CHALMERS

Visualization of Software Quality Trends

Master of Science Thesis in Computer Science and Engineering

NARJIS HACHILIF
BEHNOUSH PEJHANMANESH

Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, August 2012

Examiner: AGNETA NILSSON

Supervisor: ROBERT FELDT

Chalmers University of Technology

University of Gothenburg

Department of Computer Science and Engineering

SE-412 96 Göteborg

Sweden

Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering

Göteborg, Sweden August 2012

Acknowledgements

The authors would like to thank John Haagen, the supervisor at Ericsson AB, for his support and sharing of knowledge, which was inspiring and encouraging during this study. His positive energy and valuable suggestions were always the greatest assistance for conducting the project.

The authors would also like to thank Robert Feldt, the supervisor at Chalmers University, for his help and encouragement that made it feasible to conduct a comprehensive research in industry.

Finally, the precious support from the Ericsson's Portal Team, who provided important information for this work, was kindly appreciated.

Preface

This Master of Science Thesis is reported here in a hybrid format, i.e. the main content of the Work is reported as a scientific article conforming to the Empirical Software Engineering Journal's template.

Table of Contents

Noname manuscript No.
(will be inserted by the editor)

Visualization of Software Quality Trends

Master of Science Thesis in Computer Science and Engineering

Narjis Hachilif · Behnoush Pejhanmanesh

Received: date / Accepted: date

Abstract *Background*: Designing large and advanced software products puts specific requirements on software quality, in particular in multi-team and agile settings. One challenge is to manage and make decisions on quality efforts when the relevant data have large volume and thus becomes hard to overview, interpret and analyze. Involved staff might not only waste time, if they miss important software build and test outcome information this can result in delays in delivery as well as lost improvement opportunities. If the information can be collected and presented in a coherent and intuitive way it can support software engineers in spotting overall patterns and trends and thus support organizational learning.

Problem: A case company, Ericsson AB, was unclear on how to dynamically visualize and present information related to software quality. The objective is to investigate both how data can be intuitively represented and how it can improve efficiency of the employees using it. Previous studies have already investigated these aspects separately, but there is a lack of real-world case studies combining them.

Method: Through iterative design research we extended an existing project information portal at the case company to visualize test-related data. In order to achieve this, requirements were collected from initial interviews and workshops with users of the existing system. This data was analyzed along with results from the literature to come up with a proposal for a new visualization method. This led to the design of several prototypes and their evaluation by end users. A set of eight interviews was used to evaluate the final result.

Results: The selected prototype was then implemented as a web page displaying both an overview and a history of quality data. The integration of the new page in the existing portal was considered as an important issue to address in order to properly evaluate how this new page could improve the work's efficiency.

Conclusions: We conclude that the proposed implementation provides an intuitive visualization of large quantities of software quality data and improves the current visualization method. It helps visualizing software quality trends by displaying the evolution of results over time. The new visualization can be used both as a tool for quality data analysis at Ericsson AB, but it also provides guidelines for developing efficient software visualization tools in general. Moreover, we conclude that there is a need for further improvements in order to display data in the most efficient and intuitive way possible.

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Gothenburg, Sweden

1 Introduction

Managing large software projects requires dealing with a large amount of indicators or main measurements. These measurements are sometimes related to each other. Therefore, presenting a large amount of measurement dependencies to the user can be problematic and needs significant effort [1].

This paper aims at evaluating possible new data presentation methods in a software portal called Integration Portal which is currently used at Ericsson AB (in the following frequently referred to as the "company") as a tool for collecting and presenting quality information. Mainly targeted at software testing, it is presented in the form of a table displaying builds along with their test results. Different dimensions and types of data are

represented in the current portal, and may not be of interest for a specific type of user. Indeed, the current presentation could be confusing by providing an large number of detailed and different metrics and only a limited overview of the product quality. Users of the portal are working in various positions such as project managers, developers, designers and testers. Current data presentation is mainly based on numeric representation on detailed level for each quality type separately, which only gives a limited overview of the complete product quality. Another drawback is that some pages display condensed information that makes it difficult to perform efficient data analysis. There is a need for a visualization displaying data in a more intuitive and understandable manner in order to increase users' work's efficiency. Indeed, by providing a clearer visualization, users can carry out their tasks faster and in a better way, as they would need less effort and time to access the information they need. In this study, work's efficiency is therefore linked to the usability of the visualization.

Research has already been conducted in that area, including a study about software metrics presentation [1]. The results show that selecting the right visualizing method for a specific type of data is not trivial. One has to take into account several parameters such as usability, maintainability, ability to overview and interpret results and possibility to handle large sets of data [2]. Familiar and intuitive metaphors were found to be the ones to meet all requirements.

From an academic perspective, the challenge brought by this project is to gather and select information about data visualization in order to implement an efficient and user-friendly system. In particular, the field of visual analytics has been investigated as a suitable model to provide efficient tools for software visual analysis (SVA) [3]. Other previous studies have investigated how to provide intuitive visualization, for instance by providing a ranking of quantitative perceptual tasks [4].

In the industry, this would result in a concrete implementation integrated in the current portal and aiming at facilitating users' work. This will be done by coming up with new ways of presenting data in a more intuitive manner for a more efficient analysis. These new methods make use of existing ways to display data, both from the Integration Portal and from previous studies. Existing methods will be combined and modified based on users' feedback and will result in a new page in the Integration Portal. Previous articles will also be investigated as they can provide relevant information on how to present information to different types of users. For instance, Buse and Zimmerman designed a survey for managers and developers to find out what type of data they are the most interested in [5]. Along with interviews performed at the company, with users of the current portal, this can help us relate data visualization to work's efficiency. The research will therefore be focused on finding ways to present data, implementing a concrete example of an efficient visualization and evaluating this implementation. Based on these issues, the scope and limitations of this study are described below:

- displaying relevant information with respect to data type and data dimensions,
- providing a tool to be used in development project monitoring,
- providing both an overview of data and sufficient information to be used for every kind of user, in a common and shared view.

The paper is structured as follows; section 2 presents important visualization theories that are related with our topic. Section 3 describes the methods used to conduct this work, including research questions and data collection. Results from interviews and implementation are presented in section 4 and discussed in section 5. Finally, section 6 concludes the paper.

2 Background

This section presents previous studies related to visualization and used as a reference for this work. The first subsection describes both general and specific concepts about data visualization. The second subsection presents previous work that can be directly linked to our project.

2.1 Data visualization

Visual analytics (VA) integrates visualization, graphics, data analysis and interaction to help users in reasoning and decision making in complex situations. However, building these tools is challenging as it requires developers to be experienced in information visualization, graphics and technologies of user interaction design. Indeed, understanding the connection between the required information needed for decision making and the available data is a key factor in applying visual analytic to a new domain [3].

Application of analytics has changed the process of decision making in many areas. For instance, web analytics reduces the click-stream data volumes in order to provide managers with good decision making possibilities in different aspects of business ranging from content management to investments [5].

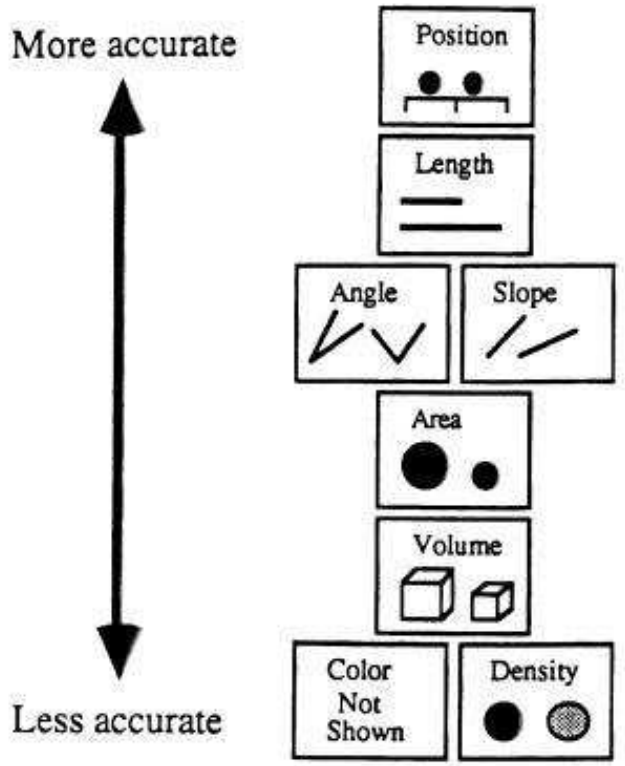

Fig. 1 Accuracy ranking of quantitative perceptual tasks [6].

Cleveland and McGill have conducted a study on graphical perception, that is how to visually decode information encoded on graphs [4]. They selected elementary tasks that people carry out when extracting information from graphs, and organized them depending on how accurately they are performed. Their analysis showed that in order to extract information from one type of graph (for example a bar chart or a pie chart), one has to perform one or several specific tasks (judging position, length, area...). Experiments were conducted in order to order tasks on the basis of how accurately people performed them. Results can guide us to choose a type of visualization that helps the user extracting the information as accurately as possible.

Mackinlay has also performed an accuracy ranking for quantitative perceptual tasks (Fig. 1). Based on this layout, the higher tasks (such as position, length, angle and slope) are carried out more accurately than the lower tasks (such as area, volume, color not shown and density) [6]. The layout provides a helpful guideline for designing an appropriate GUI with large sets of data.

Shneiderman has proposed a *Visual Information-Seeking Mantra* in order to design advanced graphical user interfaces and to understand varied set of data. This provides a set of design guidelines following the mantra: "overview first, zoom and filter, then details-on-demand" [7]. These ideas can help us to present data rapidly and allow for fast user-controlled exploration.

Representing data visually effects how the structure is perceived in this data. According to Rogowitz and Treinish, "in order to accurately represent the structure in the data, it is important to understand the relationship between data structure and visual representation" [8]. This can differ in various data types, for example nominal data, ordinal data, interval data and ratio data. We also need to have an appropriate usage of colors, since the perceptual effect of a color cannot be predicted from knowledge about blue, red and green. Therefore, without providing guidance about the psychological and physical characteristics of the colors, the user is at loss. In order to address this issue we can provide the users with a default *color-map* [8].

In order to make the user extract information in a fast way, we can use results from the study conducted by Healey, Booth and Enns on preattentive processing. Preattentive processing corresponds to the organization of the visual field based on cognitive operations that are rapid, automatic and do not require focused attention [9]. Several features such as size, spatial location and hue, were studied in relation to their capacity of helping display high-dimensional data elements in a low-dimensional environment such as a computer screen. Experiments show that some of these features allow rapid and accurate estimations without interfering with one another. This result provides a relevant guideline for choosing display properties and developing effective multi dimensional interfaces for visualization.

In order to find a suitable method which meets users' requirements while presenting data in an intuitive manner, we also investigated several types of visualization methods that are described below.

The thermometer model is a method used for displaying indicators' status [1]. It has the advantage to be intuitively understandable and to clearly present and separate different kinds of data. However, it can easily become too cluttered and therefore difficult to read, and it does not necessarily display data in a precise way.

The speedometer model is also considered as an intuitive understandable metaphor. It contains an arrow displaying the indicator's current value, along with three or more different colors in order to alert the users about the data status [1].

The dashboard overview is a popular method for displaying a large number of indicators. Unlike the speedometer model, it is not restricted to a limited number of colors. The range of colors is therefore defined according to the users' requirements. Another advantage of dashboards is that is they can provide more detailed

information such as indicator's name, indicator's value and decision criteria [1].

Bullet graphs are also a common way to display quantitative measurements by presenting them over a linear scale [10]. The advantage of this method is that all information is visible even when scaled down, which makes it a suitable method for displaying small multiples [11].

Sparklines display information in form of a chart without axes or coordinates. The problem with sparklines is that the density of the chart can lead to a loss of details when having low resolution of display. However, they are considered suitable for comparing values over time [12].

Heat maps constitute a common visualization method by providing a matrix in which the value of data is represented by colors [10]. A major advantage of this technique is that it can both be compact and present an important number of values, which constitutes one of the requirements of our work.

All these methods propose ways to display information in a user-friendly manner by investigating graphical perception, perceptual tasks and various data visualization techniques. However, they do not necessarily address the impact of visualization methods on the work's efficiency. This is the problem we are aiming to address in this study by investigating how and what should be visualized in order to improve the work's efficiency.

2.2 Related work

Development of software projects has been always unpredictable and software developers are still experiencing failures or long delays during the development process.

There is also a permanent gap between the information required by managers for decision making and the information delivered by the current tools. This is due to misunderstanding of the managers' real data needs [5]. The managers requirements has been ignored in research area since the focus in the research is more on developers' information needs. The important issues from project managers' perspective are more related to high level concerns such as: resource allocation, monitoring the work of developers, software designers and testers. For developers, issues such as code, architecture and performance are of higher importance [5].

When the requirements are not covered due to unavailability of the tools, difficulty of using them or their inappropriate presentation of data to users, managers should rely on their past experience in making important decisions in the project which might not go well [13]. Indeed, important decisions should be rather based on facts in order to prevent risks such as late deliveries and unnecessary costs.

In visual analytics, data and systematic reasoning are applied for decision making. Therefore, instead of only taking the metrics into consideration, visual analytic could help users to change from answering "What happened?" to answering to "How did it happen and why?". In this way, more information is collected via different types of analysis, which enables the users to filter and summarize the information [5].

In the area of software quality, three elements are considered to be important to measure: source code with its associated changes, results of executing test cases and alterations in test cases. However, during the development of software projects, changes related to the source code and results of tests for the whole system over time are not considered as the main metrics to support decision making process. Since it is challenging to visualize large amounts of data for the whole system, data related to failures is taken into account when a problem occurs in the development of software projects [14].

Software visualization tools are suitable methods used to help managers and developers to cope with data complexity. By using graphical techniques, software visualization tools make software more understandable by displaying different program artifacts and behavior. The strategy used for visualizing large-scale software systems is often to split them into smaller parts (modules) and visualize them individually. However, in practice, decomposition of software in modules is quite challenging and often the "big picture" of the project state is lost [15]. On the other hand, Buse and Zimmerman [5] and Ko et al. [16] argue that requirement diversity is often unavoidable.

In order to select the requirements for information visualization needed by decision makers, Buse and Zimmerman designed a survey for managers and developers working at Microsoft. According to the findings, data and metrics are considered as important information for managers in their decision making, whereas developers considered their experience to be more important [5]. Metrics used by developers and managers were also investigated. By comparing them, it was found that the most important information was related to project failures. However, at least half of the engineers and managers would use all metrics if available [10]. In conclusion, people working in different positions are interested in different information and this is due to the nature of the software project itself. Buse and Zimmerman conclude that in order to support information needs in the

area of software development, changes need to be done and data collection should focus on users' needs [5].

A study conducted by Jones et al. aims at presenting errors and faults in an obvious manner [17]. A visual mapping was proposed in order to identify and locate failures. This can provide helpful guidelines in order to highlight errors in projects that can present on-time deliveries.

Analysis of previous work point out several things that should be taken into consideration when dealing with large and complex data. In order to make users cope with the complexity of the displayed information, it is important to present the right data and to present it in the right way. This could provide an overview of results rather than going into too much details and risking that users miss important information. A focus is made on gathering requirements, since different users need to access different information. The visualization would therefore need to carefully select the data that could be useful to many types of users and facilitate communication between them. Previous studies were therefore helpful in order to find out what should be taken into consideration in our project, but they do not provide concrete examples of which data should be displayed and how, which is what we aim to investigate in this paper.

3 Methods

This section describes the methods used during this study. Section 3.1 describes the context of the project, which was conducted at the company. Section 3.2 presents a set of research questions and sub-questions investigated during this study. Section 3.3 introduces the design research illustrated by a flow chart. Section 3.4 describes how data was collected and analyzed during the project. Finally, section 3.5 explains what validity threats have been encountered in this study and how they have been addressed.

3.1 Context

Ericsson AB provides telecommunications equipment, data communication systems and services to mobile and fixed network operators. Over 40 percent of the world's mobile traffic passes through the company's networks, with more than 180 countries using its network equipment [16].

The study was performed at Ericsson Lindholmen, where the focus area is mobile internet [18]. It was conducted with the Integration Portal team that deals with the web portal used to present software quality data.

Table 1 Data types currently used in the Integration Portal.

Data types
Unit and component test results
Code coverage data
Memory errors (Purify, Valgrind, Coverty Prevent)
Function test results (REG, PTB)
System test characteristics
Code changes

Data types currently displayed in the Integration Portal are listed in table 1. As seen in figure 2, for each of these data types, a small rectangle is displayed with a different color depending on whether the test has failed or not. Additional information such as number of failing test cases and total number of tests are presented as numerical values, whereas the evolution of test results is displayed with arrows.

The drawback of this visualization is that it does not provide an intuitive and fast way to analyze and compare the evolution of results build after build.

Indeed, several users of this web portal have expressed the wish to improve it. According to them, the integration portal provides only a limited overview of software quality data, which can make it difficult and time consuming to analyze the information. The goal of the project was then to improve these aspects and provide a focus on data dimensions such as time dimension, feature/functionality dimension, test type dimension and codebase dimension. Available quality data can indeed be viewed from different angles, depending on which dimension is being considered. For instance, when a set of tests is being available in a single point of time, the feature/functionality dimension is the testing functionality. On the other hand, one can focus on the result of a test over time, in which case the time dimension is being considered. The challenge is therefore to present this complex collection of data in a way that is easily understandable to the user, and by taking into account that not all dimensions can be viewed simultaneously without providing a collection of "views" [19].

Furthermore, all data types are not needed by every kind of user and there is also a challenge in selecting and presenting the data that will provide a significant improvement in the work's efficiency of different kind of employees. Showing the right data to the right people would allow them to react early and make proper decisions in order to avoid issues that prevent on-time deliveries.

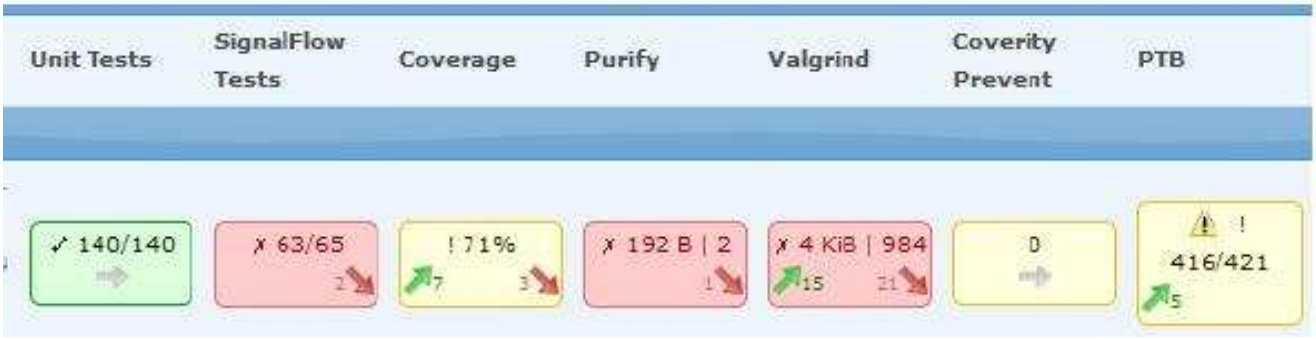

Fig. 2 Current visualization in the Integration Portal.

3.2 Research questions

A set of research questions and sub-questions has been investigated as a guideline through the project. RQ1 and RQ2, as well as their sub-questions have been studied in the prototyping and the implementation phase, whereas RQ3 refers to the evaluation of the prototype and its implementation.

RQ1:
What is a suitable method for visualizing large quantities of quality data generated in large scale software development projects?
SQ1.1: How current and historical data generated from different test types should be managed for presentation?
SQ1.2: What are the main requirements and quality attributes for the presentation methods?
SQ1.3: What channels (size, shape, color, sound or movement) are suitable for presenting data to the users?

RQ2:
Which quality data should be visualized and how shall it be combined to increase the information throughput to the user?
SQ2.1: What type of data is the most important to extract?
SQ2.2: How can we combine data of different dimensions?
SQ2.3: How can data be translated into information channels?

RQ3:
How does the visualization method effect the user's communication about the product?
SQ3.1: What benefits will be seen by users?
SQ3.2: How can the visualization improve the user's work's efficiency?

The first research question deals with finding the right methods to visualize important amounts of data. The first sub-question deals with studying how different data types should managed, i.e. displayed in a structured way. The second sub-question refers to the gathering of requirements and the third one deals with finding out which information channels can be used to present software quality data. The term channel is used to describe shapes, colors or symbols that provide information to the user.

The second research question refers to which data should be selected in order to display the right information. Finally, the last research question is related to the evaluation of the visualization and how it can improve the work's efficiency.

3.3 Design research

Design research methodology [20] has been used in this study. As seen in figure 3, the design research consists of 6 phases. Once the scope and limitations of the thesis were defined, previous studies were investigated and existing articles were reviewed in order to come up with a proposal for a visualization method. These phases of problem awareness and literature review were followed by a suggestion for a new visualization method. This led to the design phase which resulted in several prototypes. The selection of a prototype and its implementation were followed by a final evaluation with end users. However, evaluation was already being done while designing and implementing the prototypes, hence the iteration from the Evaluation phase back to Design and Implementation (Fig. 3). For instance, interviews were conducted to gather requirements, select the correct prototypes and develop a relevant implementation.

3.4 Data collection

Three types of interviews were performed as part of the evaluation of the study, as seen in table 2. These different evaluation methods are described in more details in this section.

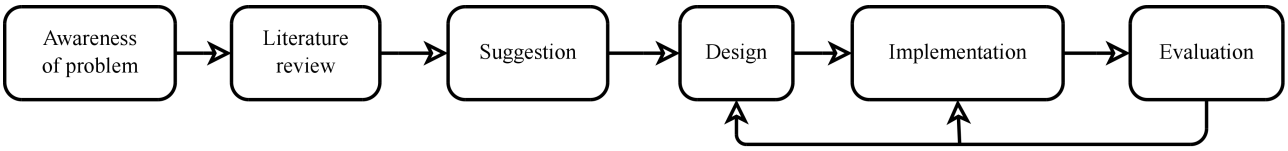

Fig. 3 Flow chart illustrating research design.

Table 2 Evaluation methods.

Evaluation method	Objective	Project phases
Focused group interview	This method aimed at gathering requirements from a group of users who occupy different roles in the company	Suggestion and design
Prototype evaluation	The method was used for exploring and evaluating early interface designs, suggestions were gathered in an interview with a project manager	Design and implementation
Individual demonstrations and semi-structured interviews	This interviews led to the evaluation of both the prototype and its implementation	Evaluation

At the beginning phase of the thesis work, a focused group interview was conducted as a technique for requirements elicitation. Focused group interview is a qualitative research method used to reveal how people think about an issue or product. Using this technique, researchers can interactively question a group of participants to solicit ideas and views about a topic [21]. Some of the benefits of using focused group interview technique are: involving the participants in a shared concrete situation, analyzing the specific context, using the interview guide (questionnaire) to test hypotheses and focusing on the subjective experiences of participants [22]. However, Grim et al. argue that this technique has some disadvantages. For instance, discussion can be dominated by a few vocal individual and some people might not get attention. Besides, the information could be difficult to analyze [21].

There were eight participants attending the interview and representing different roles such as function testers, software designers and developers. The interviewees were selected deliberately from all different roles that deal with the current representation of quality data at the company. In this way, they could provide us with their wishes concerning a visualization method and bring up different aspects of the visualization that should be taken into consideration before implementation.

The interview questions included different themes in order to document the major needs and requirements of the users group (see First interview questions in the appendix I). The purpose of using different themes in the questionnaire was to cover multiple requirements from various users' perspectives. In some cases follow up questions were brought into discussion to give a good insight about the issues. The session lasted nearly 50 minutes and results gained from the interview were recorded and applied in the paper prototyping phase. The answers provided a guideline on how to design prototypes by following the users' needs. According to them, the visualization should take into account different types of users and their corresponding needs. Designers expressed the wish to access a visualization centered on builds, whereas testers would prefer an interface based on test results. Both types of users think that managers would be interested in an overview rather than a too detailed presentation. Users are also interested in a visualization that provides traceability. In the Integration Portal, following test results build after build can be problematic and it can also be difficult to get a comparative view of results. Other suggestions concern the wish to provide more control to the user (with customization for instance) and to be notified when a test case fails.

Paper prototype is a validated technique for exploring and evaluating early interface designs. This technique provides many advantages during the design process. It allows for instance rapid externalization of design ideas with low costs and it generates numerous alternatives early in the design cycle. It also provides a good opportunity for designers to iterate on a design many times before they commit to the implementation. Evaluation of the prototypes are more focused on macro level issues (major interface screen) and since the design appears informal, designers and users tend to offer various critiques of the design [23]. There were different ideas that we could come up with during this phase. Data gathered both in the literature review and during the focused group interview was used to design prototypes. The challenge was to take into account and combine different suggestions expressed by different types of users. The first prototypes included several views

and visualization methods such as dashboards and bar charts. They were updated based on feedback and comments from users. Prototypes were also discussed during a meeting with a project manager that led to the final prototype that was chosen to be implemented. This confirmed the idea that an overview visualization would be helpful for project management. It was also pointed out that providing only one or two views would be a better way to provide interaction between the different types of users. Indeed, if several views are displayed, users with different roles do not access the same data and are less likely to be able to interact and communicate about the same information. Furthermore, the Integration Portal already provides several views displaying different types of data. Providing a single view to be used by several types of users could be a way to provide an improvement and possibly improve the users' work. There is still a challenge of making this view useful for different types of users. The suggestion at this point was to display one bar chart related to the build's status, one bar chart related to the trouble reports and other deliverables and one table displaying the test results. Traceability and history of results would be then provided by the bar charts. This prototype was once again evaluated with users. Along with knowledge about data visualization, it was found that it could be simplified in order to provide more clarity. This led to a design based on a single table that includes build names, deliverables and test results. This final prototype is described in the Results section.

After implementing this prototype, a set of individual demonstrations and interviews were conducted in order to evaluate both the prototype and its implementation (See Summarized interview questions in the appendix I). The same approach was used for the second round interview and there were some follow up questions for some of the themes in order to encourage suggestions. Ten interviews have been performed with interviewees holding several positions as described in table 3. All of these interviewees deal with software quality data to a certain extent and in different ways. Developers, designers and testers use the portal in a more frequent ways than managers for instance. Some interviewees such as IT architects do not use the portal on a regular basis but their opinion was helpful to evaluate the final implementation.

Each interview lasted 30 minutes and was recorded and used as an evaluation of the final work. Results from these final interviews are also described in the results section.

Table 3 Roles of interviewees during the final evaluation.

Role
Software Designer
Software Developer
Functional Tester
Test & Verification Engineer
IT Architect
Configuration Manager
Technical Test Coordinator
Project Manager

3.5 Validity threats

The main issue encountered is related to the complexity of the designed prototypes. The challenge was to select a method realistic enough to be implemented that still provides a significant improvement to the visualization methods currently used. It was also challenging to design a prototype that meets all requirements and could be easily accepted by the team. This was dealt with using evaluation methods such as meetings and interviews. In these interviews, validity threats have also been taken into consideration. Previous studies have shown that these issues are difficult to avoid [24][25] and should be considered from the beginning of the study.

In software engineering the aspects of validity and threats could be classified to validity in different ways. One of the schemes proposed by Runeson et al. [26], and also used by Yin [27] distinguishes between four different aspects of validity which are construct validity, internal validity, external validity and reliability [26].

Construct validity describes the extension to which the operational measures really convey what the researcher has in mind and to which the investigated issue is based on the research questions [26]. For instance, if the researcher and the interviewee do not interpret the constructs of the interview questions in the same way, the construct validity may have a threat in the research area. This aspect could be considered as a threat when we conducted a focused group interview, since one of the drawbacks with focused group interview technique is the domination of ideas which prevent some people in the group to have the same interpretation of the interview questions. In order to mitigate this risk, interview questions were designed with suitable themes and explanation so that to resolve any misunderstanding as much as possible. On the other hand, since the visualization was highly dependent on the results of the in-

terviews, interviewees were selected from different roles with extensive knowledge who were dealing with the current visualization at the company. Thus, the results of the interviews contain different types of users' perspective. Both interviewers being external researchers, there was a potential risk that the interviewees ideas were misinterpreted. However, since this visualization method was a continuation of the previous visualization work, this issue was not considered as a large threat.

Internal validity refers to the examination of causal relations, that is when a third factor is not taken into consideration when investigating a relation between two factors [26]. When studying whether one factor is effected by another factor, there is a risk that a third factor is not taken into account. Considering the nature of the project, internal validity is not considered in this study.

External validity is related to the extension to which it is possible to generalize findings and to realize how much people outside of the studied case are interested to the findings [26]. Since this study is conducted as an industrial thesis, results were only posed to a single company. This could be a potential threat to generalization of the study's results. The risk was mitigated by getting inspired from the previous representation of software quality data which is the current visualization used at the company. It is also possible to generalize visualization of test-related data to other companies dealing with software quality data.

Reliability deals with how much the analysis has dependency on the specific researcher [26]. If the same study was conducted by another researcher later, the results gained from it should be the same. For instance, the ambiguity of the interview question can lead to a threat to reliability as interviewees can interpret the same question differently. We tried to address this by writing precise and detailed questions, along with a proper explanation of the visualization.

4 Results

This section presents the results gathered during and at the end of the project. It is therefore focused on the last phases illustrated by the research flow chart on figure 3. Section 4.1 presents results from the prototyping and design phase. Section 4.2 describes the actual implementation, illustrated with screenshots and pseudo code. Section 4.3 presents the results from the evaluation conducted at the end of the study.

4.1 Prototyping

From the first design of prototypes, the idea was to make a prototype that provides helpful software build information for software testers and designers, as well as an overview of the project status for project managers. In this way, various groups of users could use the benefits of the same page view. The emphasis was put on the importance of providing an overview, rather than going too much into details, and providing the user with a history of data. The first suggestions consisted in one bar chart displaying the build and status of each sub builds and one bar chart displaying the trouble reports for different builds and their status. The history would be displayed in terms of bar charts. This idea has been replaced by a view focusing more on specific data types but the idea of a vertical history has been kept. Table 1 shows a list of data types already displayed in the current version of the portal.

Based on discussions led with users, we considered that function test results (REG and PTG) and memory errors and leaks are the most interesting data to display, along with deliverables such as trouble reports. Our goal is to make errors and failures occurring in these tests as obvious as possible. Indeed, we believe that displaying this data in an intuitive manner will help designers and project managers to interact in a more efficient way. We will not focus on Coverage data or Unit tests and Signal Flow tests as they should in principle never fail. Due to time constraints, the idea of notifying users with failing test cases has not been kept, but the visualization aims at making failing test results as obvious as possible.

The prototype was designed using papers to represent the main interface screens and colored markers to sketch contents. As shown in figure 4, build related data considered important from designers and project managers points of view is taken into consideration, such as build name, deliverables, trouble reports, REG, PTB and Purify test results. In this prototype, the green color corresponds to the passed test cases and the red color corresponds to failed ones.

4.2 Implementation

In order to finalize the mentioned paper prototype, some alterations were done according to the received feedback from different users. Indeed, evaluation was done during and after the prototyping phase as illustrated in figure 3. This evaluation led to changes that were applied in the final implementation as described below.

The application was implemented using PHP, HTML, CSS, MySQL and the Drupal framework. These tech-

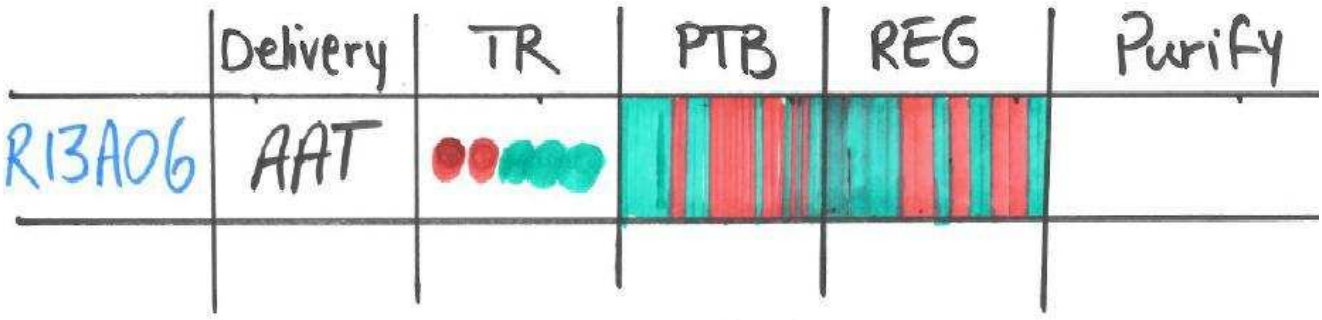

Fig. 4 Proposed prototype for the new visualization method.

nologies are already used in the current portal implemented at the company. Drupal is an open source Content Management System implemented by an important community of developers and distributed under the GNU General Public License [28]. It is written in PHP, supports diverse web technologies and is used as a back-end system for several websites across the world. Drupal was developed according to principles such as: an architecture based on modules, a commitment to quality coding and standards, low resource demands and collaboration between users [29]. Having little experience of this framework, it took us some time and explanations to be able to use Drupal efficiently. However, Drupal also aims for high usability and after encountering some difficulties, it has proved to be a helpful tool in our work. The quality data is stored in MySQL database and a copy of the database was provided to us for implementation and testing purposes. The database is characterized by a complex architecture where test results are being mapped to builds in different ways. It constituted a challenge to understand how data was connected and how to retrieve the information we were interested in. From a technical point of view, SQL statements were written using a set of PHP libraries called Doctrine. In particular, we used the Doctrine Object-Relational Mapping, that allows to write database queries in a dialect called Doctrine Query Language [30]. Even though we were not familiar with this technology, we were able to manage these difficulties, notably thanks to already available APIs in PHP.

The purpose of the implementation was to provide the user with a history of test results for the latest builds and for a specific product. The user can compare the results from different builds and gets an overview of the evolution of results build after build. Test results are also displayed along with deliverables, so that the user can relate them with each other. As illustrated in figure 5, once the user has chosen a product, a table is displayed containing build names, deliverables and test results. The build name column represents the latest builds, which can be accessed by clicking on the corresponding names. Latest builds were retrieved via an API function that takes into parameters the chosen product, the limit for the oldest build to be retrieved and the maximum numbers of builds to be fetched. For each build, data related to deliverables and test results (REG, PTB and Valgrind) is represented along the line. Originally, Purify results were represented in the paper prototype (Fig. 4), but Valgrind results were considered by users to be more interesting to display. Table 4 shows a list of deliverables being retrieved.

Table 4 List of displayed deliverables

Deliverables
Trouble Reports (TR)
Anatom (ATM)
Extra View TR (EV)
RedMine TR (RM)
Change Request (CR)
Requirement (REQ)

Deliverables are displayed inline using different widths. For instance, 1 pixel is used to display zero deliverable, 5 pixels for displaying only one deliverable, 10 pixels for displaying 2 to 4 deliverables and finally 20 pixels are used for displaying more than 5 deliverables. By hovering, the user can see a list of all deliverables names and know exactly how many deliverables there are for each build.

For each build, the test results column display the corresponding failed test cases, sorted alphabetically. Each red line corresponds to a test case that has failed and which name is displayed by the hovering feature.

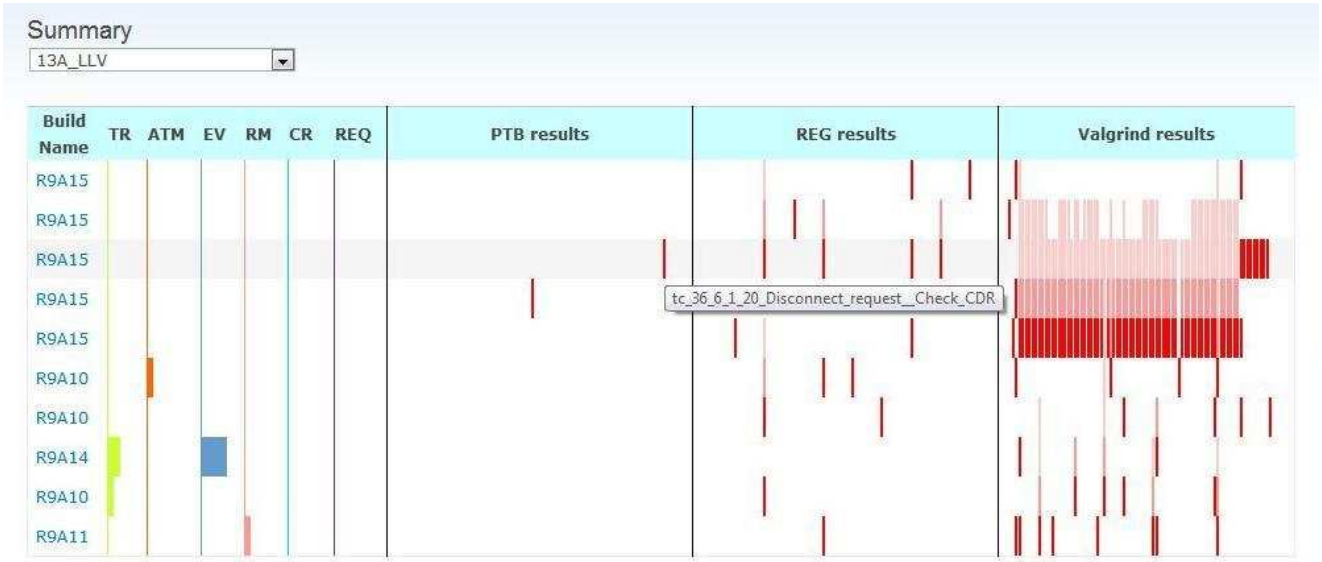

Fig. 5 The final implementation.

```
$verdicts = array(); foreach ($builds as &$build) { $buildId = $build->id;
 $verdicts[$buildId] = array(); foreach (getJobs($buildId) as $job) {
 $testCases = getJobTestCases($job->id); foreach ($testCases as $tc) { if
 (!$tc->isPassed()) { // save the verdict for this test case
 $verdicts[$buildId][$tc->getName()] = $tc->getVerdict(); } } } }
```

Fig. 6 Code retrieving failed test cases.

Along a column, the same test result is represented for different builds, which means there will be a red line if the test has failed and a white space otherwise. This was shown to be more readable than using green lines for passed results and red lines for failed results as in the paper prototype (Fig. 4). As a matter of fact, passed results are not displayed at all since our purpose is to focus on where and how the test cases failed. Indeed, in this case, the white space shows that the test case did not fail, but this does not necessarily mean that this test has passed. It was also found that displaying only failed test cases would make the visualization clearer in case of many tests failing. In order to achieve this, an array was created to store fetched test cases for each build, as seen in figure 6. For instance, $verdicts['build123']['tc_name'] is a test case from build123. If a test case has been failing for a specific build, its verdict (which is not passed) is stored in order to be displayed in the table.

In the case of many failed test cases, there will be a big red block under the corresponding test results (as seen for Valgrind results in figure 5). This is due to the fact that for each build, test results are represented by a fixed rectangle block to preserve the verticality that makes it possible for the user to compare the same test result for different builds, along the same column. Several color nuances are used when the same test case has failed for several times, red color is used when it fails for the first time and if it fails consequently for the following builds (up along the column), the color gets a lighter shade. The user can therefore have a quick overview of tests that have been failing build after build. This has been implemented by creating counters that would be updated depending on how many times in a row a test has failed. As seen in figure 8, the updating function updateCounters checks each verdicts and increases the counter in case of a failing verdict. For instance, if a test case fails for the first build, the counter is set to 1, if it fails on the next build, the counter is set to 2 etc. If this test case passes on the third build, it would be however be set back to 0. This enables users to see each time a test case is failing for the first time, even if it has been failing intermittently. Test case color would then be decided depending on the value of the counter.

As seen in figure 7, the shading is still visible with nuances of gray, with the goal of making the visualization not too color-dependent. One can also note that even if data related to deliverables is presented in different colors, the actual information is shown by the size of the rectangles and not their color.

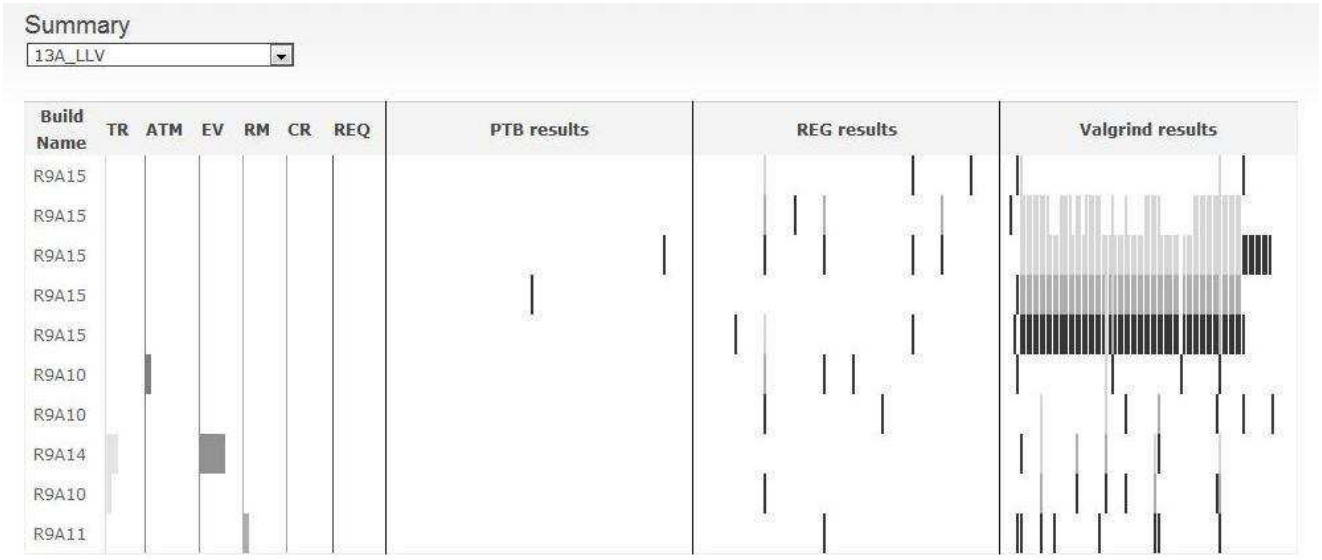

Fig. 7 The final implementation with nuances of gray.

```
function updateCounters(&$verdicts) { foreach (getUniqueLabels($verdicts) as
 $label) { $counter = 0; foreach ($verdicts as $build => &$build_verdicts) {
 if(array_key_exists($label, $build_verdicts) &&
 !$build_verdicts[$label]->isPassed()) { $counter++; // updates the array with
 a new counter value $build_verdicts[$label]->setTailCounter($counter); } else
 { $counter = 0; } } } }
```

Fig. 8 Function counting tests failing several times in a row.

More technical details about the implementation are presented in the appendix II in the form of a table displaying functions used in the code and their description.

4.3 Evaluation

This section presents results from the final evaluation that was conducted at the end of the project, as seen in figure 3. Most interviewees agreed that the visualization provides a good history of results and is easily understandable when associated with basic explanations. Before conducting the interviews, we presented the system and explained how the user could interact with the interface and get the information they need. This explanation could later be summarized in a small documentation document. Chosen shapes and colors have been considered to be meaningful but some improvements could be made. It has been suggested for instance to take into consideration color vision deficiency by using symbols or numbers. Showing how many times a test case has been failing would then be displayed in another way than with differences in color shading. Other comments proposed to remove information concerning zero deliverables (single lines) and collapse them into a narrow column would take less space and would be more useful. These lines could either be totally removed, or just hidden or extended according to the user's choice.

The visualization provides sufficient information according to designers, however interviewees carrying other positions have expressed the wish to display more data. Several comments suggested to add IPv6 information along with IPv4, to show whether a test case has failed with or without a trouble report or to include more software quality data such as Coverity in the visualization. Another recurrent suggestion was to provide the number of total test cases so that the user can see the ratio of failing test cases out of the total number of tests run. Some suggestions simply express the wish to access more detailed data, while providing trouble report information or total number of tests could significantly improve the interpretation the user can make of the interface. This could provide a more accurate comparison between builds, and make users understand better why a test case has been failing and if it has already been dealt with. There were also suggestions about displaying reasons of the test failures graphically. This could be done by using different colors to represent common reasons for tests failing. Displaying the slow running test cases could be helpful as well. Users argued that in this way they would be aware of test cases prevent-

Table 5 Positive and negative feedback from the evaluation.

Positive points	Possible improvements
Clear visualization of test runs and test result progress in a summarized form	Customization could be provided
Condensed overview of large amounts data in the same view	There should be a dotted line to represent lack of delivery
Useful data is presented	The shading should be pink for a new test case and red if it's been failing for a while
Good to visualize deliverables next to test cases	It would be good to visualize the total number of tests
Focus only on failed test cases	It would be better to connect each trouble reports to test results

ing on time delivery of the product. Different points of view were also gained about the way that test results are sorted. Some participants were satisfied with the current implementation which displays the results alphabetically. However, for some users (who deal with SGSN tests) sorting the data according to the test's start time is considered more helpful. It has also been suggested to sort the test cases based on their group (test objects) or their node roles.

Finally, some interviewees have suggested to provide more customization to the user. This idea has been suggested independently of the role occupied by the interviewee. A way to achieve this would be to implement the possibility to reduce or extend different views. Even though the focus on failing test cases was generally accepted by users, it has been noted that the white space displayed when test cases are not failing should be reduced, since it might provide confusion to the user. One might think this white space represents passed test cases, when it does not provide such information in reality. The white space can indicate that the test has not been run for this build. It would be interesting to display these non-run test cases as well, but it could be challenging to keep the visualization clear while providing such information. Other interviewees point out that it is best to avoid too much customization in order to keep a simple and clear picture. A compromise has been suggested where instead of implementing customization on the main page, the user could access another page with different views, displaying for example results from private builds.

Table 5 presents a summary of comments received during the final evaluation phase, divided between positive feedback and possible improvements.

Table 6 Investigated methods with their benefits and disadvantages.

Method	Pros	Cons
Thermometer	Intuitively understandable	Gets easily cluttered and can be difficult to read
	Clear presentation and separation of different kinds of data	Does not display data in a precise way
Speedometer	Provides detailed indicator's value	Provides only low precision
	Displays data status in an intuitive way	Restricted to a limited number of colors
Dashboard	Displays a large number of indicators	Complexity of use depending on displayed data types
	Provides a large range of colors	
	Provides detailed information about indicator's name and value	
Bullet graphs	Visible information even when scaled down	Displays too little information
	good for displaying small multiples	Requires too much space
		Can get cluttered with unnecessary information
Sparkline	Good for comparing values over time	Density of the chart can lead to a loss of details
Heat map	Can both be compact and present an important number of values	

5 Discussion

This section discusses the results from the prototyping, the implementation and the evaluation phases by answering the research questions investigated during the project.

What is a suitable method for visualizing large quantities of quality data generated in large scale software development projects?

1. *How current and historical data generated from different test types should be managed for presentation?*

   Table 6 shows a list of visualization methods considered at the beginning of the project. Despite their

advantages, the use of these methods could not result in a visualization that would fit the users' needs. The heat map method was however considered as an interesting visualization as it constitutes a good way to visualize complex and large volumes of data [14]. Our final result was inspired by this method and by interviews conducted with users. During these discussions, it has been suggested to provide a common shared view that different users could use regardless of their job position. This could be done by using several visualization methods as the ones described in table 6, but it would be then difficult to keep the visualization simple and clear. It was decided to try to condense different data types as much as possible, without making the visualization cluttered. This led to a visualization consisting of a historical summary showing the latest builds along with corresponding deliverables and test results. Displaying a historical summary enables different types of users to compare the latest builds between each other and to notice how data evolve build after build. By having an overview of data, users can spot instantly builds with a lot of failing test cases, and possibly relate these failing test results to the deliverables associated with the considered builds. Users can then access more detailed information by clicking the name of the corresponding build. Another feature that was brought by this visualization was the possibility to identify builds failing several times in a row. This makes it possible for users to notice tests that have been failing for a long time (in pink), tests failing for the first time (in red), as well as intermittent tests. All these types of tests provide important information since they might deserve special attention.

2. *What are the main requirements and quality attributes for the presentation methods?*

   The main attribute considered in order to improve the visualization methods is usability. Other attributes such as functionality, reliability or maintainability can be taken into account when evaluating software quality. However, in this study, the focus is on the visualization and its possible improvements of users' work. The goal is to improve an existing visualization method rather than to provide new functionalities. Because of time limitations, the performance and the portability of the implementation are not be taken into consideration during this study. ISO/IEC 9126-1 describes usability as based on "the effort need for use" by a certain set of users [31]. In our case, we tried to develop a system that would help users reduce their effort to access information. This was done by implementing an overview of data, rather than going too much into details and presenting a cluttered visualization. Presenting large amounts of data in the form of a historical summary can also help users to analyze and interpret the displayed information. Chosen shapes, colors and the hovering feature provide a support for usability as well. Previous studies such as the one conducted by Voinea et al. [32] have investigated visualization of large-scale software as a support for employees holding different positions. However, they do not combine different features such as an overview presentation, a historical summary and a common shared view. This combination was motivated by the fact that requirements would vary depending on the user's position. For instance, project managers were more interested in having an overall status of the product delivery. The most important data to show them was therefore to point out the obstacles preventing the online delivering of the project. However, requirements from a software designer and functional tester perspective included depicting detailed data in a way that they could keep track of the project deliverables and be aware of failed test cases along or not with trouble reports. This would allow them to fix errors on time and in a more efficient way. Color nuances were then applied in order to help them to follow the stream of test cases for each specific build. The selected attribute, as well as the requirements gathered by end users, inspired us to implement a visualization following the mantra proposed by Shneiderman: "overview first, zoom and filter, then details-on-demand" [7]. Indeed, our priority is to first present an overview of historical data, and then let the user access detailed information.

3. *What channels (size, shape, color, sound or movement) are suitable for presenting data to the users?*

   Suitable channels to achieve an intuitive representation of a fairly large amount of current and historical data include different sizes, shapes and color nuances. Among several sources used to select suitable channels for representation, the ranking of quantitative perceptual tasks by Mackinlay was studied in the early phases of the project. Results from this study show that certain tasks are carried out more accurately than others [6]. For instance, Mackinlay considers that position belongs to the "higher tasks", as seen in figure 1. In our project, position was taken into consideration by providing a vertical history when displaying results. Position can therefore be considered as a suitable channel for an intu-

itive representation of data. We also considered area perception, which is neither a higher nor a lower task according to Mackinlay [6]. In our implementation, we considered area as a channel to represent the number of deliverables. They are displayed as rectangles with different sizes depending on the number of deliverables associated with the build. Since this information is not the most important for the user (who is generally more interested in test results, and possibly their relation to deliverables), we do not consider it as a drawback, even though it is open to improvements. On the other hand, color density perception is considered as a "lower task" and was used in our implementation to display tests failing several times in a row. The use of color as an information channel can certainly be improved, as it can be difficult to interpret for users with color vision deficiency.

Which quality data should be visualized and how shall it be combined to increase the information throughput to the user?

1. *What type of data is the most important to extract?*

   According to the users, the most important information to display is data related to function test results, as well as deliverables such as trouble reports, requirements and change requests. The focus is made on function test results and the goal is to show when these tests fail in the most obvious and intuitive way possible. Certain types of tests such as unit tests and signal flow tests are not displayed since they should in principle never fail. The objective is then to select data that is the most important to visualize, since presenting all data available would result in a cluttered visualization. An important aspect to take into account when selecting data was the scalability of the system. Previous studies have shown the importance of taking this attribute into consideration when implementing a new system, or improving an existing one. For instance, when implementing an evolution matrix, it was found that the system worked for a limited number of classes [33]. Similarly, in the first steps of our project, the visualization displayed both passed and failed test cases and would become very large in case of many failing test cases. Such visualization would make data harder to both summarize and interpret. There was then a choice to focus only on failed test cases, which was proven to be generally accepted by end users.

2. *How can we combine data of different dimensions?*

   In order to present data efficiently, time and test type dimensions were combined in this visualization method. Time dimension refers to test data that is available for a specific time, test data could then be summarized for time period corresponding to the latest builds. This was used in the visualization when providing a history of builds over time. Test type dimension concerns different test types which are accomplished before project deliveries, it starts with low level testing (unit tests) and proceeds to function and system levels. In the visualization, considered test types are function test results (REG and PTB) and memory errors (Valgrind). According to results from studies, prototypes and interviews, the selected data should provide enough interesting information without making the visualization cluttered or complex.

3. *How can data be translated into information channels?*

   Once specific data was selected to be displayed, different ways were investigated in order to translate this data into information channels. First prototypes displayed deliverables with letters but did not provide information about how many were associated to each build. In the implementation, several columns are displayed for each deliverable types. Information about the number of deliverables for a specific type is shown by rectangles of different sizes, as well as single lines when nothing is delivered. By hovering, the user can have a more detailed list of deliverable names and their exact number. The size of rectangles and the hovering function are the channels used to provide information about the number and names of deliverables. As for test results, data concerning failing test case is translated by single red lines for each failing test. By hovering, the user can see the name of the test that failed. Color nuances are also used as an information channel to show when a test has been failing several times in a row.

How does the visualization method effect the user's communication about the product?

1. *What benefits will be seen by users?*

   The last interview findings indicate that the visualization of software quality data has been improved in the implementation. The visualization could meet project managers' requirements since it provides a suitable historical summary of the quality data. They can use the visualization as an overview tool by quickly noticing builds having a lot of failed test cases. Designers and testers can, on the other hand,

access more detailed information such as how long a test case has been failing, when it is failing for the first time and which test cases are failing intermittently. Designers, in particular, found it to be a useful tool for tracking the results related to test cases and in this way they could efficiently be aware of the test cases preventing the product delivery, this being especially important when multiple teams are performing activities in the same area. It can be argued that without this knowledge, it would be difficult for developers to optimize their development efforts.

It has been suggested to take into consideration color vision deficiency by not using colors but symbols and numbers instead. The trade-off of this is that it would be more difficult for the user to get an overview of data. Indeed, we believe that the chosen shapes and colors can provide a visualization that is intuitive and easy to understand. Provided with basic explanations about how the visualization works, the user can quickly extract the needed information. However, we believe that it is important to consider color vision deficiency and we tried to take it into consideration by using a visualization as clear as possible that does not only rely on colors. There are still improvements to be made, and it is certainly possible to combine symbols, numbers and colors and still provide a clear and simple visualization.

There have been several suggestions concerning the use of space in the visualization. For instance, it was suggested to narrow unnecessary information such as lines showing zero deliverables. It has also been proposed to limit the white space displayed when there are few failing test cases. Reducing this space could be a way to make the visualization simpler and to gain space but the verticality function would then not be preserved. Indeed, a red line that turns into a white space shows that this test is not failing anymore, even though it does not give information about whether the test has passed or not. Such information could be interesting to display, but in our case, the choice was to focus on failing test cases, and to point out when a test case has been failing several times in a row or in an intermittent manner.

Differences of opinions about how the visualization should be could be solved by providing customization. This would allow users to access either different possible views or a dynamic page where they could extend or condense information according to whether they are interested in it or not. It could also provide a solution for sorting the data in different ways (either time based or name based for instance). Customization certainly provides an interesting improvement of the visualization, but it can also compromise the wish to keep the presentation as simple and intuitive as possible. A good compromise would be to access both an overview page, as it is proposed in this paper, and a more personal page corresponding to the user's role and responsibility. In all cases, it is important to keep a shared common overview of data, in order to provide interaction between users. In this way, users holding different positions can then efficiently communicate since they have access to the same view and therefore the same data.

Comments suggesting to display more data seem easier to implement as they do not interfere with the actual visualization method. This constitutes an interesting challenge to the portal, as the visualization must stay simple and not too cluttered. Data should also be selected so that it provides interesting information to users regardless of their roles. According to interviews, this might not be the case with Coverity data for instance, but displaying IPv6 results along with IPv4 ones, and showing test cases failing with or without trouble reports would definitely be valuable to different types of users. Sorting results in a specific manner is also depending on users' roles and might not be relevant for an overview interface. Alphabetical order was chosen in order to provide a simple common visualization, the focus being on keeping a historical comparison of results and not visualizing them in a particular order. Recurring comments suggested to display the total number of test cases. This would constitute interesting information to different types of users, but it can compromise the focus on failing test cases, which constitutes one of the requirements of our product. However, like other suggestions being expressed by interviewees holding different positions, they are important to consider for future improvements.

2. *How can the visualization improve the user's work's efficiency?*

   By accessing an overview page displaying information about various data types, users do not need to navigate between different pages when different types of data are needed. It also provides a possibility for comparing and relating builds, deliverables and test results to each other. Specific data was selected and displayed according to users' needs. This can reduce the effort to retrieve the interesting information to users. Furthermore, the visualization can point out builds with several failing test cases and other situations that might require quick and spe-

cial attention. By providing a history of results, it also can help users noticing patterns and the evolution of the results build after build. Due to a limited time frame, it was not possible to actually measure the impact of the visualization on the users' work. This could be done by conducting interviews once the users have used the visualization for a certain time. One could also notice if there have been improvements in decision making, in the planning and making of projects. The effect of the visualization on the users' interaction could be shown by the use of the visualization in communication, during group meetings for instance. These possible suggestions for measure work's efficiency would however require the use of the visualization for an extended time. For now, we can only rely on the final evaluation and feedback from the users to state whether the visualization could improve their work's efficiency. Previous studies such as the one conducted by Voinea et al. [32] have investigated visualization of large-scale software and related it to the quality of work of employees holding different positions. However, in this work, an overview of data is provided along with the possibility for users to track results over time.

6 Conclusion

The purpose of this study was to improve the current visualization of software quality data at the company by proposing new visualization methods. The study began with investigating web based methods for visualizing large amount of data with a focus on intuitive user presentation. Several methods such as dashboards, pie charts and bar graphs were investigated during the research and prototyping phase. These methods provide a way to display information intuitively. However, their disadvantages were considered as obstacles in our goal for a simple visualization that could be shared by different types of users. By conducting several interviews, important data from different types of users' point of views was extracted. First, we were able to set up requirements and a more limited scope to our project, which enabled us to design more accurate prototypes. We then presented these prototypes to users during a first set of interviews, in order to get improvements and more details about how the visualization should be. After designing several prototypes, we concluded that illustrating data in a table which acts as a timeline (showing the latest changes in builds and test results) would be a suitable way to visualize this type of data in an intuitive and effective manner.

This visualization provided both a history and an overview of results by presenting build information on a horizontal line and failing test cases as vertical red lines. If a test case has been failing several times in a row, it is displayed as a longer continuous vertical line with a fading color shade. Otherwise, it turns into a white space, which can turn again into a red line, showing that the test has been failing intermittently. Additional data about the build such as type and number of deliverables are displayed next to test results.

This visualization was implemented using the same tools used for the current portal implementation, that is PHP, MySQL, HTML, CSS and the Drupal framework. We believe that this visualization can give several advantages for users, such as providing, in a common shared view, sufficient information to designers, testers and project managers. By using suitable applied shapes and colors, we wanted to make the visualization understandable for all these types of users. Whether this goal was reached or not was measured by interviewing users at several steps of the project. While the results from the first interviews were to be applied in the prototyping and implementation phase, the second set of interviews consisted more of an evaluation of the final work. This evaluation led to diverse and interesting suggestions such as providing more customization, displaying total number of test cases and connecting test results to trouble reports. These suggestions could be used in the future to improve this way of visualization. The findings can be generalized to other products within and outside the company. Any software project can indeed benefit from an overall summary of software quality data, in order to notice quickly obstacles preventing the right progress of the project. Previous studies investigated several ways to visualize complex and large volumes of data and proposed various efficient visualization methods. There is however a lack of industrial studies investigating how visualization can be help the users in their work. The result of this study provides an overview of data, a history of results and different data types in a shared common view that can facilitate interaction between users. We believe that this project can therefore contribute to the current state-of-the-art and provide an industrial example of how to display data in an intuitive manner while aiming at improving work's efficiency.

References

1. Pandazo, K., Shollo, A., Staron, M., Meding, W.: Presenting Software Metrics Indicators: A Case Study. Proceedings of the 20th International Conference on Software Product and Process Measurement (MENSURA), Vol. 20, No. 1, (2010).
2. Johansson, L., Meding, W., Staron, M.: An Industrial Case Study on Visualization of Dependencies between Software

Measurements. 7th Conference on Software Engineering Research and Practice in Sweden , Vol. 1 (February 2007), pp. 23-33, ISBN/ISSN: 1654-4870.
3. Reniers, D., Voinea, L., Ersoy, O., and Telea, A.: The solid* toolset for software visual analytics of program structure and metrics comprehension: From research prototype to product. Science of Computer Programming, 2012.
4. Cleveland, W.S., McGill, R.: Graphical Perception: Theory, Experimentation and Application to the Development Graphical Methods. Journal of the American Statistical Association, Vol. 79, No. 387. (September, 1984), pp. 531-554.
5. Buse, R. P., Zimmermann, T.: Information needs for software development analytics, Proceedings of the 34th International Conference on Software Engineering (June, 2012), pp. 987-996, ISBN: 978-1-4673-1067-3.
6. Mackinlay, J.: Automating the Design of Graphical Presentations of Relational Information, Journal of ACM Transactions on Graphics (TOG), Vol. 5, No. 2, (April, 1986), pp. 110-141.
7. Shneiderman, B.: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations,.Visual Languages Proceedings., IEEE Symposium (September, 1996), pp. 336- 343.
8. Rogowitz, B.E., Treinish, L.A., How NOT to Lie with Visualization, Journal of Computers in Physics, American Institute of Physics Inc. Woodbury, NY, USA, Vol. 10, No. 3., (June, 1996), pp. 268-273, ISBN: 0-8186-7508-X.
9. Healey, C.G., Booth, K.S., Enns, J.T.: High-Speed Visual Estimation Using Preattentive Processing. Dept. of Comput. Sci., Maryland Univ., College Park, MD, (June 1996), Vol. 3, No. 2, pp. 107 135, ISBN:1049-2615.
10. MT-35. Alette, T., Fritzon, V.: Introducing Product and Process Visualizations to Support Software Development, Master of Science thesis in Software Engineering and Technology, Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden (June, 2012).
11. Few, S.: Business objects' bullet graphs: A good idea, implemented poorly. [Online]. Available: http://www.perceptualedge.com/blog/?p=160
12. Tufte, S.: Beautiful Evidence: A Journey through the Mind of Edward Tufte, 1st ed. Graphics Press, (July 2006), ISBN:0961392177.
13. Davenport, T., Harris, J., Morison, R.: Analytics at Work: Smarter Decisions, Better Results, ser. Harvard Business School Press. Harvard Business Press (2010), ISBN: 1422177696.
14. Feldt, R., Staron, M., Hult, E. and Liljegren, T.: Using Heatmaps to Visualize Test Outcomes and Source Code Changes: A Case Study at RUAG Space, 2012. (submitted 2012)
15. Ball, T. J., Eick, S. G.: Software Visualization in the Large, IEEE Computer Society Press Los Alamitos, CA, USA,(April 1996). Vol. 29, No. 4, pp. 33-43.
16. Ko, A. J., DeLine, R., Venolia, G.: Information needs in collocated software development teams, Proceedings of the 29th international conference on Software Engineering, ser. ICSE '07. Washington, DC, USA: IEEE Computer Society (May 2007), pp. 344353, ISBN: 0-7695-2828-7.
17. J. Jones, M. Harrold, and J. Stasko. Visualization of test information to assist fault localization. In Proceedings of the 24th international conference on Software engineering, pages 467477. ACM, 2002.
18. Ericsson company facts. [Online]. Available: http://www.ericsson.com/thecompany/company_facts
19. Haagen, J.: Master Thesis-Visualization of Software Quality Trend: Master thesis proposal at Ericsson AB (2011).
20. Collins, A., Joseph, D. and Bielaczyc, K.: Design research: Theoretical and methodological issues, The Journal of the learning sciences,Taylor & Francis publisher, Vol. 13, No.1, pp. 15-42, (2004)
21. Grim, B.J., Harmon, A., Gromis, J.: Focused Group Interviews as an Innovative QuantiQualitative Methodology (QQM): Integrating Quantitative Elements into a Qualitative Methodology, Vol. 11, No. 3. (September 2006), pp. 516- 537.
22. Merton, R. K., Kendall, P. L.: The focused interview. P. F. Lazarsfeld & M. Rosenberg (Eds.) 2nd edition, The language of social research, pp. 476-489. New York: The Free Press. (1955), ISBN: 0029209862.
23. Beiley, B, Biehl, J.,Cook, D., Metcalf, H.: Adapting paper prototyping for designing user interfaces for multiple display environments, published by Springer (2008), Vol. 12, No.3 (January 2008), pp. 269-277.
24. Staron, M., Meding, W.: Using Models to Develop Measurement Systems: A Method and Its Industrial Use. IWSM '09 /Mensura '09 Proceedings of the International Conferences on Software Process and Product Measurement, Vol. 5891, No. 3 (2009), pp. 212-226, ISBN: 978-3-642-05414-3.
25. Staron, M., Meding, W., Nilsson, C.: A framework for developing measurement systems and its industrial evaluation. Journal of Information and Software Technology, published by Elsevier, Vol. 51. (April 2009), pp. 721-737.
26. Runeson, P., Hst, M.: Guidelines for conducting and reporting case study research in software engineering. Department Computer Science, Lund University, published by Springer, Vol. 14, No. 2, (December 2008), pp. 131-164.
27. Yin, R.K.: Case study research. Design and methods, 3rd edn. London, Sage, (December 24, 2002), ISBN: 0761925538.
28. About Drupal. [Online]. Available: http://drupal.org/about
29. Drupal principles. [Online]. Available: http://drupal.org/principles
30. Object Relational Mapper, Doctrine. [Online]. Available: http://www.doctrineproject. org/projects/orm.html
31. ISO/IEC 9126-1:2001 Software engineering – Product quality – Part 1: Quality model.
32. Voinea, L., Lukkien, J., and Telea, A.: Visual assessment of software evolution. Science of Computer Programming, 65(3):222248, 2007.
33. Lanza, M.: The Evolution Matrix: Recovering Software Evolution using Software Visualization Techniques. IWPSE '01 Proceedings of the 4th International Workshop on Principles of Software Evolution, New York, NY, USA, (2001), pp. 3742, ISBN:1-58113-508-4.

Appendix I – Interview questions

Theme	First interview questions
GUI Design	What is your idea about the shape, color and arrows applied in test results? Do you think they are meaningful? Would it be useful to add more color nuances and an indicator of how close to one color the test result is? Would it be better to present raw data and let users interpret it? Would you rather see different personalized views or avoid too much customization?
Usability	Are the builds and test results (test cases) easily traceable? If not, why? Can you easily get an overview of the status of the project? How easy is it to find the detailed value of indicators? Can you easily access a specific type of test? How easy is it for you to access different types of data you need? Is there any feature that is too complicated to handle/understand?
Functionality	Do you think the status of the project should be more clearly presented? Would it be useful to show that the status is about to change? Would it be useful to have a notification of failed results, and what kind of notification would be preferred? Would it be useful to visualize the source code? Would it be better to focus on the failed results in the presentation? Would it be better to filter the results to present them in a specific order, or to display only some of them? Is there some information that is not so important or needed? (“nice to have”) Would it be interesting to be able to compare different results and data? Would you like to see some data grouped together or on the contrary strictly separated?
Efficiency	How well does this way of presentation fulfill your information needs? How do you think the presentation of the portal can be improved to increase work efficiency? Are there some features or displays that are obstacles to a quick access to information? (slow down the work)
Conclusion	Do you find the portal user-friendly? Why? Is the information clear, unambiguous and simple? Could you describe the negative and positive points with the integration portal? What are your expectations (requirements) for this product? What stops you from working efficiently?

Themes	Summarized interview questions
Introduction	Is this way of visualization clear to you? Do you think it will be helpful in your job (designer or tester)?
GUI Design	What is your idea about the applied shape and colors in the results? Do you think they are meaningful? Would you be interested in having more raw data displayed? Would you rather see different personalized views (filters) or avoid too much customization? Is it interesting to have data in alphabetical order?
Usability	Is there any feature that is too complicated to handle/understand? Is this data easily traceable?
Functionality	Is it good that only failed results have been focused in presenting the product status or would it be better to display passed results as well? Is there any information that is not so important to display in this presentation? Would it be interesting to be able to compare other kind of results and data?
Efficiency	How well does this way of presentation fulfill your information needs? How do you think the new presentation can be improved to increase work efficiency? Are there some features or displays that are obstacles to a quick access to information? (slow down the work)
Conclusion	Could you describe the negative and positive points with the new presentation? What are your expectations about the new presentation?

Appendix II – Implementation details

This table gives an overview of the implementation design by presenting functions used to retrieve and display data.

Function name	Arguments	Description
generateDeliverables	Array of events	Generates html to display deliverables in the form of rectangles.
generateRectangle	Array of test result verdicts Array of test names Width of the generated rectangle Width of the failed result line	Generates html to display test results in the form of vertical lines.
getAllLabels	Array of test result verdicts Whether to sort the array or not	Return an array containing all unique test names.
notEmpty	Array of (failed) test cases	Filter function that checks if a build have any failed test cases or not.
printDeliverables	Number of deliverables List of deliverable names Color of the rectangle Html attributes of the rectangle	Echos the html code for a rectangle displaying information about deliverables.
renderSelectBuild	Name of the current product	Echos the html code for the drop-down menu where the user can select a product name. If a current product is passed, it is selected by default.
updateCounters	A sorted array of verdicts	Update counters of how many times in a row a test has failed and returns a set of test names. (see getAllLabels)
verdictItem	The name of a test result	Returns a lightweight data structure representing a failed verdict.