Software Test Data Visualization with Heatmaps – an Initial Survey

Per Erik Strandberg

Westermo Research and Development, AB
Email: per.strandberg@westermo.se

Abstract

A visualization of nightly regression test results provides rapid feedback, and may aid in making decisions on software projects. However, visualizing test results from sparse and distributed nightly regression testing is non-trivial. We have conducted a structured literature review focusing on how heatmaps are best used for software test data visualization. By identifying existing studies, we have discovered how stakeholders use the visualizations, how they are valuable, the approaches used for plotting and the proposed future research. The review identified four papers that describe visualizations that target managers, experts, testers and code maintainers. The visualizations are used for making decisions, giving support, or provide early warnings. Typically the source code of the software under test is an important part of the plots.

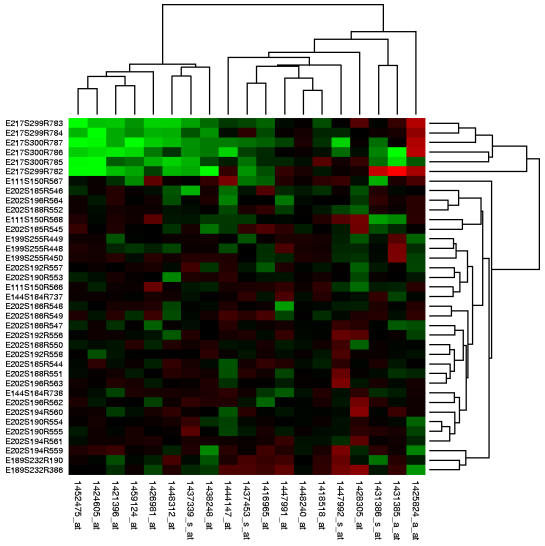

Figure 1: Example of a heatmap that illustrate results from an affinity experiment where different genes and proteins bind. Dark areas indicate a low interaction. This image is also suitable for illustrating a common challenge for the color blind: they can often not distinguish the red areas from the green areas in this plot. Image from [25], in public domain.

1 Introduction

Visualizing source code properties such as structure and code churn (defined in 2.2) are well studied practices. Approaches where these characteristics can be seen over time have also been studied [11].

Westermo Research and Development AB (Westermo) design hardware and software for robust industrial communication devices. In order to provide rapid feedback for ongoing software development these devices are tested nightly for regressions using a test framework that has been developed and supported over several years. The nightly testing is performed on a number of test systems, each with several devices in a network topology.

This nightly testing generates a lot of test results data. This data is sparse: not every test is tested every night. The data is also multidimensional: a test outcome is tied not only to a test case and a point in time, but also to one of many code branches under test, and to one of many physical test systems. The software solution implemented to select test cases for the nightly regression test suites, SuiteBuilder, is described in [21].

A heatmap can be defined as a graphical representation of data. A two-dimensional matrix can be represented with colored rectangles for each element, given the values in the matrix. In Figure 1 affinity between genes and proteins is illustrated with colors close to black as no or low affinity. Other colors indicate other types of affinity. Heatmaps have been in use since 1957. The term was coined in 1991. [25].

At Westermo we have implemented several prototype visualizations of the nightly test results using techniques similar to heatmaps. But these are under frequent discussion with issues such as poor maintainability, low usability for the color blind, and missing features such as the typical “back” button in a web browser.

In this systematic literature review (SLR), or survey, we investigate literature on visualizations of test result data *using heatmaps*, with or without source code characteristics.

The contributions of this paper are all related to summarizing existing research: (i) Existing research on software test result heatmaps indicate that these are used to support test planning, for warning systems, and to aid in making other decisions. (ii) Existing research indicates that these visualizations target senior developers,

project managers, test managers or testers. (iii) Existing research has to a large extent been evaluated in cooperation with industry practitioners and large free, libre or open source software (FLOSS) projects. (iv) Data used for the visualizations are version control system (VCS) data from both the software under test and the test framework, historical test data, code coverage data and data from issue trackers. (v) Future work mentioned in the literature includes: increasing the scale, improving the evaluation, improving usability, and incorporating automated warnings.

In Section 2 we introduce related work. Section 3 describes the survey method used. Section 4 describes the findings of the survey and in Section 5 we discuss these findings. The paper is concluded in Section 6. Summarized review protocols are presented in Appendix A.

2 Related Work

In this section we discuss previous research related to visualization of test data.

2.1 Visualization of Source Code

Source code can be hard to grasp. Visualizing it can be done in many different ways. An influential publication in this field is [4], from 1992, where source code is shown from a helicopter view as thin colored lines. The colors represent an attribute of the code, such as age.

Several surveys on software visualization have been published, a recent example is Caserta and Zendra [2]. Here the authors present methods for visualizing static aspects with the focus of source code lines, class metrics, relationships, and architectural metrics. Six methods for visualizing evolution are also presented. Methods include metaphors like cities (with skyscrapers for large software units, flatter building for smaller ones, and links illustrating code dependencies) and also more known approaches such as UML diagrams.

2.2 Software Fault Prediction

An influential publication where test results data is used is [12], from 2002, where a combination of source code, code coverage, unit tests and test results is illustrated as a sort of extended Integrated Development Environment (IDE) view. This publication does not mention heatmaps and is not included as a relevant publication in this survey.

Nagappan and Ball defined code churn in 2005, [14], as: *"Code churn is a measure of the amount of code change taking place within a software unit over time"*. They use it as a predictor of defects. Another, widely cited, paper by Ostrand et al, [17], uses source code data combined with data from other sources, such as issue trackers, to predict the number and location of faults in software units. By using code churn and other characteristics software faults can be predicted with very good accuracy.

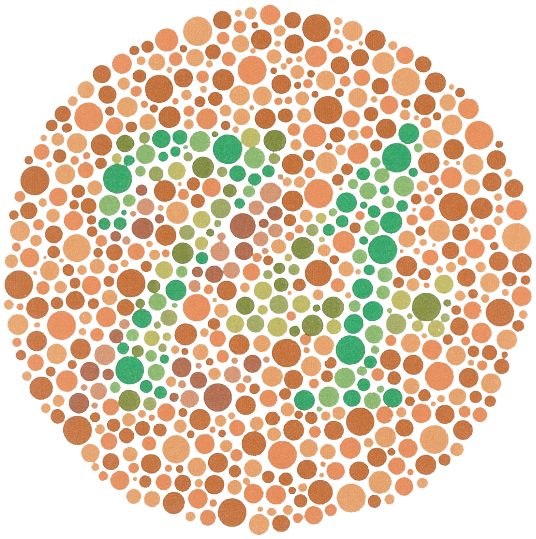

Figure 2: Example of a color test plate from the Ishihara test, a color perception test for red-green color deficiencies. This test plate is designed to show the number 74 for individuals with normal color vision and the number 21, or nothing, for individuals with various types of color blindness. Image from [24] in public domain.

2.3 Software Evolution

The need to visualize changes in source code has lead to attempts to stabilize the layout of source code units. One promising and recent approach is presented in [11], where the authors present a proof of concept for visualizing changes in cyclomatic complexity over time in a chart that also visualizes source code structure, with a more deterministic positioning of code components.

2.4 Dashboard Visualization

A common approach for continuous quality monitoring and control in the industry is having a "dashboard", [3]. These aim at presenting one or more key performance indicator (KPI) over time. An recent paper with lessons learned is presented in [9].

2.5 Color blindness, Ishihara, and Usability

Color blindness is a condition where an individual cannot fully see color or differences in color. It was first described by John Dalton in 1798. Roughly 8% of all men have some form of color blindness. [24]

When a difference in color is used to carry information there is a risk that color blind individuals cannot receive the information, or receives distorted information. This is used in the classic Ishihara test, where a number is hidden in a colorful dotted pattern. The test was designed to detect color blindness. Figure 2 illustrates a color test plate where color blindness leads to distorted information, in this case the number 21 is seen instead of the correct number 74. [26]

The scope for this paper is not on color blindness. However, many users of the current implementation at Westermo are color blind. Zeileis et al phrases it as follows when discussing colors for statistical graphics: "different areas of a plot should still be distinguishable when the graphic is displayed on an LCD projector rather than a computer screen, or when it is printed on a grayscale printer, or when the person viewing the graphic is color-blind." [23]

2.6 Two Reference Papers

Prior to this survey two publications on software test data visualization with heatmaps were known to us. One was written by Feldt et al in 2013, [8], and the second by Engström et al, [7] in 2014. In this paper we refer to these two publications as the two "reference papers".

These papers propose visualizations of test results data. Feldt et al use a heatmap where test results over time and over test cases are used. This is to be combined with a heatmap showing code churn over time, a simplified illustration is shown in Figure 4. Engström et al propose a heatmap where the structure of the source code is displayed in a tile-like pattern where the tiles are colored based on test results, see Figure 3.

2.7 Test Results Visualization Surveys

However, to the best of our knowledge there are no existing surveys on visualization techniques for software test data that explicitly covers *heatmaps*.

In this initial survey we wanted to discover a wider range of scientific publications on visualizing regression test results with heatmaps. We knew about two reference papers but wanted a broader knowledge of the field.

3 Survey Method

Our goal in this study is to follow the guidelines for conducting a SLR proposed by Kitchenham et al in [13]. The coming sections describe the survey phases: planning, conducting and notable events.

3.1 Planning the Survey

In order for this survey to have a relevance to practitioners in the industry we chose the following research questions (RQs) that we also motivate and explain:

RQ1: *What studies have been performed on heatmaps in a software testing context?* We want to know about other publications than the two reference papers. What Universities and/or companies have been involved in this research?

RQ2: *Who are the involved stakeholders?* What roles are these visualizations targeting. Our initial speculation is that test managers and/or project managers are interested in monitoring progress and in allocating test resources for further testing. Another approach that can be considered is that software developers, test framework developers and test case developers want to monitor the progress of the code or tests. We speculate that top level managers and customers are not a typical target audience, but we would gladly incorporate results from any such publication in this survey.

RQ3: *How are heatmaps a valuable tool for practitioners in the industry?* This question might be sensitive: If we consider the possibility that heatmaps have no or very little value for practitioners in the industry, then there is still a possibility that publications portray this field of research as very promising for reasons other than the public good: perhaps to reassure continued research funding. We speculate that an evaluation with an industry partner indicates that an approaches has *actual* value to practitioners.

RQ4: *What approaches to visualization have been investigated?* We want to know about possible data sources used in the visualizations. What possible dimension on the axes has been explored?

Questions regarding implementation techniques have been omitted, for example: *"Is the Python library X more or less suitable than JavaScript library Y for a visualization of Z?"*.

The structure of the test results data is also ignored in this initial study. The data structures used is an important topic that may have a direct impact on the possibilities for visualization since without adequate data for X we cannot visualize X properly. We consider these questions to be out of scope for this survey.

A protocol was designed in order to capture the relevant aspects of the publications, and to not miss any information needed to answer the research questions. The headings of this protocol and examples are presented in the following paragraphs.

Affiliation Universities, or the like. Examples: (i) University of Zurich, or (ii) Ericsson AB

Focus or Research Question What is the focus of the paper? Example: Identify early warning indicators about potential problems with unexpected cost.

Data sources What data is used in the visualization? Examples: (i) Version control system (VCS) information, or source code, (ii) Data from issue trackers.

Stakeholder Recipient or user of the visualization. Examples: (i) Senior Developer, (ii) Test Manager, (iii) Project Manager, and so on.

What is Visualized? What is at focus in the visualization(s) used in the publication? Example: code churn per source code component over time.

How is this valuable? Why is this visualization valuable to the stakeholders? Example: Support in test planning.

Environment In what type of environment is this study conducted? In particular: are there any industrial

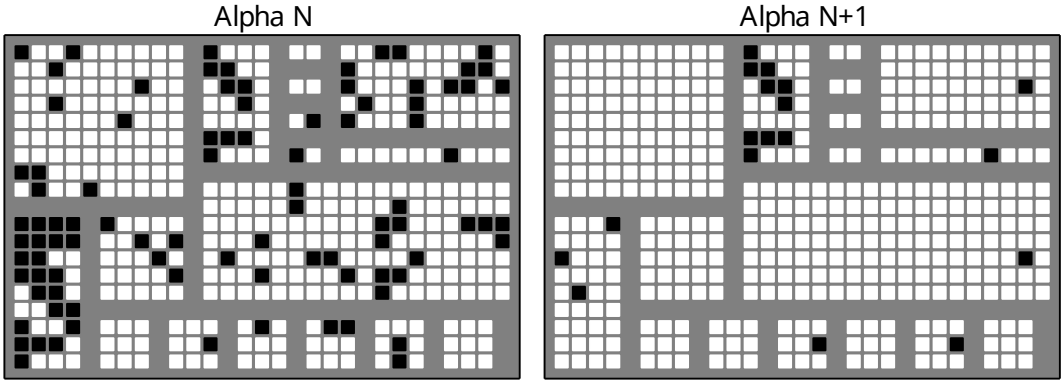

Figure 3: Visualization of test coverage items in two releases, adapted from Engström et al, [7], with colors omitted for clarity. The left and right structures both illustrates test coverage items (source code structure) with many (left) or few (right) test failures.

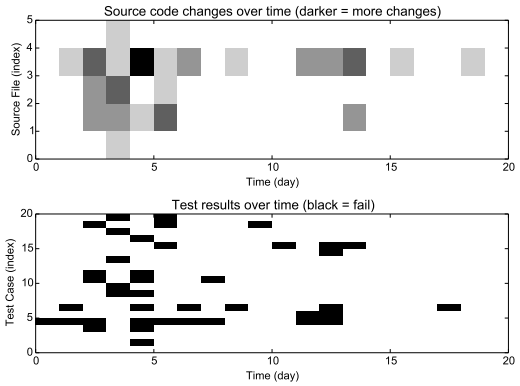

Figure 4: Illustration of a visualization of code changes in source code (top), and test results over time (bottom). From Feldt et al, [8], with colors omitted for clarity.

partners? Examples: (i) Evaluation with data from FLOSS projects. (ii) Evaluated with two separate industrial partners.

Other Comments Examples: (i) Does not include test results data. (ii) First two words in abstract are "test managers".

3.2 Conducting the Survey

The stages involved in this study are (1) inclusion based on search terms, (2) exclusion based on search terms, (3) exclusion of duplicates, (4) exclusion based on title, (5) exclusion based on abstract, (6) exclusion based on availability, and (7) exclusion based on the entire publication. The rest of this section describes these phases in detail, an overview is presented in Table 1.

Several electronic databases exist for supporting structured review papers. Notable examples include: Inspec, Compendex, ACM Digital Library, IEEE Xplore, ScienceDirect, Springer LNCS and Web of Science. Due to time constraints we have only investigated two databases: IEEE Xplore[1] and ACM Digital Library[2].

In order to satisfy our research questions we used four search terms: (i) *visuali** in order to limit the search to papers that explicitly mention visualization, (ii) *test** and (iii) *software* were added to limit the search to papers on software testing. Finally (iv) *heatmap** was added with *"heat map*"* as an alternative spelling. One of the reference papers did not occur in the search results unless the final term had an asterisk, this was due to the use of "heatmaps" as opposed to "heatmap". These terms yielded 368 publications. To handle the many papers, and to support the review process, we used spread sheets exported from the databases.

An initial look at the title of the papers revealed that many papers were obvious false positives. This leads to a set of excluding search terms: *patient*, *climate* and *biolog**. This excluded 82 papers.

Given the terms for inclusion and exclusion we found a total of 286 candidate papers. Four duplicates were removed from this set. Papers were now excluded based on reading nothing but their title. Publications with titles such as "Dolphin Detection and Tracking" were among the excluded ones. This reduced the number of candidates to 54.

The abstracts of the remaining publications were now read in order to only keep relevant publications. Another 37 papers were excluded. A surprisingly large amount of these dealt with experiments involving eye tracking. We speculate that this might be related to how eye tracking experiments are performed: perhaps a *software* layout is under evaluation, it is *tested* with real people, and the results *visualized* with a *heatmap*. If this is the case then there is a close mapping to the key terms we were interested in.

Of the seventeen papers that remained we were unable to find five in electronic format. These five were all from 1990 or older so these were also excluded. The twelve papers that remained were fully read and each got a survey protocol, as described in Appendix A. Af-

[1] http://ieeexplore.ieee.org/
[2] http://dl.acm.org/

Table 1: Phases of inclusion and exclusion.

Step	Activity	Criteria	IEEE	ACM	tot.
1	inclusion	terms	269	99	368
2	exclusion	terms	189	97	286
3	exclusion	duplicates	189	93	282
4	exclusion	title	24	30	54
5	exclusion	abstract	9	8	17
6	exclusion	availability	9	3	12
7	exclusion	full text	3	1	4

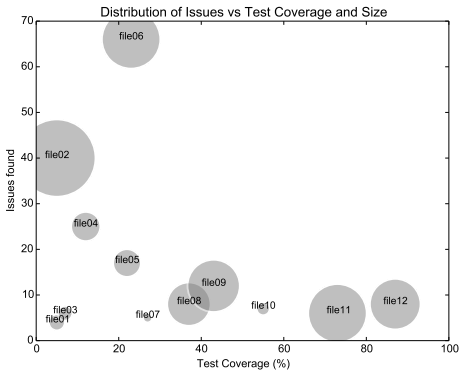

Figure 5: Illustration of a visualization of test coverage, issues found and size of source files. From Haron and Syed-Mohamad, [10], with colors omitted for clarity.

ter reading the complete papers only four relevant publications remained, two of these were the reference papers, the third one was a publication by Haron and Syed-Mohamad, [10], and the fourth one by Orso et al [16].

3.3 Notable Events during the Survey

Two papers were known a priori, [7, 8]. These were used as an evaluation of the criteria for inclusion and exclusion.

During the reading of the full papers we sometimes rapidly discovered that papers were obviously not relevant. This could be understood even after only reading the first page of a paper, or even after reading just the introduction. For future studies we would like to propose an additional filter before reading the entire paper: exclusion based on first page.

4 Results

4.1 Overview

The databases investigated and the method used revealed four relevant publications. These were published in 2003, 2013, 2014 and 2015. Two of them have had a clear industrial focus and were also evaluated with industry partners, whereas the other two were evaluated with FLOSS projects or with a limited set of individuals in the academia. All publications included in this survey have been used for visualizations of large software systems.

Two of the publications are from Universities in the south of Sweden and since a few of the authors of these papers have also co-authored other publications it is tempting to assume that they have somehow exchanged ideas or had an informal collaboration when conducting these studies. The other two publications are from Georgia Institute of Technology and Universiti Sains Malaysia. We think of these geographically separated areas as three different research clusters.

Three of the publications have a clear goal of supporting activities – this is made clear by their titles: "Supporting software decision meetings...", "Supporting regression test scoping..." and "...assessment of software test adequacy" respectively. The fourth publication aims at locating issues in the code based on customer usage. In a strict sense they are not using test data, but they specifically mention that their approach is a generalization of another approach where test-case information is used for fault localization, so for this reason it is included in this study.

It appears as if two main visualization approaches have been used: either the source code structure is used, or evolution over time is used. Compare for example Figure 3 that uses structure, with Figure 4 that uses time.

Only Orso et al mentions color blindness, but did not let the possibility of having color blind stakeholders affect their choice of colors: *"we could use other ranges of the color spectrum"*, but they have not designed or evaluated the spectrum they use with respect to color blindness, nor has any of the other papers in the study done so. We speculate that poor choice of colors in combination with color blind staff renders parts of the visualization unusable.

4.2 Answers to Research Questions

This section answers the research questions identified and summarize the future proposed by the studies investigated.

RQ1: What studies have been performed on heatmaps in a software testing context? We have identified four publications: Orso et al 2003 [16], Feldt et al 2013 [8], Engström et al 2014 [7], and finally Haron and Syed-Mohamad 2015 [10].

RQ2: Who are the involved stakeholders? We have identified four groups: (i) managers such as test managers, project managers and also the more generic group "managers", (ii) experts such as technical leaders or test experts, (iii) testers, and (iv) code maintainers.

RQ3: How are heatmaps a valuable tool for practitioners in the Industry? Value, or rather use cases, mentioned in the publications are: (i) enabling early warnings for quality risks, (ii) being able to give suggestions for improvements of the code base, (iii) planning support, and (iv) support for making decisions.

RQ4: What approaches to visualization have been investigated? Many of the publications focus on the source code, and color it based on some property. Engström et al illustrate how the source code is partitioned and color it based on test results, see Figure 3. Others,

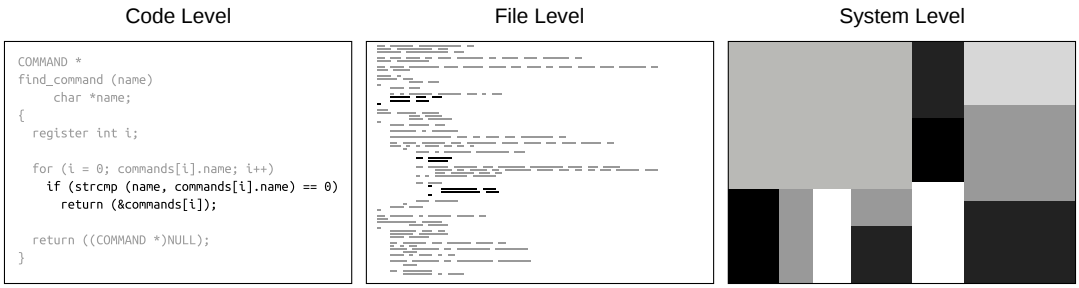

Figure 6: Visualization of three levels of abstraction. From statement (left) level, via file level (middle) to system level (right). Darker shades of gray, or black, indicate a larger need for attention (such as failing tests). Three levels from Orso et al, [16], with colors omitted for clarity.

including Feldt et al, illustrate this property over time without showing the structure of the source code, see Figure 4. Orso et al illustrates the source code in three different levels of abstraction: from line of code up to system level, see Figure 6. Haron and Syed-Mohamad used size in lines of code of the components instead of code structure as an axis, as well as test coverage and number of issues found in the components, see Figure 5. They illustrated this in a bubble chart, as opposed to in a heatmap. Interestingly the study by Haron and Syed-Mohamad was identified in our SLR since they discuss another publication using heatmaps in their section on related work, we come back to this topic in the Discussion, in particular in Section 5.3.

The test results was sometimes shown over time, [7, 8], or sometimes used as a static property, [10, 16]. One publication used test coverage instead of test results: [10].

Dimensions other than those of the source code and those of test results that had been used were time, and number of defects. Engström et al mentions that *"some participants also suggested that the visual analytics tool should collect data from additional data sources such as the source code repository. . . , requirement management system. . . and the defect tracking system"*, indicating that many other data sources have been considered.

Future work mentioned. To gather future work mentioned in the publications was not formulated as an explicit research question, but summarizing it is still meaningful. Here we present the proposed future work: (i) Increase scale: The authors requested answers to questions like: Will the implemented approach scale? How can we use data-mining approaches to improve the visualizations? Also: Can other data sources be integrated? (ii) Evaluation: One publication requested a more thorough evaluation: What direct and indirect effects are there on the organization? (iii) Usability, clickability and portability: Clicking or hovering with a mouse on areas in the heatmap should provide further information. Migrating to a web-based user interface was also mentioned. (iv) Automatic warnings: The possibility to add features like automatic warnings was mentioned, this is an extension to having an application for visualization with a human looking at it.

5 Discussion

5.1 Publications not included

An informal study of author homepages has revealed that at least two of the three research clusters identified have published more recent papers on visualization of test results. The method we used did not identify these papers, perhaps due to poor usage of search terms, the narrow focus on heatmaps or the limited number of databases used.

Only four publications were identified in this survey. Apart from this representing threats to validity (as discussed in the *Validity analysis* in section 5.6), this is also an interesting discovery: Is the approach of using heatmaps a new approach? Have other visualization techniques been used in the past, or has there perhaps been a recent drift in terminology?

Only peer-reviewed papers were investigated: Perhaps commercial or FLOSS tools do this already? In particular: tool kits for visualization were not included.

No commercial tools were included in this analysis. How could the findings from analyzing for example the test results views in popular tools such as MS Visual Studio, or Eclipse be included in a future study? To what extent are practitioners in the industry using commercial-off-the-shelf (COTS) software for these kinds of visualizations? Are COTS tools as good as the ones in scientific publications?

A popular belief is that other databases than those mentioned in Section 3.2 are very relevant, for example Google scholar.[3] Queries here could of course also have been considered. There is also a possibility that unpublished literature could have been used as input for this survey – a mitigation for this risk could have been to contact the authors whom had published relevant publications to ask them for unpublished literature of interest for this survey.

This SLR did not use techniques such as citation snowballing, [22]. This technique extends the number of publications that are potentially relevant by using the references in a relevant paper to find other publications of interest.

[3] https://scholar.google.com/

5.2 Visualization of Source Code

Most of the publications use source code in some way in the visualizations, typically with test coverage data. By doing this the authors seem to make a few assumptions: (i) Source code data is applicable. (ii) If an executed test case fails then the source code covered by the test case is highlighted for attention – meaning that the test cases produce results that are "true negatives" or "true positives". (iii) Data from any test framework source code is not of interest, and (iv) test coverage data is available and correct.

We speculate that the approach of using source code as the basis for the visualization may be a poor choice in some cases. In particular in projects with agile, rapid release or continuous integration approaches. This is supported by Elbaum et al, [6]: *"Traditional techniques tend to rely on code instrumentation and be applicable only to discrete, complete sets of test cases. In continuous integration, however, testing requests arrive at frequent intervals, rendering techniques that require significant analysis time [code instrumentation] overly expensive, and rendering techniques that must be applied to complete sets of test cases overly constrained..."*

This approach also assumes that test cases have very few *false negatives* (incorrect test failures), as these would "contaminate" the visualization.

We speculate that the assumption that VCS data is available is also slightly optimistic: Why should we exclude the possibility of using visualizations with heatmaps in scenarios where the software under test is a "black box", such as for typical system level testing?

Furthermore, the selection of source code as a base for visualization is only meaningful for stakeholders where source code is meaningful and also more intuitive than test cases (or any other artifact). This might eliminate requirements analysts, testers, and many other roles as stakeholders. We speculate that this limits the value of the visualization as it does not target a larger audience.

If source code is used then it should be noted that the industry practitioners in the study by Engström et al requested a fixed position of the source code units over time, as mentioned in Section 2.3.

5.3 Why Heatmaps?

This survey focused on discovering publications related to visualization of software testing data with heatmaps. The reader might ask "why heatmaps?", why not any other visualization technique such as a scatter plot matrix, as discussed in [19]? Perhaps an adaptation of these visualization to using implementations better suited for the discrete nature of test results can be meaningful?

During this survey it has become obvious that any future extension of this SLR should not limit the scope to only heatmaps. It might also be interesting to use an approach where researchers adapt visualizations from other domains into the domain of software testing.

5.4 Limitations and Future Research

This survey is limited, most notably when it comes to the number of publications reviewed (four), the number of databases investigated (two), the number of reviewers used (one), and also in the number of plot types considered (one).

Future research could focus on visualizations in other fields and adapt these to software testing. It could also be possible to stay within the field of software testing, but focus on other types of visualizations?

Engström et al explicitly mentions that industry practitioners were interested in seeing the integration of other sources of data: not only VCS, but also the requirement management systems and the defect tracking system. Haron et al mentions that their approach could be extended to become a suite of tools for business analysis.

A limited set of stakeholders were identified in this study. What would happen if roles such as requirements analysts and junior testers were considered? How would the visualizations be different if high level mangers or customers were considered?

This survey should be continued on the databases mentioned in Section 3.2. Searching in only two databases gives a first set of insights but can not be said to represent the full field of research. Continued research should also increase the structure in the SLR by more strictly adopting the methods proposed by [13], for example by using multiple reviewers.

5.5 Process for Introducing Visualizations

Engström et al mention Munzner's nested model, [15], a process for introducing visual analytics into a context and describe the four design stages (DS) as follows: *DS1) The first stage is to characterize the problem and the available data... DS2) to map the problem characterization to an information visualization problem, e.g. data types and operations, DS3) to design the visual encoding and interaction, and DS4) to implement this design with an effective algorithm.* Analyzing existing studies on how to successfully introduce visualizations could also be part of future research.

Other survey papers in fields related to software testing, perhaps system engineering, risk analysis or dashboard design, could also have been considered.

5.6 Validity Analysis

This is a limited study, there are several threats to validity. In the coming sections we perform a validity analysis based on the guidelines proposed by Runeson and Höst, [18].

5.6.1 Threats to Construct Validity

We remind the reader that research with good *construct validity* is research where the phenomenon under study represents what the researchers wanted to study. In

short: “did we study what we thought we were studying?”

We were unable to identify existing surveys where test results are visualized with heatmaps, but other surveys on similar topics might have rendered this survey redundant, or provided insightful additions. This could have been mitigated with a broader pre-study.

The reference papers were known prior to this study, it is possible that this SLR was over-engineered in order to capture these two. The limited sample size indicates the risk of a “convenience sample”. Two of the four studies included were Swedish. The reviewer was not supposed to know about the list of authors or affiliations of them during the review. It is still possible, however, that the reviewer favored Swedish researchers, a form of researcher bias.

A review protocol was only used for last parts of the survey, only the include/exclude *decision* of steps 3, 4, 5 and 6 was logged for earlier steps. A number of standard reasons for excluding papers could have been used to facilitate a review of the study. This could also have been useful in a future extensions.

During the review it was made very clear to us that the inclusion/exclusion criteria yielded very few papers. A possible mitigation approach could have been to have had a control step after reading the titles of the publications, in order to determine whether or not the search terms were good enough. At this point a decision could have been made to start over with better search terms.

Due to resource constraints only one reviewer was used. This increases the risk for researcher bias.

There are many threats to construct validity in this survey that are not mitigated, but the insights generated by the survey still represent relevant, but possibly incomplete, information.

5.6.2 Threats to Internal Validity

Studies with good *internal validity* can show the causality in their data, this is mostly relevant for data analysis. In short: “can we show that A leads to B?”

No data analysis has been done on the papers of interest, this means that no statistical methods have been used and that the data is qualitative. So conclusions drawn from this study have a risk of having researcher bias.

5.6.3 Threats to External Validity

Studies with good *external validity* have results that are valid in other contexts than the ones in which the study was performed. In short: “can these results be used elsewhere?”

The conclusions drawn in this paper might be valid now, but future studies might add to them, or revise them. Practitioners in the industry might have different data sources available. Conclusions in this paper on how practitioners could visualize test results may not be valid for the cases where a data source recommended is not available.

However, the findings in the included papers have been evaluated with a total of three industrial partners and several FLOSS projects. This makes us believe that the external validity is good.

5.6.4 Threats to Reliability

Studies with good *reliability* are independent of the researchers that performed them. In short: “Would other researchers come to the same conclusion with the same method?”

No external review of the research questions, the search terms, or the steps used for inclusion and exclusion has been conducted. These reviews are recommended parts according to Kitchenham, [13]. The actual outcome of the survey steps have not been reviewed by a second part. This further increases the risk of researcher bias.

The methods and databases used in this paper are well defined, we believe that the reliability of our results are rather good. There is, however, a non-negligible amount of human judgment involved in the steps used for inclusion and exclusion, described in Table 1, that might have an impact on reliability.

6 Conclusions

Visualizing tests results is hard. This survey used a structured approach to discover state of the art in visualizing test data with heatmaps, by investigating two electronic literature databases. We discovered that test results heatmaps have been used to target managers, experts, testers and code maintainers. The purpose has been to, for example, enable support in making decisions. These visualizations often, but not always, rely on the assumption that the software of the system under test is important and that the visualization should focus on it.

Despite limitations in this survey the knowledge summarized is valid, but might be extended. The publications identified as relevant have proposed the following future research: increase the scale of the visualizations, improve the evaluation, make them easier to use and extend with use cases such as automatic warnings.

7 Acknowledgments

This paper was written as a part of the DVA450 course at Mälardalen University (MDH), in the fall of 2016. It was subject to peer review by other students, all of them industry practitioners, in the scope of this course.

8 Author Biography

Per Erik Strandberg is a test lead at Westermo Research and Development AB, with more than a decade of experience working with software development, software

testing and test automation. His research interests include large-scale software test automation of embedded systems. Strandberg received a M.Sc. degree in bioinformatics, and also a M.Sc. degree in applied mathematics from Linköping University. He started as an Industrial Doctoral Student at Mälardalen University in 2017. Contact him at per.strandberg@westermo.se.

9 References

[1] D. Beyer & A. E. Hassan (2006, October). "Animated Visualization of Software History using Evolution Storyboards". In *WCRE (Vol. 6, pp. 199-210)*.

[2] P. Caserta and O. Zendra. "Visualization of the static aspects of software: A survey." In *IEEE transactions on visualization and computer graphics 17.7 (2011): 913-933*.

[3] F. Deissenboeck, E. Juergens, B. Hummel, S. Wagner, B. M. y Parareda, & M. Pizka. (2008). "Tool support for continuous quality control". In *IEEE software. 25(5)*, 60-67.

[4] S. C. Eick, J. L. Steffen & E. E. Sumner. (1992). "Seesoft-a tool for visualizing line oriented software statistics". In *IEEE Transactions on Software Engineering, 18(11), 957-968*.

[5] J. Ekanayake, J. Tappolet, H. C. Gall & A. Bernstein (2009, May). "Tracking concept drift of software projects using defect prediction quality". In *2009 6th IEEE International Working Conference on Mining Software Repositories* (pp. 51-60). IEEE.

[6] S. Elbaum, G. Rothermel & J. Penix. (2014, November). "Techniques for improving regression testing in continuous integration development environments". In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 235-245). ACM.

[7] E. Engström, M. Mantylä, P. Runeson & M. Borg (2014, March). "Supporting regression test scoping with visual analytics". In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation* (pp. 283-292). IEEE.

[8] R. Feldt, M. Staron, E. Hult & T. Liljegren (2013, September). "Supporting software decision meetings: Heatmaps for visualising test and code measurements". In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications* (pp. 62-69). IEEE.

[9] M. E. Froese & M. Tory (2016). "Lessons Learned from Designing Visualization Dashboards". In *IEEE computer graphics and applications*, 36(2), 83-89.

[10] N. H. Haron & S. M. Syed-Mohamad (2015, December). "Test and Defect Coverage Analytics Model for the assessment of software test adequacy". In *2015 9th Malaysian Software Engineering Conference (MySEC)* (pp. 13-18). IEEE.

[11] R. van Hees & J. Hage (2015, September). "Stable Voronoi-based visualizations for software quality monitoring". In *Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on* (pp. 6-15). IEEE.

[12] J. A. Jones, M. J. Harrold, & J. Stasko, (2002). "Visualization of test information to assist fault localization". In *Proceedings of the 24th international conference on Software engineering* (pp. 467-477). ACM.

[13] S. Keele (2007). "Guidelines for performing systematic literature reviews in software engineering". In *Technical report, Ver. 2.3 EBSE Technical Report. EBSE.*

[14] N. Nagappan, and T. Ball, (2005, May). "Use of relative code churn measures to predict system defect density". In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.* (pp. 284-292). IEEE.

[15] T. Munzner, (2009). "A nested model for visualization design and validation". In *IEEE transactions on visualization and computer graphics*, 15(6), 921-928.

[16] A. Orso, J. Jones & M. J. Harrold, (2003). "Visualization of program-execution data for deployed software". In *Proceedings of the 2003 ACM symposium on Software visualization* (pp. 67-ff). ACM.

[17] T. J. Ostrand, E. J. Weyuker, and R. M. Bell. (2005). "Predicting the location and number of faults in large software systems". In *IEEE Transactions on Software Engineering, 31(4), 340-355*.

[18] P. Runeson & M. Höst (2009). "Guidelines for conducting and reporting case study research in software engineering." In *Empirical software engineering, 14(2)*, 131-164.

[19] M. Sedlmair, T. Munzner & M. Tory (2013). "Empirical guidance on scatterplot and dimension reduction technique choices". *IEEE Transactions on Visualization and Computer Graphics*, 19(12), 2634-2643.

[20] M. Staron, J. Hansson, R. Feldt, A. Henriksson, W. Meding, S. Nilsson & C. Höglund (2013, October). "Measuring and Visualizing Code Stability– A Case Study at Three Companies". In *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint*

Conference of the 23rd International Workshop on (pp. 191-200). IEEE.

[21] P. E. Strandberg, D. Sundmark, W. Afzal, T. J. Ostrand & E. J. Weyuker (2016, October). "Experience Report: Automated System Level Regression Test Prioritization using multiple factors", in *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE)*, 2016.

[22] C. Wohlin (2014, May). "Guidelines for snowballing in systematic literature studies and a replication in software engineering". In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (p. 38). ACM.

[23] A. Zeileis, K. Hornik, & P. Murrell. (2009). "Escaping RGBland: selecting colors for statistical graphics". In *Computational Statistics & Data Analysis*, 53(9), 3259-3270.

[24] Color blindness. In *Wikipedia, The Free Encyclopedia*. Retrieved November 1, 2016.

[25] Heat map. In *Wikipedia, The Free Encyclopedia*. Retrieved November 1, 2016.

[26] Ishihara test. In *Wikipedia, The Free Encyclopedia*. Retrieved November 1, 2016.

A Appendix: Review Protocols

A.1 Included Papers

A.1.1 Visualization of Program-Execution...

Reference Orso et al, 2003, [16]

Affiliation Georgia Institute of Technology

Focus or Research Question Collect, transform and visualize program execution data. Transform it so it can be understood.

Data sources VCS, code execution data

Stakeholder Code maintainer (?)

What is Visualized? Three levels: statement, file and system level. In each one color hue an brightness is used to visualize the attention needed and the relevance or a line of code or a component.

How is this valuable? Find issues that occur in production (strange OS combined with strange Java distribution).

Environment/Evaluation evaluated with FLOSS projects.

Other Comments This paper was included since it explicitly mentioned that a "...system to visualize test-case information for fault localization...could be a specific instance of the approach described in this paper..."

A.1.2 Supporting software decision meetings...

Reference Feldt et al, 2013, [8]

Affiliation University of Göteborg.

Focus or Research Question Identify early warning indicators about potential problems with unexpected cost.

Data sources VCS target SW. VCS test system. Test outcomes.

Stakeholder Project manager, technical leader, test leader

What is Visualized? code churn over time, and test fail over time.

How is this valuable? Early warning. Suggest area for improvement.

Environment/Evaluation One company.

Other Comments This is one of the two reference papers.

A.1.3 Supporting regression test scoping...

Reference Engström et al, 2014, [7]

Affiliation Lund University, Aalto University

Focus or Research Question How design visualization to support regression test scoping?

Data sources Test coverage items vs historical test data.

Stakeholder Test manager, test experts, managers, testers

What is Visualized? Test coverage items vs historical test data (pass/fail per components)

How is this valuable? Support test planning, decision support.

Environment/Evaluation 2 industry partners as well as a FLOSS project.

Other Comments First two words in abstract are "test managers". Participants suggested also including data from VCS, requirements management systems as well as issue trackers. This is one of the two reference papers.

A.1.4 Test and Defect Coverage Analytics...

Reference Haron and Syed-Mohamad, 2015, [10]

Affiliation Universiti Sains Malaysia

Focus or Research Question How integrate test coverage and defect coverage.

Data sources Test coverage, defect coverage.

Stakeholder Test managers, project managers.

What is Visualized? Coverage vs defects vs number of SLOC, as an ID plugin.

How is this valuable? Make decisions on software test adequacy

Environment/Evaluation evaluated with FLOSS projects.

Other Comments Feels like an "initial study". No industrial connection.

A.2 Excluded Papers

For completeness the following sections contain the review protocols from the eight papers that were excluded in step 7, see Table 1.

A.2.1 Animated Visualization of Software...

Reference [1]

Year 2006

Affiliation EPFL Switzerland, University of Victoria

Focus or Research Question Visualize good and bad code quality over time

Data sources VCS Data

Stakeholder Unclear, but senior developers and system experts are explicitly mentioned in other contexts.

What is Visualized? Code component dependencies.

How is this valuable? Study and understand large code bases. Capture the informal knowledge. Useful when refactoring or rearchitecting.

Environment Evaluated with three FLOSS projects.

Other Comments Does not include test results data.

A.2.2 Tracking concept drift of software...

Reference [5]

Year 2009

Affiliation University of Zurich

Focus or Research Question Extend defect prediction with "concept drift".

Data sources VCS and issue tracker data.

Stakeholder Project Managers.

What is Visualized? Quality of the prediction of faults.

How is this valuable? Better quality of the predictions.

Environment Evaluated with four FLOSS projects.

Other Comments By "concept" they mean the bug generation processes. These change over time as for example authors come and go.

A.2.3 Empirical guidance on scatterplot...

Reference [19]

Year 2013

Affiliation University of Vienna, University of Brittish Columbia, University of Victoria

Focus or Research Question Data study.

Data sources More than 75 data sets.

Stakeholder Data analyst

What is Visualized? "any data"

How is this valuable? The methods are key in this paper.

Environment Not applicable.

Other Comments Few developers looked at many data visualizations and ranked them based on separation of data types. This is a very interesting paper for future work on visualizations, but this is not specific to the domain of software testing.

A.2.4 Measuring and Visualizing Code Stability...

Reference [20]

Year 2013

Affiliation University of Göteborg, Chalmers (another University in Göteborg), Volvo AB, Ericsson AB, Saab AB.

Focus or Research Question Monitor stability of software components, by monitoring changes in source code.

Data sources VCS data.

Stakeholder architect, process experts, designers, integration specialists

What is Visualized? code churn per component over time.

How is this valuable? Quality metric

Environment Three companies

Other Comments Does not include test results data.

A.2.5 Stable Voronoi-based visualizations...

Reference [11]

Year 2015

Affiliation Utrecht University

Focus or Research Question Stabilize a special form of source code visualization with respect to spatial stability over time.

Data sources VCS data.

Stakeholder Unclear.

What is Visualized? Source code: packages, files, ...

How is this valuable? Make it possible to compare a code base in different points in time.

Environment Evaluated with FLOSS projects.

Other Comments Does not include test results data.

A.2.6 Lessons Learned from... Dashboards

Reference [9]

Year 2016

Affiliation University of Victoria, Tableau Research

Focus or Research Question Lessons learned from making visualization dashboards.

Data sources Many

Stakeholder Middle to high level executives.

What is Visualized? Many things. Typically different KPIs per component.

How is this valuable? Make decisions

Environment Small start-up companies and institutes

Other Comments The user stories mentioned in the paper might be relevant for future work on visualizations, but this is not specific to the domain of software testing.