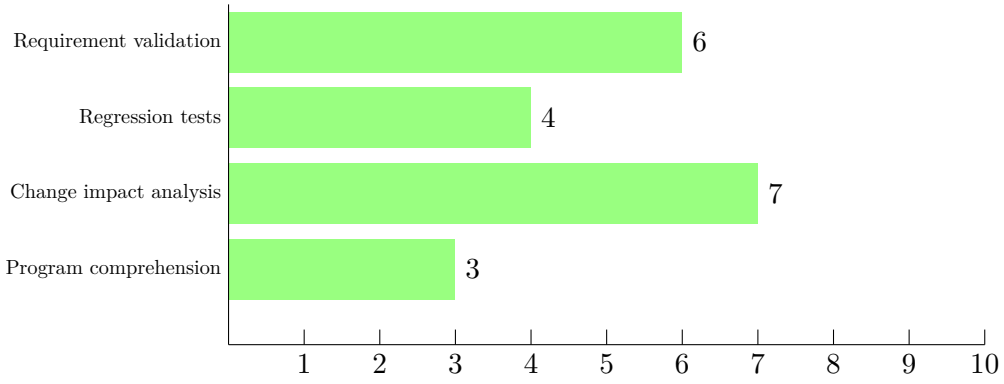
1. How would you rate the understandability of the above picture?



2. What score would you give to the visualization's usefulness?



3. For which traceability purpose do you think this visualization can be useful?

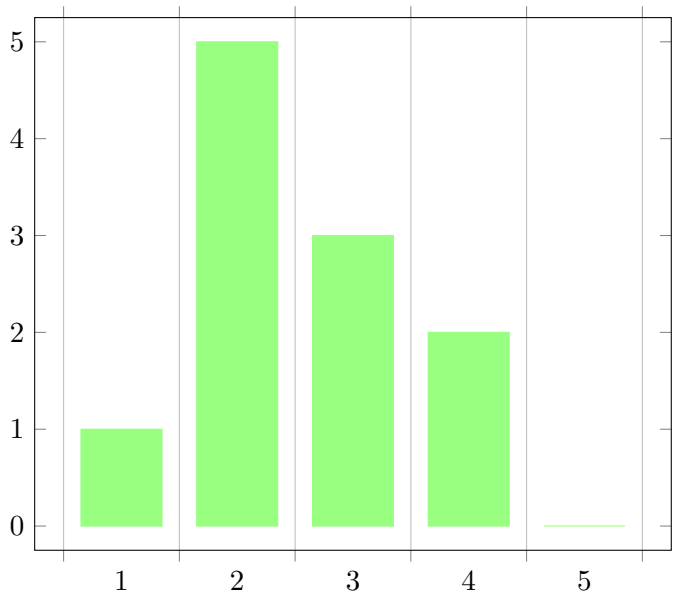


Other answers we got:

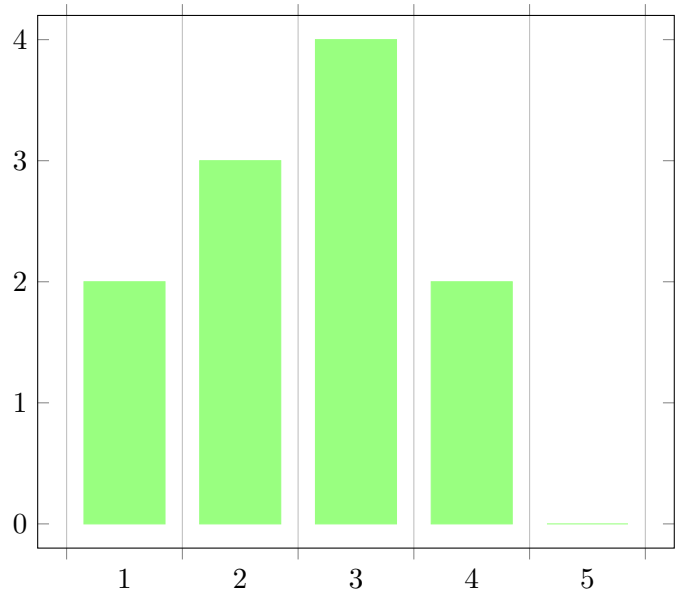
- Useful if subtree can be selected starting from any node and navigating either forward to leaves or backward to root.

Weighted Graph

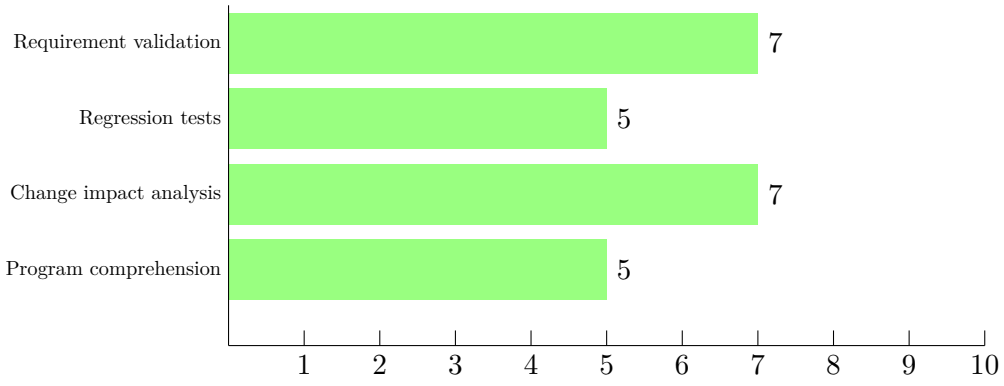
1. How would you rate the understandability of the above picture?



2. What score would you give to the visualization's usefulness?



3. For which traceability purpose do you think this visualization can be useful?

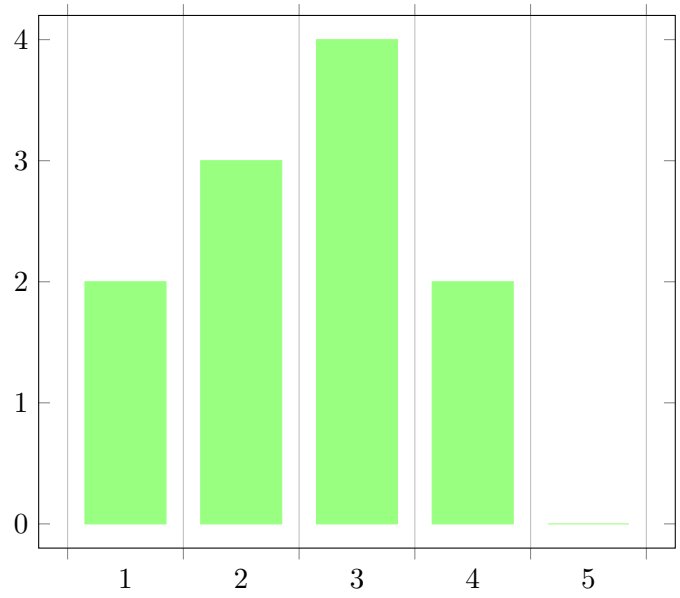


Other answers we got:

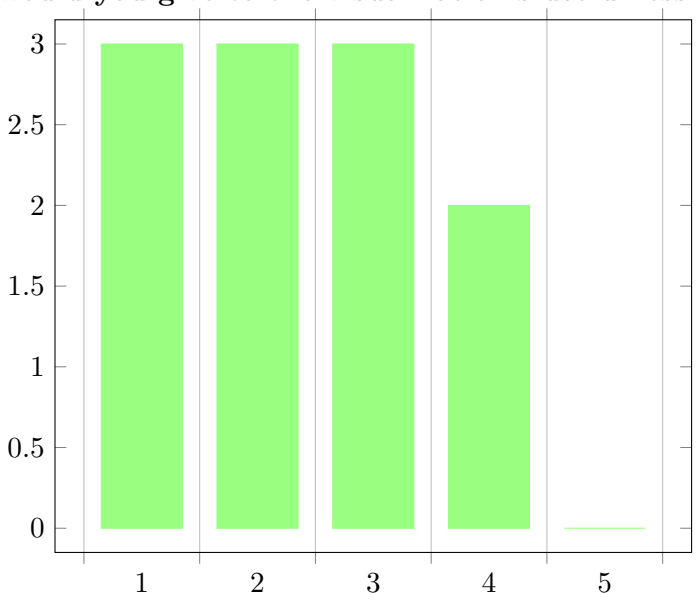
- To much information on a single picture is not useful.
- Two-sided browsing of links works ok; this would be a good addition to the table

Reingold-Tilford

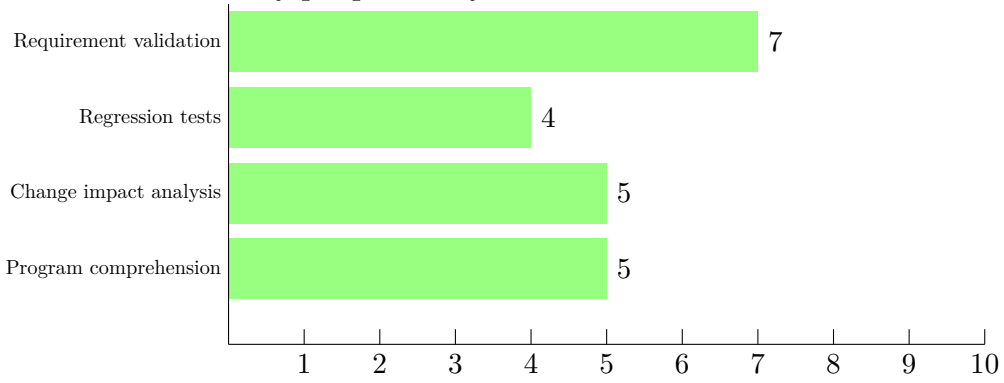
1. How would you rate the understandability of the above picture?



2. What score would you give to the visualization's usefulness?



3. For which traceability purpose do you think this visualization can be useful?

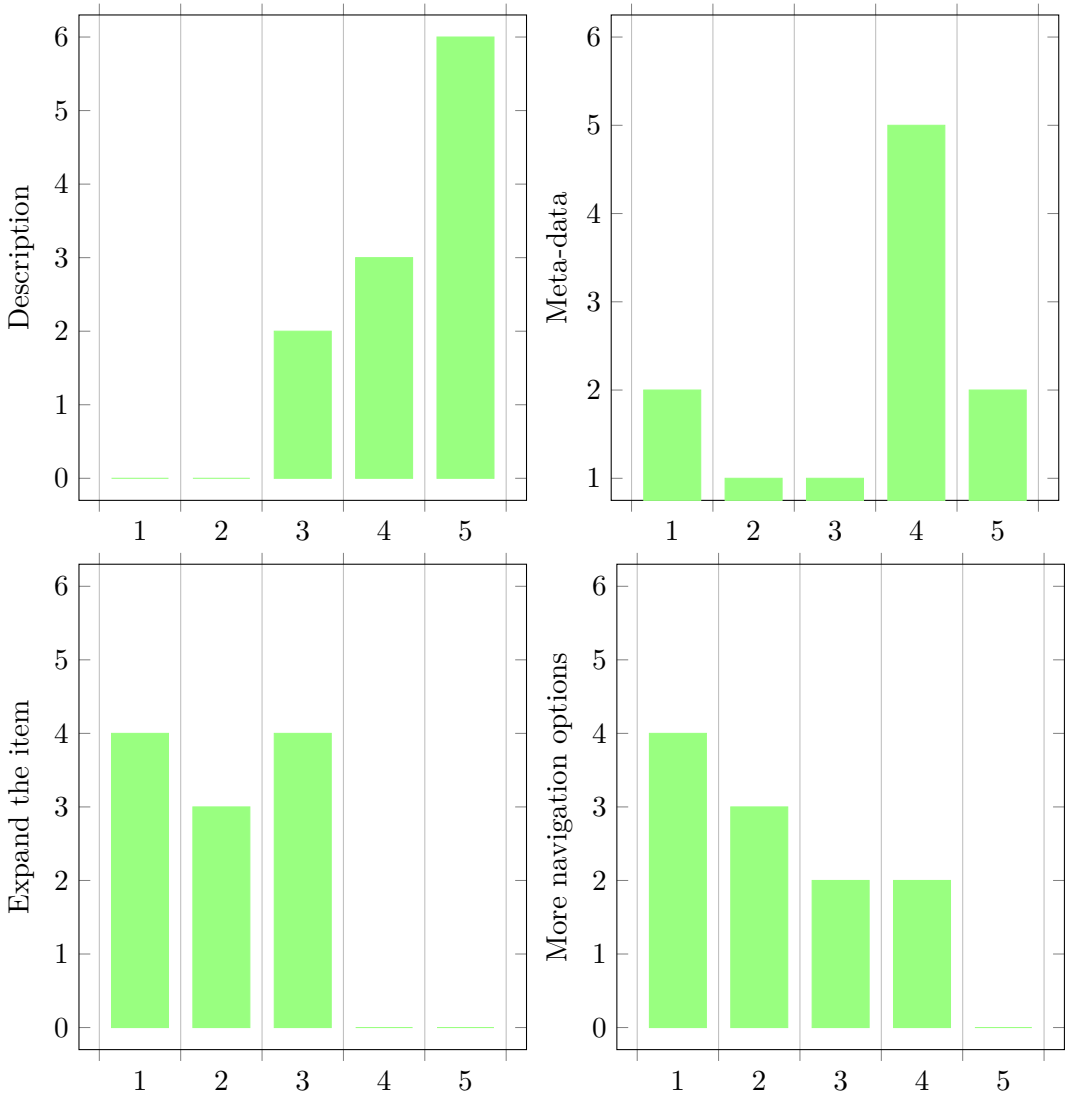


Other answers we got:

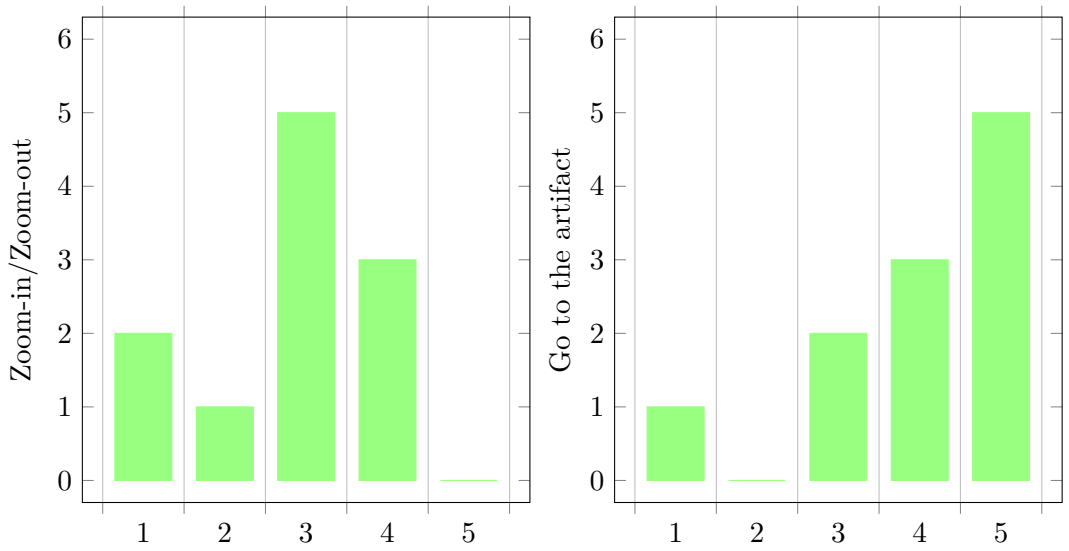
- To much information on a single picture is not useful.

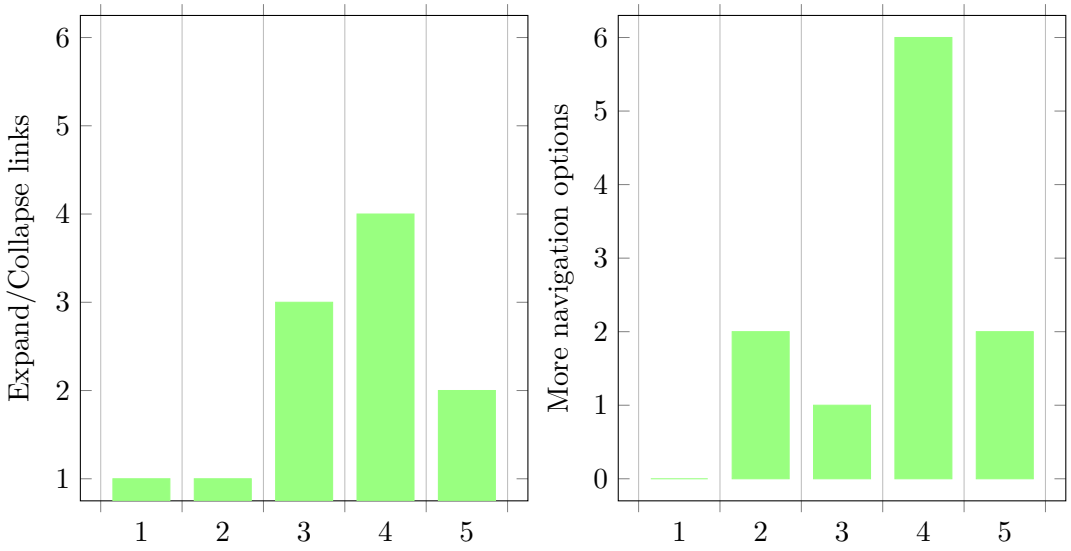
Navigation

1. When I hover over an item, I want:

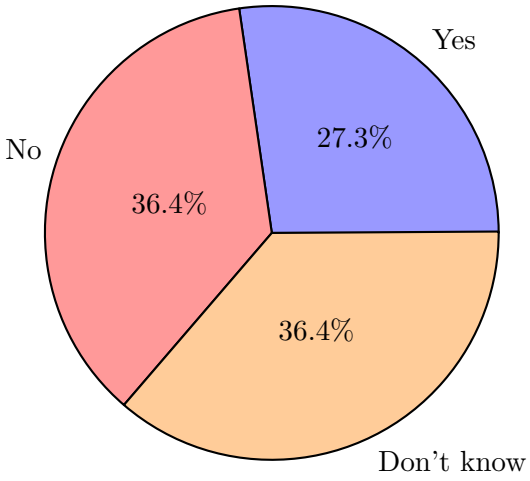


2. When I click an item, I want to:





3. Would your previous visualization scores change if you considered the above navigation options?



4. Which one would you give a different score if you were able to navigate?