Supporting Regression Test Scoping with Visual Analytics

Emelie Engström*, Mika Mantylä†, Per Runeson* and Markus Borg*
* Dept. of Computer Science, Lund University, Sweden
(emelie.engstrom, per.runeson, markus.borg)@cs.lth.se
† Dep. of Computer Science and Engineering, Aalto University, Finland
mika.mantyla@aalto.fi

Abstract*—*Background: **Test managers have to repeatedly select test cases for test activities during evolution of large software systems. Researchers have widely studied automated test scoping, but have not fully investigated decision support with human interaction. We previously proposed the introduction of visual analytics for this purpose. *Aim:* In this empirical study we investigate how to design such decision support. *Method:* We explored the use of visual analytics using heat maps of historical test data for test scoping support by letting test managers evaluate prototype visualizations in three focus groups with in total nine industrial test experts. *Results:* All test managers in the study found the visual analytics useful for supporting test planning. However, our results show that different tasks and contexts require different types of visualizations. *Conclusion:* Important properties for test planning support are: ability to overview testing from different perspectives, ability to filter and zoom to compare subsets of the testing with respect to various attributes and the ability to manipulate the subset under analysis by selecting and deselecting test cases. Our results may be used to support the introduction of visual test analytics in practice.**

I. Introduction

In the evolution of large software systems, test management has to repeatedly select test cases for test activities to detect as many faults as possible within a given time and resource budget. In this paper we refer to these decisions as test scoping and in case they relate to software that has been previously tested we use the term regression test scoping. A common industrial practice is to base test scoping on practitioners' expertise and experience. As software systems grow in size and complexity the combinatorial explosion of test possibilities makes it infeasible to assess the plausibility of these decisions without tool support.

Research on regression testing has introduced several techniques for test selection. We found 28 empirically evaluated techniques [1] in a systematic literature review. However, most of the techniques do not involve human interaction in the selection procedure and thus they offer limited support in a complex and variability intensive industrial environment. This challenge is further confirmed in our mapping of test scoping techniques for software product line testing [2] and calls for new approaches to test scoping, which handle large numbers of test cases, are flexible to context variations and allow dynamic interaction with test managers to utilize their experience.

Visual analytics [3] may support tasks requiring interplay between humans and computers in order to reason from complex data. It is defined as *"the science of analytical reasoning facilitated by interactive visual interfaces"* [4]. One example of how visual analytics of test data is currently used in practice is project managers using visualization of failure rates of testing to assess product quality and answer questions like: *What areas need more defect fixing effort?* or: *When can we release the product?* The focus of this study is to use visual analytics for the purpose of improving test quality or planning of testing, for example, answering questions like: *Are there gaps or redundancy in our testing?* or: *What do we need to test due to this change?*

According to Munzner, introducing visual analytics into a context involves four design stages (DS) of development and validation [5]. DS1) The first stage is to characterize the problem and the available data. The following three stages are more creative and involve three different design problems: 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. We have previously studied the test scoping problem through a survey on regression testing practices [6], an action based case study on regression test improvement [7], and an in depth case study on test overlay [8]. DS1 of this paper builds on our experiences from those studies, while we here focus on DS2 and DS3 based on a case study on data from three different commercial organizations, and three focus groups with professionals from five software companies. We used an existing visualization front end [9] to provide hands on prototypes which we used for evaluation. The tool was selected to demonstrate some important visualization features but is not a complete solution to our visual analytics problem. Thus validation of the front end implementation is not within the scope of the paper.

In this paper we present a generic evaluation of our solution concept and a design proposal based on three industrial cases. Main contributions are: 1) a representation of test coverage items, TCIs [8], and identification of 15 corresponding attributes to focus the analysis on, 2) identification of 11 generic operations on the data needed to accomplish the scoping tasks, 3) evaluation of the visual encoding from TCIs to heat maps and 4) a discussion about which data transformations should be interactive and not. Sections II and III introduce the background and related work. Section IV describes our case study procedure and methods. In Section VI we discuss the results of the case study. Finally the paper is concluded in Section VII

II. Background

The solution concept evaluated in this paper aims at meeting challenges identified in industry [6], [7], [8] regarding regression test scoping in complex software development contexts. These challenges originate from the size and complexity of the software systems under development, market demands on frequent and customized releases of software and the development strategies that software organizations apply to meet those challenges.

In our industrial survey on regression testing practices [6] several problems related to test case selection were revealed such as:

- the difficulty in assessing the impact of changes,
- prioritizing with respect to product risks and fault detection ability, and
- determining a sufficient amount of tests and assessing different aspects of test coverage.

To enable concept evaluation we prototyped a visualization design which we present in this paper. This design proposal builds on two previous studies:

1) *A case study where we studied history based regression testing* [7], comparing tool based test scoping with manual scoping in a project. We identified a set of prioritization criteria which the practitioners considered important for test case selection: *historical effectiveness, execution history, static priority, coverage of functional areas* and *current and previous status of a test case*. Although our quantitative comparison showed that the tool based selection was more effective in terms of early fault detection than the manual selection, our qualitative evaluation, i.e. expert reviews, of the tool based selection suggests that it could be improved further. In this study we target this by adding visualization and human interaction to the test scoping procedure.

2) *An in depth industrial case study on test overlay* [8]. Based on observed improvement needs we elaborate on which type of visualization would be supportive. We conclude that decision support could be provided with visualization of:

- *Test design coverage from different perspectives* – Testing in a software project is carried out in accordance with various testing goals which may be both explicitly stated and implicit in the design of the test cases. A test case may cover one or more aspects of one or more testing goals. By enabling test coverage visualization from different perspectives and at different abstraction levels, test managers get support in evaluating their test strategies.
- *The aggregated test execution progress* – Tests are executed throughout and across projects in the software development organization. Tests of similar software are repeated due to evolution of the software itself but also due to differences between contexts where the software executes. By visualizing aggregated test execution information from different parts of the organization, test managers get support in identifying test overlay and inefficient testing, as well as inadequately fulfilled testing goals.
- *Priorities of coverage items based on different criteria* – Testers need to continuously prioritize which tests to execute, constrained by time and resources. The conditions for these priorities vary over time and across the organization. By enabling visual analysis of different prioritization criteria testers get support in making those decisions.

III. Related work

The regression testing literature offers a range of empirically evaluated techniques to support decisions on what to test in a given situation. These techniques and related evidence are overviewed in two recent literature reviews [1], [10]. The techniques are all on a rather high level of automation, i.e. providing decisions with little user interaction. Most of the techniques are also very context dependent and many test situations tend to be too complex to benefit from a single regression testing technique.

Visualization helps reducing cognitive complexity [11]. Software visualization is applied in various areas: e.g. static and dynamic code visualization [12], fault diagnostics and requirements analysis [13]. A survey of software maintenance, reengineering and reverse engineering [14] shows that 82% of researchers consider software visualization important or necessary for their work. Another survey with industry practitioners [15] identifies benefits of software visualization tools: reducing costs, increasing comprehension of software, increasing productivity and quality, management of complexity, and finding errors [16].

In the software testing area there are few studies on the use of visualization. Jones et al. [17] show how visualization of test information may support fault localization. Araya presents a visualization tool HaPAO to visually assess the quality of code coverage of unit tests [18]. A range of metrics for each class are visualized to illustrate how the methods in it have been covered during a set of tests. Similar to our results Feldt et al. [19] found heat maps of historical test data effective in analyzing patterns and thus useful at decision meetings.

Kienle and Müller identify requirements for software visualization tools with a focus on tools targeting the domain of software maintenance, reengineering, and reverse engineering [20]. Our results regard requirements on visualization for test scoping purposes and are in line with their findings.

Our main visualization approach is a heat map, a.k.a. mosaic visualization. Heat maps have been utilized by scientists and engineers for decades and been characterized as an ingenious display [21]. Wilkinson and Friendly [21] cite an article from 1873 which summarizes social statistics in different regions of Paris[1]. Perhaps the most notable use of heat in software engineering is the Seesoft tool [22] that displays the age, type or rate of code changes with different colors. To prototype our approach we used a tool called MosaiCode [9] which is designed by Maletic et al. to visualize metrics of source code files based on the Seesoft principles. In this paper the heat map is used to visualize test history data related to test coverage items, TCIs, e.g. test-case executions, failures, time since previous executions. To the authors' best knowledge

[1] http://www.math.yorku.ca/SCS/Gallery/images/loua1873-scalogram.jpg

there are no prior works presenting the use of the heat map of test-history data and the evaluation of such concept in test scoping activities as we do in this paper.

IV. Method

This study comprises two out of four design stages (DS) in Munzner's nested model for visualization design [5]. The four stages may be summarized as follows:

DS1 *Problem characterization* – The first design stage (which is based on previous studies) is the characterization of the domain problem and the available data.
DS2 *Mapping to solution concept* – The second stage is the mapping of the problem characterization to an information visualization problem, e.g. data types and operations.
DS3 *Solution design* – The third stage is the design of the visual encoding and interaction. Although this stage is covered in this study (we collected feedback regarding DS3 in the focus groups), we have not made an extensive comparison of alternative solutions. Instead we selected one representative instantiation to demonstrate the general solution concept.
DS4 *Design implementation* – The fourth stage (which is not covered in this report) is the implementation of the solution design with an effective algorithm.

The empirical study in this paper has characteristics of a case study, in that it "studies a contemporary phenomenon within its real-life context" [23, p12]. We study how visual analytics can be used to support regression test scoping in a variability intensive software development context. However, we isolate a portion of the context, in terms of test data from the three cases, and add the prototype visualizations as an intervention in the study. Further, we use focus groups for data collection [23, p54-55], to make the data collection more efficient, and for the added value of interaction between the study participants. We organized three focus group meetings, one for each of the organizations in the study. The focus group participants conducted a set of test evaluation and scoping tasks, reported and discussed their findings during the tasks, and the researchers observed their actions and reflections. Figure 1 provides an overview of the study design. The primary steps of the research procedure were:

1. *Design the study* – Describe the characteristics of the domain problem and data (DS1), set up the study goals (DS2, DS3), collect and explore the test data from the three cases
2. *Refine the solution concept*– Map the domain specific characteristics to generic operations and data type abstractions (DS2) and further to visualizations and interactions (DS3)
3. *Prepare for concept evaluation* – Generate prototype visualizations, pre-process the test data (DS2) for use in MosaiCode [9], define focus group procedures and questions, schedule meetings.
4. *Collect evaluation data* – Run the focus group meetings and have participants do the tasks, catalyze discussions, observe actions, take notes (DS2, DS3).
5. *Analyze evaluation data* – Collect notes from participants and observers, code and structure qualitative information (DS2, DS3).

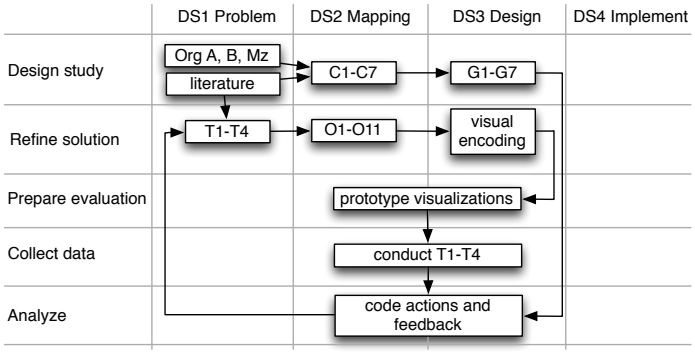

Fig. 1. Overview of the study design. Columns illustrate the evolution of the solution design and rows show the chronology of the current empirical study. C denotes challenges derived from the problem domain. G denotes goals for our study derived from the challenges. T denotes tasks performed in the focus group meetings. O denotes generic visual analytics operations needed to meet the challenges

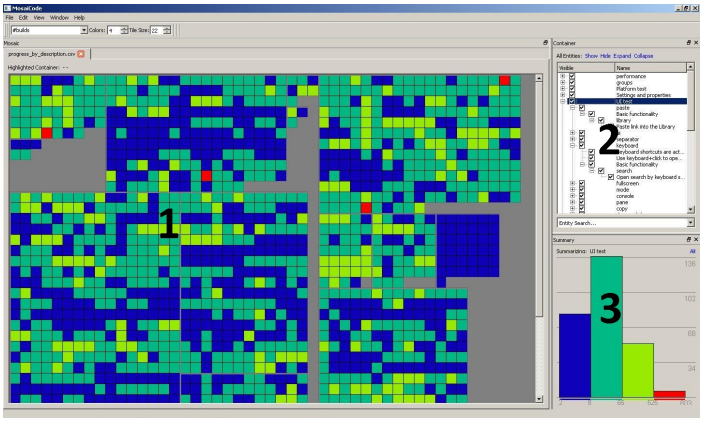

Fig. 2. MosaiCode tool visualization [9]. Screenshot of one of the prototype visualizations (1), tiles grouped in rectangles (containers), (2) tree browser window, and (3) summary window.

A. Prototype visualizations

As a front end for our prototype visualizations, we used the MosaiCode prototype tool [9], which was originally designed for visualizing structure and source code metrics of large scale software, see Figure 2. We mapped the test cases or test coverage items[2], TCIs [8][3], to these tiles, i.e. each tile represents one test coverage item from one perspective of the testing. Each tile is associated with a set of attribute values which may be visualized with different colors for different ranges of values. A tree browser window (2) visualizes the hierarchical structure of the tiles and enables selection of tiles to study. Below the tree browser there is functionality for searching tiles. Finally, a summary window (3) shows a histogram of the data visualized in the mosaic window.

B. Case descriptions

We searched for a generic solution concept and experimented on data from three different software development contexts, two proprietary and one open source context, which

[2]Entities describing the scope of a test or set of tests
[3]In the referenced paper the term used is only coverage items

TABLE I. THE THREE DATA SETS USED FOR EXPERIMENTATION AND PROTOTYPING

Case	Organization A	Organization B	Mozilla foundation
Characteristics	Emerging software product line	Safety critical software	Open source
Scope	Testing of one single function across organization	7 consecutive executions for a specific release	Full functional test suite for releases 2.0-9.0 of the Firefox web-browser.
Test cases	192	1 059	1 524
Test case executions	517	1 456	272 832
Time period	22 weeks	19 months	6 years

are described in this section. We refer to them as organization A, organization B and Mozilla Firefox. Table I shows an overview of the contexts of the three data sets.

1) Organization A: develops mobile devices based on the Android platform. The development context is variability intensive, i.e. a software project comprises about 10 different product variants, instances of specific combinations of hardware and software, which in turn are customized for hundreds of different pairs of customers and market segments. The development organization is globally distributed over three continents.

Testing tasks are organizationally distributed over three main units: core software, application software and product composition. Organization A applies incremental development practices which imply a need for continuous regression testing at all levels of test. The test data from organization A that we used in this study represents a small snapshot of the testing across the organization and was manually extracted and analyzed in an in-depth case study [8].

The four participants in the organization A focus group represent different levels of test and organizational units. All participants analyze test results as part of their work but with different purposes, e.g. test management in specific development projects, general test process improvement, line management, and product quality assessment.

2) Organization B: is a large multinational company operating in the robotics, power and automation sector. The studied development context comprises safety-critical embedded development in the domain of industrial control systems.

The verification process covers software testing on several different abstraction levels, partly conducted by independent testers. Furthermore, the process includes a partly automated suite of test cases that are executed daily, as well as a suite of regression tests that are executed prior to releasing new versions. As the regression suite is part of the safety case, required for the safety certification, the selection of test cases to include is crucial.

In the organization B focus group, a test manager and a project manager participated. Both participants had experience of planning test activities, including resource allocation and prioritization.

3) Mozilla foundation: develops, among other systems, the Firefox web-browser. Currently, Firefox has roughly 10 million lines of code and the typical number of people committing code at least once monthly is over 200.

Developers perform lower level testing, and developers and feature managers test that new features have been implemented correctly. Large crowds participate in the alpha and beta testing of the upcoming product releases. The main regression testing involves about 30 active testers frequently reporting test results to the open test management database.

In the mixed focus group, three software testing consultants from three different companies studied the visualizations of test execution data from the Firefox projects. The participants had worked in various roles, e.g. a test manager, test competence manager, coach of software testing, software test and process auditor, and project manager. They all had hands on as well as management experience in software testing.

C. Focus group meetings

The data from the three cases were used to prototype the solution concept for the focus group evaluations. In the mixed focus group meeting, testing consultants from three different companies evaluated the solution concept based on the Mozilla Firefox data. In the other two meetings, practitioners from organization A and organization B evaluated the solution concept based on their own data in addition to the Mozilla Firefox data. Each focus group meeting endured about two hours.

We provided between three and six different prototype variants[4] per context to focus the evaluation on different aspects of the visual analytics. In addition to the variation in data set, prototypes varied in terms of amount of available information (i.e. we added more information as tasks became more complicated), how different dimensions of the testing were mapped to the visual structure and whether or not similarities in execution patterns were visualized.

In the focus group meetings, we altered discussions with practical exercises[5]. The practitioners used the prototypes to perform test scoping tasks, T1–4 below, for specified development scenarios:

- T1 Get an overview of what is included in the test database. *Is the amount of designed test cases reasonable?*
- T2 Assess test execution progress. *Which areas are well-tested? Which areas need more testing? Is the test coverage sufficient? Explain! Which areas lack test cases?*
- T3 Plan a test session for a new platform (no changes in functionality). *How many/which test cases need to be run?*

[4]Note that prototypes here refer to the different visualizations provided by the tool rather than the tool itself.

[5]Preprocessed test data and focus group guide are available at https://serg.cs.lth.se/research/experiment_packages/rtsvis

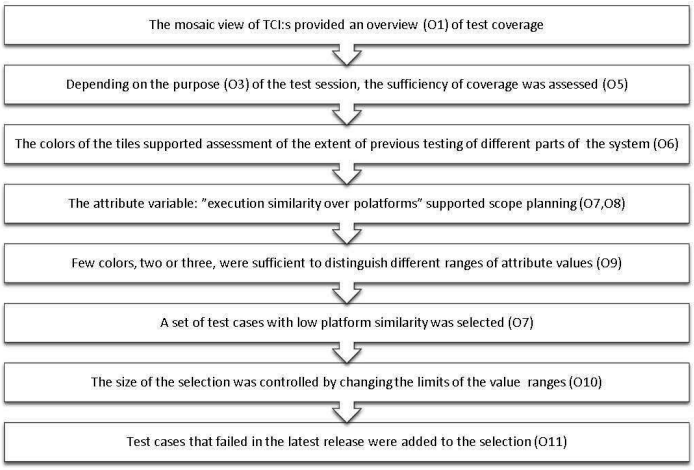

Fig. 3. Example workflow.

T4 Select and prioritize test cases after a change. *How many/which test cases need to be run?*

An example workflow from the focus group meetings is illustrated in Figure 3.

At least two researchers were present in each session to moderate the discussions and take notes. We collected written and oral feedback on the views and interactions from the participants. Their feedback regarded why, how and in which order views and interactions were used, as well as which views and interactions were not useful and which views and interactions were missing.

D. Analysis of evaluation data

After the focus group meetings, key findings were identified using qualitative analysis methods [23, p65-70]. Three authors independently carried out exploratory coding of the notes from the focus group meetings. We used these sets of findings to form the conclusions. The fourth author coded the notes based on the merged coding scheme from the first three co-authors. The outcome was then used for validation.

Four main groups of codes were identified: 1) observations regarding the tasks to be supported, 2) observations regarding information need and information access, 3) observations regarding the visual appearance and 4) observations regarding tool interaction. These groups of observation correspond well with the different facets of information visualization as described by Card et. al. [11].

E. Limitations

The threats to validity in the conceptual mapping stage (DS2), is the potential mismatch between needs of the target audience and the chosen abstractions, i.e. the generic operations and data types [5]. We addressed this threat by letting representatives from the target audience use prototype visualizations to perform typical but simplistic test management tasks and discuss their perceptions in the focus group format. The participants in the focus groups represented different testing contexts and test management roles and their different perspectives provided valuable insight to the problem. However, since we have focused on finding a generic solution concept rather than tailored our solution for one specific context, the solution need to be refined and evaluated further in the actual application context.

Threats to validity at the solution design stage (DS3) relate to how effective the visual mapping and interaction communicates the desired abstractions [5]. In our focus group sessions participants performed typical test management tasks supported by the visualizations. We did not quantitatively evaluate how well those tasks were performed. Instead we focused on the participants perceptions of usability of the solution concept and their opinions about how to design the user interface. We also drew conclusions from our observations of how the participants solved the tasks, i.e. whether they within the limited time frame of our focus group sessions (2 hours) managed to understand the tool, the structure of the data and the measurements and find a satisfying strategy to complete the tasks supported by the visualizations.

V. Results and analysis

All study participants confirmed potential usefulness of the visual analytics for test scoping as well as for communicating decisions with managers and subordinate testers. In this section we report a synthesis of the feedback from the focus groups and our experiences from experimentation and prototype generation. Contributions of this paper relate to the second and third stages of the nested design model by Munzner [5], which are described in Section IV, i.e. operation and data type abstraction (DS2), and visual encoding and interaction design (DS3). As recommended by Munzner we distinguish between the two design levels in the presentation and treat them separately. The presentation is further structured according to the different facets of information visualization as described by Card et al. [11]: the task to support with the visualization, the available data, the transformations (from raw data to data tables, to visual structures, to views) and the user interaction in the transformations. A summary is provided in Table II. The second left column contains the topics analyzed while the two rightmost columns contain observations and suggestions to guide the implementation of visual analytics in support of test management.

VI. Results and analysis

All study participants confirmed potential usefulness of the visual analytics for test scoping as well as for communicating decisions with managers and subordinate testers. In this section we report a synthesis of the feedback from the focus groups and our experiences from experimentation and prototype generation. Contributions of this paper relate to the second and third stages of the nested design model by Munzner [5], which are described in Section IV, i.e. operation and data type abstraction (DS2) and visual encoding and interaction design (DS3). As recommended by Munzner we distinguish between the two design levels in the presentation and treat them separately. The presentation is further structured according to the different facets of information visualization as described by Card et al. [11]: the task to support with the visualization, the available data, the transformations (from raw data to data tables, to visual structures, to views) and the user interaction in the transformations. A summary is provided in

TABLE II. OVERVIEW OF THE FACETS OF VISUAL TEST ANALYTICS (TASKS, RAW DATA, DATA TRANSFORMATION, VISUAL STRUCTURES AND HUMAN INTERACTION) EVALUATED IN OUR STUDY. THE COLUMNS SHOW HOW AND IF IT WAS IMPLEMENTED IN A PROTOTYPE AND THE MAIN OBSERVATIONS AND SUGGESTIONS FROM THE FOCUS GROUP DISCUSSIONS.

Facets		Implemented in prototypes	Observations	Suggestions
Tasks	Overview test cases	Performed in focus groups	Visual analytics was useful	
	Assess test design coverage		Visual analytics was useful to identify well tested areas	Add more information to identify gaps
	Assess test execution progress			
	Plan testing after change			
	Plan testing for new platform		Visual analytics was useful	
Raw Data	Test execution history	Test reports from test management databases and MS word documents	Visual analytics was good to overview large amounts of data	Aggregate all data in organization and enable filtering
	Change impact analyses	Not in prototype	Would improve test planning support	Link to delta information from source code repository and defect database
	System information		Would support assessment of test design coverage	Link to requirements and source code statistics
Data Transformation	TCIs	Test cases	Easily extracted from existing documentation	
		Implicit testing goals	Requires more effort, reveals test overlay in complex organizations	
	Dimension variables of TCIs	Project	Good for assessing progress	Visualize organizational dimension to evaluate teams or individuals. Visualize time dimension to assess progress. Visualize multiple dimensions of the TCIs
		Functionality	Good to get overview of scope	
		Execution pattern	Good for planning, less intuitive	
		Platform variants	Good for planning	
		Process		
	Attribute variables of TCIs	Similarity in execution patterns over different platforms	Useful for planning testing on new platform	
		#Executions, #Failures, #Builds, #Languages, #Platforms, #Days since first execution, #Days since last execution	Useful for planning and assessing execution progress	
		failure rate	Useful for all tasks	
Visual Structures	Marks	TCIs mapped to Mosaic tiles	Provides a good overview of both test design coverage and TCI execution progress, especially when data set is large	
	Use of space	Arranged in containers		Add text labels on containers
		Positioned in unstructured coordinate system		Maintain spatial position, Map TCI dimensions to nominal axes
	Graphical properties of marks	Attribute values mapped to colors	Only few colors are needed	
	Additional views	Dimension values mapped to tree structure	Useful for navigating when familiar with scope	
		Summary in histogram	Useful for planning	
Human Interaction	Interactive data transformation	Not in prototype	Human interaction is important	Enable manual filtering of data
	Interactive visual mapping			Enable manual mapping of TCI dimension to spatial position
				Enable manual mapping of attribute values to colors
	Interactive view transformation	Control of tile size	Prefer automatic setting	Enable selection and counting of test cases to support test planning
		Control number of colors		
		Search	Useful for planning testing after change	
		Browsing		
		Zooming	Very useful	

Table II. The second left column contains the topics analyzed while the two rightmost columns contain observations and suggestions to guide the implementation of visual analytics in support of test management.

A. Mapping to solution concept

From previous studies (DS1) the following list of challenges (C1–7) that need to be addressed by a visual analytics solution was observed: C1) huge amounts of tests repeated in time, space and system abstractions [24], [8]; C2) distribution of test responsibilities between organizational units [8]; C3) poor documentation and structure of test cases [8]; C4) parallel work [8]; C5) insufficient delta analysis [8]; C6) huge amounts of overlaid and potentially redundant testing [8]; C7) variation in test situations [6]. Based on this list of challenges we formulated a list of goals (G1–7), see Table III.

The remainder of this section describes our mapping of the domain-specific problem and data into abstract and generic descriptions (DS2). In terms of information visualization, the aim is to describe the operations and data types needed as input for the visual encoding (DS3) [5]. Our starting point is the list of goals (G1–G7) described in Table III. Subsection VI-B describe the mapping from concrete tasks to operations while Subsections VI-C and VI-D focus on the available data and the data transformations. Observations regarding tasks, available data and data transformations are summarized in Table II. Our findings regarding visual mapping and human interaction are

TABLE III. SUMMARY OF THE GOALS FOR THE VISUAL ANALYTICS

The visual analytics should support (mapped to Challenges in Section VI-A):		
G1	delta analysis	(C5)
G2	identification of redundancy	(C6)
G3	division of responsibilities	(C2, C4)
G4	users' interpretation of old test information	(C3)
by providing:		
G5	an aggregated view of test execution history from different perspectives	(C1)
G6	a transparent communication interface between different groups	(C2, C3, C4)
G7	flexible and adaptive decision support	(C7)

TABLE IV. SUMMARY OF THE REQUIRED GENERIC OPERATIONS

O1	*overview* and
O2	*browse* historical test information,
O3	*vary perspective* of the overview,
O4	*zoom* to test case level of details,
O5	visualize *missing information*,
O6	*compare* subsets of the testing,
O7	*identify* and *select* critical subsets,
O8	*select* attribute for comparison,
O9	*filter* on attributes,
O10	*vary size* of selection,
O11	*continue* analysis based on selection and deselection of tests.

described in Subsections VI-E and VI-F respectively.

B. Operations

In the focus groups, we asked the participants to perform concrete tasks, see T1–T4 in Section IV, supported by our prototype visualizations. The tasks involved *assessing the sufficiency of the testing* with respect to test design coverage (T1) and test execution progress (T2) and make decisions about what to test, *plan test*, in two different situations: after a specified functional change (T3) and after porting to a new platform without any changes in functionality (T4). T1 and T2 enable us to elaborate on the achievement of goals G2–G5; while T3 and T4 enable us to elaborate on the achievement of goals G1, G4, G6 and G7.

Based on the hands-on usage of the prototype visualizations, all participants perceived that the visual analytics could be used for many purposes, for example, reducing test resources and evaluating aspects of the testing, but also for generating new test ideas. From the practitioners' feedback on the usefulness of the visual analytics in relation to different tasks we extracted generic operations (O1–O11) needed to fulfill the goals. These operations resemble common visualization requirements listed in previous surveys [25], [15], [20]. The extracted operations are listed in Table IV.

Figure 3 shows an example workflow for tasks T1-T4 based on practitioners' feedback and our observations. All participants found the visual analytics useful to get an overview (O1) and for browsing (O2) and investigating details (O4) about the test data. Some participants also requested mosaic views where other perspectives (O3) of the testing were exposed, for example the time, project, or quality dimensions.

When assessing test design coverage (T1), the participants saw that test case execution data could provide information about which areas are well covered. To identify areas lacking testing (O5), they wished to combine this information with information from other sources such as customer feedback: *"It is the areas where failures end up in the hands of users"* or the defect tracking system. While some of this additional information is well documented and maintained in databases, some is not equally accessible but may exist in distributed reports or informal email conversations. Thus, combining information from several sources in an effective way to make informed test scoping decisions is attributed to the test managers' experience and expertise. This illustrates the need to incorporate humans in the loop rather than to design a fully automated decision support system, although a visual analytics system may evolve incrementally towards higher levels of automation. When assessing test execution progress (T2), participants' where positive: *"The visualization could be used to show everyone's progress." "The visualization could pinpoint areas to improve during spare time."* In cases where the participants were well familiar with the test scope, the gain in visualization was less *"By experience I know what to keep track of"*.

For both the decision making tasks (T3 and T4), all participants found views that they perceived supportive: *"Deciding on scope, the visualization helps identify similarities and to sort out.""Good support to make informed decision"*.

C. Available data

We focused on visualizing historical test information and extracted data from the test documentation in the three organizations. In the Mozilla Firefox case, we extracted test data from their open, web-based, test management system, Litmus, with a web crawler. In organization A data are stored in a similar commercial system, HP Quality Center, from which we manually extracted test information. In organization B, test executions were documented in MS Word documents and stored in a general document management system. We extracted the historical test data (semi-)automatically.

The focus group participants perceived that visualization of test execution history was useful for assessing test sufficiency and for making decisions. As stated above, some participants also suggested that the visual analytics tool should collect data from additional data sources such as the source code repository: *"We need a link to software delta"*, requirement management system: *"Link to requirements would be powerful as well"* and the defect tracking system. Such a system would have similarities with project intelligence platforms which enable trend and comparative analyses by collecting process and product metrics from several sources [26]. However, a prerequisite for enabling such analyses is that the information is available, and that organizations have policies allowing information integration. Unfortunately, large organizations often

manage information in different content management systems, typically offering limited interoperability (i.e. resulting in "information silos") [27].

D. Data transformation

In our visualization proposal, we transform test execution information to data tables or "cases by variable arrays" [11] where test coverage items (TCIs) are the cases. A TCI describes the scope of a test or set of tests [8] and is defined by a set of variable values. We differ between two types of variables: *dimension* variables with textual values and *attribute* variables with numerical values. A dimension value describes the hierarchical position of the TCI from one perspective, see Table V.

There is no limitation in how many variables to include in the analysis. However, the number of TCIs increases exponentially with the number of dimension variables. In total, we visualized five different dimension variables: *project* and *functionality*, for all data sets; *execution patterns*, for the Mozilla Firefox and organization B data; *platform variants*, for the Mozilla Firefox and organization A data; and *process* for the organization A data. We could extract some dimension values directly from the documentation, such as project information (in terms of release numbers), platform variants and process information (in terms of test activities). To derive the hierarchical structure of the functionality dimension, we had to interpret the test case descriptions in the light of available documentation and expertise. In the Mozilla Firefox and organization B cases we based this functional classification on keywords, while in the organization A case we did it manually. The *execution patterns* dimension refers to a classification based on hierarchical clustering on similarities between test cases with respect to when, on what and with which verdict they have been executed.

Our evaluation revealed needs to visualize different dimensions or perspectives of the testing (O3), depending on the user's role and tasks. In the organization A focus group, one of the participants suggested an *organizational* dimension to evaluate the testing by teams or even by individuals. The *functionality* dimension was considered useful to get an overview (O1) of the scope and to navigate (O2) the information. The classification based on *similarities in execution patterns* was appreciated for planning but was not intuitive and required clarification during the focus group.

Our evaluation also showed the need to measure different attributes (O6, O7) of the TCIs for different tasks. *Failure rate* showed to be useful for assessing the test design coverage (T1). Several attributes were considered by at least one of the participants when assessing execution progress (T2): *#builds tested, #languages tested, #platforms tested, #days since last execution, #executions, #failures, failure rate* and *#days since first execution*. In addition to these attributes they used measurements of *similarity of execution patterns* over different platforms to plan testing of new product variants (T3). It is important that the naming or description of attributes is clear and thus consistently understood by all users of the tool.

E. Visual mapping

We mapped the test cases or test coverage items TCIs [8], to the tiles in the mosaic view, i.e. each tile represents one TCI

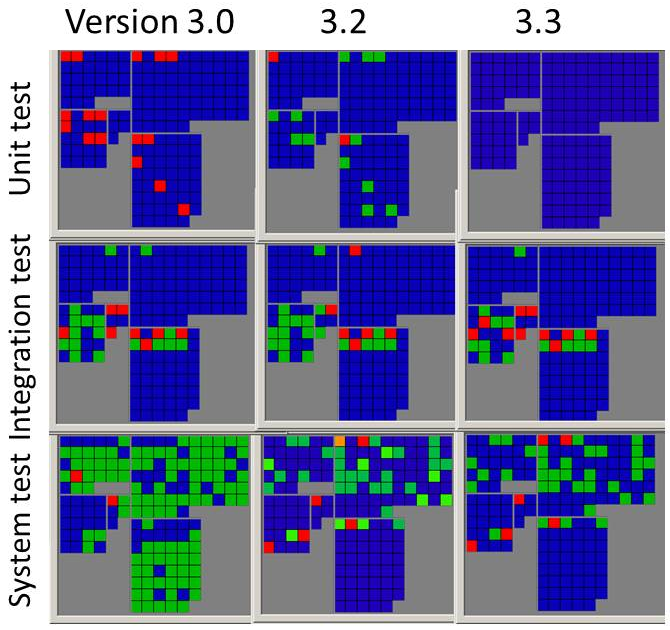

Fig. 4. Example of pairwise overview of the TCI dimensions process and project

from one perspective of the testing. Each element is associated with a set of attribute values which may be visualized with different colors for different ranges of values, see Figure 2.

The main components of a visual structure are the use of space, the marks and the graphical properties of the marks [11]. In the prototype visualizations, the spatial position encoded the dimension values of the TCIs in the mosaic window, which provided a good overview from one perspective (O1). Altering between different spatial encodings provided possibility to change perspective (O3). Preferably, several dimensions should be available at the same time (O3), for example with multiple tabs. In our prototype visualizations we did not make use of the two-dimensional positioning in the mosaic window, which could enable pairwise overviews of TCI dimensions (O1), e.g. functionality versus variability. Instead we created a two-dimensional overview by mapping the dimension values of one dimension variable to a set of attribute variables. Figure 4 shows an example of a pairwise overview (process/project) of the test execution progress.

The attribute values are visible in the colors of the mosaic tiles and in the histogram view on the right, see Figure 2. This enables comparison of TCIs (O6). Participants commented on the number of colors used to represent different ranges of values of the attributes. The general opinion was that only few colors should be used; two or three would be enough to identify critical tests (O7).

The MosaiCode tool [9] only supported unstructured type of axis [11] in the mosaic window. This means that the coordinates of the tiles had no meaning. Instead the arrangement of tiles into containers carried meaning. Thus, the dimension values, or the hierarchal position, are visible to the user through the grouping of tiles and in the tree browser, see Figure 2. This lack of a coordinate system in the mosaic window was frustrating for the testers. Participants in all focus groups

TABLE V. DATA TABLE OF EXAMPLE TEST COVERAGE ITEMS – THE HIERARCHICAL POSITIONS ARE DESCRIBED ALONG SIX DIMENSIONS OF THE TESTING STRATEGY. TCI 11 IS HERE A SUB TCI OF TCI 1021 WITH RESPECT TO THE TIME DIMENSION. THE ABSTRACTION LEVEL OF THE TCI DETERMINES THE DERIVED VALUES OF THE LIST OF ATTRIBUTES.

	TCI 1021	**TCI 11**
Dimension: Functionality	System/Social phonebook/Add contact	System/Social phonebook/Add contact
Dimension: Purpose	All/Interaction/Incoming message	All/Interaction/Incoming message
Dimension: Variant	All/Blue star	All/Blue star
Dimension: Platform	All	All
Dimension: Organization	OrganizationA/Application software/System integration test	OrganizationA/Application software/System integration test
Dimension: Time	All	All/2012/w46
Attribute: #executions	13	0
Attribute: #failures	0	0
Attribute: #days since last execution	30	30
...	...	...

requested at least a fixed position of tiles. They also suggested that the meaning of the containers should be visible, not only in the tree browser window, but also in the mosaic window. This could be done by adding text labels to the containers or to nominal axes. The use of nominal axis would also support the visualization of missing information (O5). In Figure 4 nominal axes are combined with unstructured to provide a 3D view. Similarity in functional coverage is here visible in the containers and progress over project and process is visible through the coordinate system.

In the evaluation, we studied how the participants used the different views to solve the tasks. The tree browser view showed to be useful for navigating (O2) when participants were familiar with the test scope and the structure of TCIs. The mosaic view provides a good overview (O1) of both TCI design coverage (T1) and TCI execution progress (T2) and showed to be useful when the participants were new to the data or when the dataset was large.

F. Human interaction

Participants in all three focus groups stressed the importance of human interaction. Several techniques exist to let the user control the different types of data transformations in the visual analytics process [11]. In the prototype visualizations, the *transformation from raw data into data tables* as well as the *mapping from data tables to visual structures* was hard coded. Thus, the user interaction provided in focus groups, with respect to these transformations, was limited to selecting which prototype visualization to use for a certain task. Participants in all focus groups requested a possibility to filter (O9) the displayed data on different criteria, e.g. visualize test execution history of functional area F by team A from Nov 2011.

Participants discussed two types of interactive visual mapping which they considered useful namely 1) the selection of which TCI dimension to be mapped to the spatial position in the mosaic view (O3) and 2) the setting of percentiles in the histogram, that is, the mapping of colors to attribute values to vary the size of the selection (O10).

The MosaiCode tool provides *interactive view transformations* in terms of browsing (O2) and zooming (O4) as well as controlling the tile size and number of different colors on tiles. In addition, automatic mapping between the different views enables searching for elements in, for example, the tree browser by selecting them in the mosaic window. Practitioners considered the zooming function very useful and stressed the importance of a direct mapping between different views i.e. if a certain test coverage item is selected in the tree browser view, the same item should be selected in the mosaic view. Furthermore they requested automatic setting of tile sizes but manual setting of tile color ranges based on the TCI attributes. Additionally, they wanted to be able to select and count test cases to support the test planning (O7, O8).

G. Summary of results

The outcome of DS2 is a mapping of the regression test scoping problem to an information visualization problem. We used data tables of TCIs to represent test execution information. TCIs may be defined by dimension variables and attribute variables. We identified 5 dimension variables and 10 attribute variables that were useful to visualize in different regression test scoping decisions. These are listed and explained in SectionVI-D. We also identified 11 generic visual analytics operations (See Table IV), which were required to meet previously identified challenges in regression test scoping.

The outcome of DS3 is the design of the visual encoding and interaction. We used heat maps to visualize the TCIs. Colors of tiles represent attribute values and spatial position of tiles represents the dimension values. During evaluation we observed needs related to the views and interactions. While performing the scope analysis it should be possible to alter between different perspectives, i.e. change both dimension variables and attribute values and to select and count TCIs. To elaborate on the size of the selection user need also be able to manually set attribute value ranges for the colors. Only few colors (two or three) are needed to differentiate ranges of attribute values.

VII. CONCLUSION

We have continued our design study on how to support regression test scoping in a complex software development context. We provided prototype visualizations based on three different sets of industrial data and evaluated the visual analytics in three focus groups with participants from software industry. Contributions presented in this paper are the description and evaluation of a generic solution concept in terms of required operations and a proposed data representation; as well as an exemplification and evaluation of how to design the visual mapping and interaction. The generic solution provided in this paper is relevant in cases were large amounts of test

cases are executed and there is a need to cut testing costs. However it need to be further refined and evaluated in the actual application context. Both indirect and direct effects [28] on the organization should be considered.

ACKNOWLEDGEMENT

The authors are very thankful to the case companies and the focus group participants, for letting us access them and their data, as well as giving valuable feedback on our findings. We also thank the SERG reading group for review comments to an earlier version of the paper. This work was partly funded by ELLIIT (The Linköping-Lund Initiative on IT and Mobile Communication, www.elliit.liu.se) and EASE (Industrial Excellence Center on Embedded Applications Software Engineering, ease.cs.lth.se). Emelie Engström and Markus Borg are members of the SWELL research school, (Swedish V&V Excellence, www.swell.se).

REFERENCES

[1] E. Engström, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," *Information and Software Technology*, vol. 52, no. 1, pp. 14–30, Jan. 2010.

[2] E. Engström and P. Runeson, "Software product line testing a systematic mapping study," *Information and Software Technology*, vol. 53, no. 1, pp. 2–13, 2011.

[3] P. C. Wong and J. Thomas, "Visual analytics," *IEEE Computer Graphics and Applications*, vol. 24, no. 5, pp. 20 – 21, Oct. 2004.

[4] K. A. Cook and J. J. Thomas, "Illuminating the path: The research and development agenda for visual analytics," Pacific Northwest National Laboratory (PNNL), Richland, WA (US), Tech. Rep. PNNL-SA-45230, May 2005.

[5] T. Munzner, "A nested model for visualization design and validation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 921–928, Dec. 2009.

[6] E. Engström and P. Runeson, "A qualitative survey of regression testing practices," in *Product-Focused Software Process Improvement*, ser. Lecture Notes in Computer Science, M. Ali Babar, M. Vierimaa, and M. Oivo, Eds. Springer Berlin / Heidelberg, 2010, vol. 6156, pp. 3–16.

[7] E. Engström, P. Runeson, and A. Ljung, "Improving regression testing transparency and efficiency with history based prioritization an industrial case study," in *Proceedings of the 4th International Conference on Software Testing Verification and Validation (ICST'11)*, 2011, pp. 367 –376.

[8] E. Engström and P. Runeson, "Test overlay in an emerging software product line an industrial case study," *Information and Software Technology*, vol. 55, no. 3, pp. 581–594, Mar. 2013.

[9] J. I. Maletic, D. J. Mosora, C. D. Newman, M. L. Collard, A. Sutton, and B. P. Robinson, "MosaiCode: visualizing large scale software: A tool demonstration," in *2011 6th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, Sep. 2011, pp. 1–4.

[10] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, Mar. 2012.

[11] S. K. Card, J. Mackinlay, and B. Shneiderman, Eds., *Readings in Information Visualization: Using Vision to Think*, 1st ed. Academic Press, Feb. 1999.

[12] T. A. Ball and S. G. Eick, "Software visualization in the large," *Computer*, vol. 29, no. 4, pp. 33 –43, Apr. 1996.

[13] D. Graanin, K. Matkovi, and M. Eltoweissy, "Software visualization," *Innovations in Systems and Software Engineering*, vol. 1, no. 2, pp. 221–230, 2005.

[14] R. Koschke, "Software visualization for reverse engineering," in *Software Visualization*, ser. Lecture Notes in Computer Science, S. Diehl, Ed. Springer Berlin Heidelberg, Jan. 2002, no. 2269, pp. 138–150.

[15] S. Bassil and R. K. Keller, "Software visualization tools: survey and analysis," in *Proceedings of the 9th International Workshop on Program Comprehension (IWPC 2001)*, 2001, pp. 7–17.

[16] S. Diehl, *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*, 1st ed. Springer, May 2007.

[17] J. A. Jones, M. J. Harrold, and J. Stasko, "Visualization of test information to assist fault localization," in *Proceedings of the ACM 24th International Conference on Software Engineering (ICSE '02)*, 2002, pp. 467–477.

[18] V. P. Araya, "Test blueprint: an effective visual support for test coverage," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. New York, NY, USA: ACM, 2011, pp. 1140—1142.

[19] R. Feldt, M. Staron, E. Hult, and T. Liljegren, "Supporting software decision meetings: Heatmaps for visualising test and code measurements," in *2013 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, 2013, pp. 62–69.

[20] H. M. Kienle and H. A. Müller, "Requirements of software visualization tools: A literature survey," in *IEEE Proceedings of the 4th International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2007)*, Jun. 2007, pp. 2 –9.

[21] L. Wilkinson and M. Friendly, "The history of the cluster heat map," *The American Statistician*, vol. 63, no. 2, 2009.

[22] S. Eick, J. Steffen, and J. Sumner, E.E., "Seesoft-a tool for visualizing line oriented software statistics," *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 957–968, 1992.

[23] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering Guidelines and Examples*. Wiley, 2012.

[24] P. Runeson and E. Engström, "Software product line testing a 3D regression testing problem," in *Proceedings of the IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST)*, Apr. 2012, pp. 742 –746.

[25] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," in *Proceedings IEEE Symposium on Visual Languages*, Sep. 1996, pp. 336–343.

[26] F. Deissenboeck, E. Juergens, B. Hummel, S. Wagner, B. Mas y Parareda, and M. Pizka, "Tool support for continuous quality control," *IEEE Software*, vol. 25, no. 5, pp. 60–67, Oct. 2008.

[27] J. A. Vayghan, S. M. Garfinkle, C. Walenta, D. C. Healy, and Z. Valentin, "The internal information transformation of IBM," *IBM Systems Journal*, vol. 46, no. 4, pp. 669–684, 2007.

[28] E. Engström, R. Feldt, and R. Torkar, "Indirect effects in evidential assessment: a case study on regression test technology adoption," in *Proceedings of the 2nd international workshop on Evidential assessment of software technologies*, ser. EAST '12. ACM, 2012, pp. 15–20.