Cardiogram: Visual Analytics for Automotive Engineers

Michael Sedlmair[1], Petra Isenberg[2], Dominikus Baur[3], Michael Mauerer[3], Christian Pigorsch[4], Andreas Butz[3]

[1]University of British Columbia, Vancouver, BC, Canada
[2]INRIA, Paris, France
[3]University of Munich, Munich, Germany
[4]BMW Group, Munich, Germany

msedl@cs.ubc.ca, petra.isenberg@inria.fr, dominikus.baur@ifi.lmu.de, michael.mauerer@cip.ifi.lmu.de, christian.pigorsch@bmw.de, butz@ifi.lmu.de

ABSTRACT

We present Cardiogram, a visual analytics system that supports automotive engineers in debugging masses of traces each consisting of millions of recorded messages from in-car communication networks. With their increasing complexity, ensuring these safety-critical networks to be error-free has become a major task and challenge for automotive engineers. To overcome shortcomings of current analysis tools, Cardiogram combines visualization techniques with a data preprocessing approach to automatically reduce complexity based on engineers' domain knowledge. In this paper, we provide the findings from an exploratory, three-year field study within a large automotive company, studying current practices of engineers, the challenges they meet and the characteristics for integrating novel visual analytics tools into their work practices. We then introduce Cardiogram, discuss how our field analysis influenced our design decisions, and present a qualitative, long-term, in-depth evaluation. Results of this study showed that our participants successfully used Cardiogram to increase the amount of analyzable information, to externalize domain knowledge, and to provide new insights into trace data. Our design approach finally led to the adoption of Cardiogram into engineers' daily practices.

Author Keywords

Visual Analytics, Visualization, Automotive

ACM Classification Keywords

H.5.0 Information Interfaces and Presentation[General]

General Terms

Design

INTRODUCTION

The field of visual analytics (VA) is continuously growing with research efforts expanding into many different domains. Visual analytics tools address the challenge of analyzing overwhelming amounts of data by combining methods from various disciplines, including information visualization (InfoVis), HCI and data analysis techniques from statistics, data mining, and others [36]. One very crucial aspect for the future of VA is to bridge the gap from research to application as the success of the field may eventually be measured by how many people will actively be using solutions in their everyday work [5]. Thus, InfoVis and VA researchers have increasingly called for more rich descriptions of novel problem domains for visual analytics and for design studies in these areas [4, 13, 14, 22] in order to gain a better understanding of everyday data analysis practices across different domains.

The research we present in this paper follows this call. We present the results of a three-year research endeavor (6 researchers / approx. 7 man-years) with the goal to develop, connect, and integrate visual analytics tools directly within an industrial work context. We worked closely with domain experts in a large automotive company to support data analysis challenges encountered during the debugging of sensor networks in modern automobiles and provided them with several solutions for their everyday work. In order to ensure that our tools would meet the requirements of the industrial work context, we engaged in a long-term problem characterization phase [13, 22], designed and tested various prototype and tool designs, and finally adopted and integrated some of them into the engineers' daily working practices. The largest of these tools, Cardiogram, is the focus of this paper.

Our work in the visual analytics domain makes three primary contributions: We first discuss our in-depth understanding from a three-year exploratory field analysis. Second, we introduce Cardiogram, a novel tool for the analysis of in-car communication traces. Cardiogram was built based on our rich understanding gathered from the field analysis and developed in close cooperation with domain experts. We discuss how we integrated the tool in the industrial work context and present the results of a qualitative, long-term, in-depth user study which showed various benefits extending current working practices of engineers. Finally, we review our experience from the domain-specific aspects of the project and discuss design decisions that led to our successful integration of visual analytics tools in a large industrial context.

While described characteristics of our application domain are meant to guide practitioners and developers in the same or similar domains, the description of our process may prove to be useful for visual analytics practitioners in many areas who wish to form closer connections to industrial partners.

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

INDUSTRIAL BACKGROUND

Our work is situated in the domain of automotive engineering where we worked with engineers involved in the development of sensor networks in cars. An in-car sensor network consists of sensors which send information such as the vehicle's speed, actuators translating electronic information into mechanical behavior, and electronic control units (ECUs) that compute this data. Over the last years, the concentration of electronic and software components in automobiles has increased enormously resulting in highly complex in-car networks. Current networks consist of up to 100 ECUs and 300 sensors/actuators, all interconnected via several, heterogenous bus technologies (CAN, FlexRay, MOST, LIN, Ethernet) [1]. This increasing complexity has brought a variety of novel challenges such as intricate error diagnosis and recovery to automotive engineers [3, 8, 25]. ECUs of high-performance cars, for example, can deliver information at speeds of more than 1,000 readings per second over the in-car network and use this information to trigger specific actions. In the case of an airbag, for example, accelerometers send information to a microprocessor at 10ms or faster intervals and the evaluation of this data determines whether and how to inflate the airbag [1]. Transporting all relevant information typically results in more than 15,000 messages per second distributed over the network.

To verify the correctness of sensor networks, engineers log network messages via specific hardware that is installed in test cars. A one-hour log file typically consists of several GB of data with up to 50 million recorded messages. In the case an error has been occurred in a test drive, our target users' task is to inspect the log files and to identify the error source. Time and experience is needed to understand in-car communication processes and to detect sources of errors in the networks and their complex correlations. In addition, analysts cannot rely on simulations for error detection as errors are often caused by hardware or external factors. Therefore, log files are currently the only lead. These data related challenges make this work domain a prime candidate for dedicated visual analytics tools. Providing effective tools for the analysis of this data is of highest importance as the safety of the automobile and its passengers hinges on the ability of automotive engineers to understand and debug this sensor data.

RELATED WORK

We review related work of previous rich domain studies in information visualization and visual analytics and on usage of visualization in automotive engineering.

End User Integration in Visual Analytics

Recently, information visualization researchers have explicitly called for a closer integration of end-users in InfoVis/VA tool development and for the dissemination of qualitative reports on data analysis practices in real-life work contexts to ground both design and/or subsequent evaluation [4, 13, 14, 22]. Indeed, more and more InfoVis/VA projects focus on solutions for real-world, data-intensive application domains and integrate users into their development processes. Two recent examples incude a visual analytics tool for patent specialists [18] and a financial data analysis tool [35]. In our work, we used ethnographic field analysis of current practices to richly inform the design of our systems. The need to base system design on an in-depth understanding of a domain has been understood in HCI for a long time [34], and recently has also been more and more addressed by InfoVis/VA research: Tory et al. [37], for instance, conducted a qualitative analysis in the building design field, Isenberg et al. [12] studied the work of traditional collaborative data analysis, McLachlan et al. [20] worked together with system management professionals, Meyer et al. [21] with gene expression biologists, or Kincaid with oscilloscope users [17]. None of these studies so far, however, have been conducted within a large industrial setting or in our application domain.

Visualization in the Automotive Domain

Visualization in the automotive domain is most commonly used in the context of computer-aided-design, virtual reality, and scientific visualization [33]. Within scientific visualization, many techniques have focused on the analysis of physically-based (often simulated) data, such as the flow of particles for car body development [26]. Such techniques have also been integrated with information visualizations such as scatterplots and histograms for the analysis of, for example, a Diesel exhaust system [6]. While some of the scientific visualization work (e. g., [19]) shows increasing interest in integrating information visualization, considerably less work has been dedicated to the support of *electronic* engineering for car development and testing. In our own previous work, we presented several prototypes for visualizing in-car communication networks [27, 28, 30]. These systems were standalone point-solutions for specific analysis challenges and helped us to explore the design space. In this paper, we build on this work and present extended insights, a novel, fully integrated solution and its in-depth evaluation.

METHODOLOGY

We followed a general development cycle including stages of pre-design analysis, prototyping, development, and testing. Our methodology was chosen based on two main objectives: (1) Learning about the field including current practices, problems and challenges, and (2) learning about our tool and how to design and integrate it successfully in our target domain. For this purpose, we primarily focused on working closely with experts in our target domain. Both the complexity of data and analysis tasks as well as the time-constraints of the industrial work context influenced the study methods we chose and how we adopted them to these specific requirements. In spirit, our approach is similar to MILCs [32] (Multi-dimensional In-depth Long-term Case Studies) as advocated previously for information visualization for addressing the specific needs of this domain with its complex, exploratory and ill-defined tasks. Similar to the traditional MILC approach, we used a *multi-dimensional* set of study methods, and carried out *in-depth* and *long-term* investigations over a period of three years. We studied the data analysis practices of a large number of 50 expert analysts using both, their own, currently used tools and our novel tools, and did not focus on studying any tool in particular. We, thus, conducted an "ambitious" MILC as proposed for future work in [32]. Reflecting the typical structure of our target users, approx. 90% of the analysts we worked with were male and had been working in the problem domain for 2–16 years.

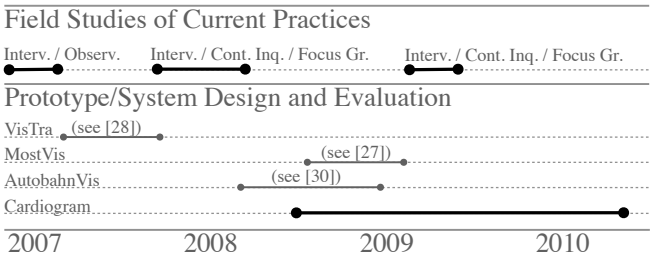

Figure 1: Overview of our development process. Black lines indicate work reported on here. Gray lines indicate phases reported on earlier.

In the following, we describe the various methods we used for studying current practices for introducing VA in our domain. We also briefly reflect on our objectives and experience using these methods for InfoVis/VA in a large company setting in order to provide other researchers with guidance for such undertakings; for a detailed description of our experience with specific evaluation requirements in large company settings we refer the reader to [29]. Our study is among one of the first in-depth endeavors of studying the complex pattern of data analysis using long-term MILCs. The time investment of three years was substantial but essential to our success later. Figure 1 provides a temporal overview of our studies as well as the different participatory tool design and evaluation phases in which we closely cooperated with our target users.

Field Studies of Current Practices

Interviews: Especially early on, we relied on semi-structured interviews to gain insights into aspects beyond what we could derive from technical documentation. Interviews provided us with a rough understanding of work practices, and showed engineers' estimations on challenges and problems as well as their ideas and approaches to overcome them. Overall, we conducted 30 1h-long interviews with both, analysts and analysis tool designers. From our notes and recordings we derived main categories of tasks and analysis challenges.

User Observation and Contextual Inquiries: In order to collect more real-life data of our experts' work practices, we started observational studies with a fly-on-the-wall technique in which we followed but did not interfere with the daily practices of the engineers. However, due to task complexity and high expertise in our domain it was not possible for a non-domain-expert to derive meaningful results through observation alone. Therefore, we chose to use contextual inquiries [11] instead which we conducted with 14 analysts (11 previously participated in interviews) in up to five 1h-long sessions per participant. Due to IPR restrictions, we solely used note-taking for data logging purposes. We used 2–3 note takers to counterbalance the lack of recording equipment. We used these results to broaden our understanding of the diversity of working practices in our domain and to verify or refute our prior findings.

Focus Groups: To evaluate, improve, and focus our findings from interviews and user observations we conducted 17 focus groups of 3–10 participants. We chose focus groups because we frequently encountered diverse and even opposed practices and statements during individual interviews. Our major goal in bringing experts together was, therefore, to form a common understanding—not just between the end users and us but also among the end users themselves. The groups consisted of varying groups of analysts, tool designers and we also invited decision makers. The latter are an important group to address in large company settings as they usually decide whether a novel solution will be integrated and funded or not. As our goal was to reach adoption of our solutions, we decided to integrate decision makers early in the process and this turned out to be essential to our success.

Participatory Design and Tool Evaluation

We augmented the methods used above with a tight collaboration with end users during the design and evaluation of our own VA tools. Both lessons learned from our previous tools but also of Cardiogram itself helped us gain additional insights into applying VA in our domain and specific challenges for VA tools and their integration into the end users' work environment. Our long-term process was, therefore, an iterative mix of current-practice field studies, tool design and tool evaluation—each of them informing another by providing us but also our end users with new insights (cf. Figure 1). For designing and evaluating our tools, including Cardiogram, we used the following methods:

Design Workshops and Personal Feedback: We designed our tools in close cooperation with end users applying a participatory design approach [16]. During several exploratory design workshops we introduced engineers to visual analytics techniques, discussed ideas, and fine-tuned possible solutions (system designs, features, etc.), and finally developed a basic concept. Next, we started developing interactive tools and provided a group of carefully chosen test engineers with frequent iterative releases and elicited feedback in personal meetings. In the case of Cardiogram, we used three different paper prototypes to evaluate early ideas, and a high-fidelity prototype before we came up with the final system presented in this paper. In doing so, we excluded alternative representation techniques and fine-tuned our approach of automatic data preprocessing (see below). A similar approach has been previously lead to success in the data analysis domain [20].

Think Aloud Protocols, Lab Studies and Field Studies: To evaluate the domain value of our tools we used both lab as well as field studies. We studied our early systems using think-aloud protocols and lab studies, investigating domain experts using our tools in artificial lab settings. While we received valuable feedback about the potentials and improvements of our tools, we did not gain insight into their usage under real-world conditions (cf. [23]). A major hindrance to gaining real-world insight on tool use was the missing close integration of our tools into current systems already in use at the company [29]. We report on how this hindrance was overcome with Cardiogram in later sections.

Informal Collaboration

Besides our formal studies, we found it invaluable for our successful integration to engage in frequent informal conversations with engineers. These conversations were, for instance, meetings at the company's cafeteria, a joint lunch, or casual discussions at the workplace. We improved our understanding of the various facets of our target domain, iteratively refined our requirements for visual analytics tools, and established successful collaborations. A drawback of informal conversations, however, is the restricted opportunity to log data for scientific rigor. In order to allow for some direct data

collection we always carried notepads in case spontaneous conversations would occur. Over the three year period we collected information from roughly 80 of these spontaneous encounters, with approx. 50 different engineers.

UNDERSTANDING DATA, TASKS, AND TOOLS

In the following, we summarize the understanding we gained of our domain and its data analysis challenges by using this variety of study methods. We organize our results by task, data, current data analysis tools, and a more detailed description of practices and challenges—all of them relevant parameters for designing visual analytics tools in this domain. It is meant both as such a description to enlighten our general understanding of real-life data analysis but also as a guide to practitioners in the field. Eventually, the results of our field analysis also yielded a fundamental source for deriving the design implications we present in the 'Discussion' part.

Main Task

The main task of the automotive engineers we studied was *finding errors in in-car sensor network communication* with the goal to verify a vehicle series safety, security, and functionality. For this purpose, 1.5 years before start of production the car manufacturer produces a test vehicle fleet of about 150 test vehicles. These vehicles are driven either by specialized test drivers or by the analysis engineers themselves in order to verify the correctness of in-car networks both in ordinary as well as extreme real-world driving situations. Each test vehicle is equipped with specific hardware recording messages transported on the network. This information together with a verbal or written error report is then transferred to the analysis experts of our target group. If an error has been detected during a test drive, it is the task of the analysis engineer to locate the error source by analyzing the recorded traces (see below) and to initiate further steps to solve the problem.

Data

All communication information logged during a test drive is stored in large text files called *traces*—a temporally ordered list of all messages enriched with entries for errors that either had been automatically detected and set (e. g., pre-specified error conditions in ECU specifications) or manually annotated (e. g., pressing an "error buzzer" in the test car & manual log annotation later). Traces can get tremendously large with the duration of a test drive and the amount of logged subcomponents (state of the art: approx. 50 million messages/hour for the entire network). These traces come in different formats (depending on the recording hardware used) and are not necessarily complete (e. g., in case of failure of recording hardware). In order to handle the enormous size of traces, engineers use *journals*—as specific, pre-filtered formats to reduce trace data to a few specific message types such as error frames or fault memory entries—and manually add information such as markers, triggers or predefined events.

Current Tools

To analyze trace files, several special-purpose analysis tools were in use by our engineers. The most important ones are an in-house tool called *Carmen* as well as the commercial tool *Canalyzer* (the functionality of these tools is very similar, for a state-of-the-art function description see [38]). Both of them were considered most relevant and powerful due to their scalability and compatibility to various data formats, and the wide availability of special-purpose plugins and data interpreters. Both tools are based on the combination of different digital modules that allow an individual configuration of a tailored measuring setup. Available modules include those for data loading, interpretation, filtering, or visual representation. Typical representation modules can show dynamically interpreted lists of messages, temporal signal plots, or rudimentary overviews. To analyze traces, our experts often used one or several of these tools in combination with general-purpose tools such as text editors.

Data Analysis: Practices, Problems and Challenges

Engineers typically spent their error analysis switching between data foraging and sense-making activities [36]. Using their analysis tools they typically attempted to (a) derive a hypothesis, (b) iteratively refine the hypothesis, or (c) dismiss the hypothesis and start anew. Based on their initial hypothesis, the analysts took different approaches to finding an error. If a clear hypothesis about the error source existed, engineers commonly started to check interpreted or even raw values directly. If the hypothesis was not solid from the beginning, the error description was rather vague or if the error source was estimated to be more complex, our participants preferably started with an overview using journals and then iteratively filtered and analyzed interpreted message lists and signal plots. A main difference to regular software debugging was the inability to reproduce errors through experimentation as there is (currently) no opportunity for analysts to directly change the software of ECUs or sending additional messages to bus systems: their only means of discovering the sources were traces and journals from test runs.

Our studies showed that engineers had to track errors of varying degrees of complexity which tremendously influenced the process of finding these errors in terms of processing time, costs, and engineers involved. Simple errors outnumbered complex errors but finding and solving complex errors was often a tedious, lengthy and highly collaborative undertaking. One engineer commented *"I can track down a simple error in several minutes, but solving a complex problem can take weeks or even months"* (quote from a contextual interview, translated from German). In close cooperation with engineers we identified three main origins of complexity:

Traceability: Finding the actual source of an error proved difficult due to external circumstances such as temperature, extreme driving situations, or incorrectly specified error conditions. For example: After ending a test run, our engineers detected that all car windows would open unexpectedly. Engineers spent several weeks analyzing and trying to reproduce this error. The actual reason was a specific test case in which all four doors were simultaneously slammed shut which activated an overpressure sensor that opened the windows.

Dispersion: Many highly distributed and inter-related hard- and software systems exist in a vehicle and errors often propagate over several systems before they are automatically detected and logged. The interplay between two or more intrinsically and separately correct subsystems can lead to complex, unpredictable errors. The more dispersed errors or involved systems are, the higher the chance that they might be complex. This is also true for the above example where the

actual reason was not located in the window system but in the accident security system.

Degree of trace preparation: Not every bus system or recording hardware supports journals to reduce and abstract the recorded data. Without any abstraction it can become complex and laborious to analyze message traces especially when exact error timings are not available, for example for manually recorded errors. Additionally, engineers have to be aware of the fact that measurement hardware can be the source of errors and inaccuracies in the data.

As a result of these complexities, our engineers relied on an array of different tools. Along with Carmen and Canalyzer, one engineer used another 12 tools during a 1h analysis session. He liked all his tools but was burdened by the additional work of switching between them: *"Analyzing alone is a complex task, but handling the entire overhead of using so many incoherent tools is overkill [...] each tool is a valuable part of my work but they are not well coordinated and integrated, this means a lot of additional, redundant work to me [...]"*.

CARDIOGRAM

In the following, we introduce Cardiogram, a novel tool we designed based on our current-practice field studies and on the lessons learned from our previous endeavors [27, 28, 30]. We integrated the tool into daily practices of our target users and tested it under real working conditions.

Data Analysis Requirements

The main analytics objective of Cardiogram was twofold: (a) supporting engineers in handling the masses of trace data and (b) providing new insights into temporal, logical, and behavioral aspects within the data.

1. Handling the masses of data: In our studies, we found that handling the masses of trace data is a huge practical problem. While data storage is no longer a pressing problem, analyzing all relevant data in detail is nearly impossible for engineers. In collaboration with our target users, we derived two requirements to encounter this challenge. First, *automated filtering of data* might be helpful as often only a very small subset of the data is involved in the error finding process. The reduced data can then be used as input for traditional and novel representation techniques. Second, *support for automated error detection* is required and was frequently requested by our engineers. Automatic error detection would help our target users to reduce the amount of traces to analyze but also to quickly inform initial hypotheses about errors.

2. Providing new insights: Our study also revealed that most current data representations were based on lists and simple signal-value plots. Beyond the capabilities of these tools, engineers needed to analyze timing aspects and message propagation, detect outliers and see correlation between messages and mechanical behavior. In particular, engineers needed visual support for *correlating the temporal and the logical layer* of trace messages to better understand the influence of message timing and physical behavior, and vice versa. Current techniques could not support this requirement. In addition, *novel overview* and *data abstraction techniques* are required to support better understanding of comprehensive correlations and functional dependencies in the data. Journals (see above) are a rudimentary approach for this requirement, but often they are not available depending on the used recording hard- and software, and—when available—are restricted to a very high level view onto traces. However, understanding comprehensive aspects on various levels of traces is essential for the detection of complex errors but as we saw, extremely difficult to retrieve directly from raw data or journals.

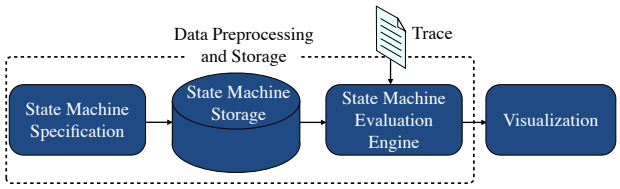

Figure 2: Workflow and components of Cardiogram.

Design

Based on these requirements, we designed Cardiogram as two major software components: (1) A *Data Preprocessing and Storage Component* for automatic data filtering, error detection and data abstraction and (2) a *Visualization Component* for overviews of data preparation results and for gaining insight into correlations between time and logic. In terms of the sensemaking loop [36], the first component was meant to mainly aid in data foraging activities, while the visualization component was meant mainly for sense-making activities; together they allowed the engineers to switch between different activities involved in sense-making and to derive solutions to their analysis questions. Figure 2 gives an overview of the workflow in Cardiogram and shows the two basic components and its subcomponents. In the following, we describe these components in more detail:

Data Preprocessing and Storage:

This unit is responsible for the automatic preparation, abstraction and filtering of traces. To realize this, we used a model-based analysis approach utilizing state machines which was suggested for automotive system testing [2]. Our approach of using state machines for testing purposes allows for (a) reusing knowledge from software design processes, (b) specifying models for complex test cases, and (c) for automatically running these models. Our Data Preprocessing and Storage Component is made up of three subcomponents:

State Machine Specification: In order to make semantic trace interpretations automatically testable, we provide engineers with the opportunity to specify vehicle behavior in state machines. To do so, they can use a textual or a graphical editor, including an Eclipse plugin we designed, to define a set of logical vehicle states such as *"window front left open"*, transitions between states such as *"open window"*, and what events in a trace result in which transitions, e. g., *"message xy opens window"*. To test predefined conditions, each state can be additionally annotated as *error*, *warning*, or *okay*. Two main kinds of state machine designs can be specified: Verification State Machines and Context Information State Machines. Verification State Machines test a predefined situation, e. g., if a specific error condition is met or not. Context Information State Machines are more general and represent generic vehicle information such as mechanical activities (e. g., monitoring window opening) or ECU behavior. In the following, we refer to both simply as state machines.

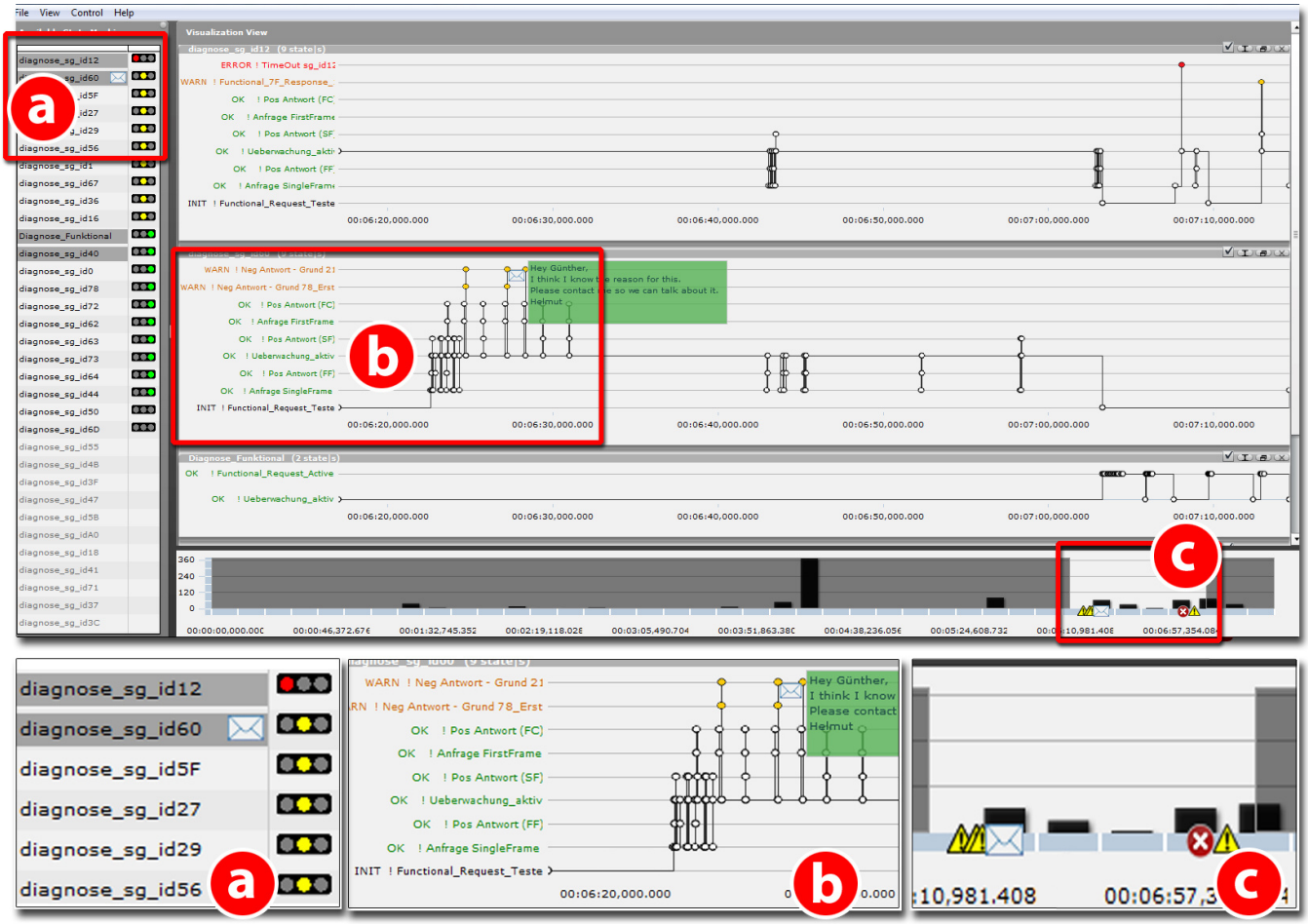

Figure 3: Screenshot of Cardiogram Visualization: (a) State Machine View showing all tested state machines ordered by relevance for bugfixing; (b) Visualization View with several detailed state/time plots showing transitions via vertical and horizontal lines and additional glyphs at target states; (c) a combined range slider/ overview bar showing the sum of all transitions within discrete time intervals.

State Machine Storage: Our target user group is located in a large company setting where thousands of employees work on highly specified tasks. To form a greater understanding based on individual expertise and to master comprehensive data analysis challenges, collaboration is indispensable. In a next step, we therefore opted for a central storage of the state machines and their descriptions in a database using a common XML format. This approach was targeted specifically at complex error analysis. By making descriptions available to all engineers they can avoid repetitive and redundant work. Previously, engineers had to spend considerable time converting data manually using different import/export formats and they specifically requested a new approach which would alleviate this burden.

State Machine Evaluation Engine: This automatic trace preprocessing unit is the heart of Cardiogram and supports the analysts by loading specific state machines, importing traces, and testing these traces with the specified state machines. Using all loaded state machines, the State Machine Evaluation Engine queries over one or more traces, identifies all occurrences of predefined transition patterns, and stores them to a temporal ordered list of transitions (similar to temporal data analysis techniques such as PatternFinder [7]). For each analyzed state machine, an additional global tag, called *aggregated state machine result*, indicates the occurrences of *error* and *warning* states which can be used for later analysis. The engine thus reduces and pre-processes the data to filtered and interpreted version of the trace(s) at hand—filtered and interpreted by the vehicle behavior the engineers had specified in state machines before. This step makes novel overview techniques possible which were particularly missing from previous tools but crucial for complex error detection.

Visualization:
The main purpose of Cardiogram's visualization (cf. Figure 3) is to support the exploration of errors, warnings, or other hints which may require further inspection because they cannot be automatically detected. In such situations, Cardiogram visualizes all state transitions of all tested state machines and helps to provide insight into incorrect vehicle states and into timing correlations between state machines, i. e., between specified vehicle behavior. Together with the three data preparation steps presented above, this visualization allows engineers to gain a novel perspective on complex dependencies of in-car networks and correlate logical with timing aspects.

Cardiogram's visualization is centered around a zoomable timeline, multiple coordinated views, and the use of consistent representation and interaction techniques. Our solution was inspired by work on previous trace visualization design studies (most importantly [10] and [24]). Similar to these solutions we chose to focus on the temporal aspect of trace data. The general setup of the visualization consists of three views, (1) the State Machine View providing an overview over all tested state machines, (2) the Visualization View allowing for investigating the timing of transitions, and (3) a Navigation/Overview Bar at the bottom.

The design of the visualization was driven by several needs uncovered during the field evaluation period:

Prioritization: The *State Machine View* (Figure 3-a) lists all state machines tested on a specific trace. It provides the contextual overview which previous tools were lacking. As requested by our target group, this list is sorted according to priority based on aggregated state machine results (first those with at least one *error* state, then the ones with at least one *warning*, etc.).

Familiarity and Fast Access: Even though our target group of engineers did not have the same academic or work background, we found that several different types of network visualizations, charts, and diagrams were in use amongst them and familiar to all. In order to support the interchange and collaboration between these engineers we decided to make use of elements from these familiar representations. We used commonly used traffic light icons next to each state machine entry in the State Machine View (Figure 3-a) to encode the aggregated state machine outcomes *error*, *warning*, *okay* using the colors red, yellow, green, and no color coding for other outcomes. By selecting a state machine from the State Machine View a similarly familiar state/time plot is shown in the Visualization View showing all transitions of this state machine according to its transition table (Figure 3-b and context). Detailed information about transitions can be retrieved by zooming in and/or hovering the mouse pointer over these dots. Note that by zooming also nearly time-equivalent transitions—appearing as vertical lines in the overview—can be resolved to their correct temporal order. Fast access to the underlying trace file is given by an integrated backlink to a traditional list presentation of the trace. Engineers were used to working with hexadecimals and regularly had to check single bytes and bits during their work. For them, raw data must always be ready at hand in order to immediately prove or discard hypotheses based on raw values. "Fast access to raw data" was one of our most requested requirements.

Time Reduction: In addition to allowing the user temporal navigation, the navigation/overview bar at the bottom of the visualization provides an overview over all transitions of selected state machines (Figure 3-c). According to a time slot size pre-defined by the user, transitions of selected state machines are subsumed and bars are added which encode the overall number of transitions in a time slot. This provides an abstracted indication about busy, calm, steady and void areas. Changes into *error* and *warning* states are indicated not only with red and yellow dots in time/state plots but also with accordingly colored, domain specific symbols (see "familiarity" above) in the overview bar. This supports fast readability of transitions and can be relevant for bugfixing.

Collaboration: Interactive annotation of the data is also supported. Each analyst can freely attach colored notes directly into a state/time plot as was done in area (b) of Figure 3. Once an annotation is set, it is displayed at the exact timestamp and state it was created at. Symbols for notes are also shown in the state machine list and in the overview bar where they indicate the position within the global timeline. These annotations can be exported together with the data and sent to a colleague for further inspection or inquiry. This supports the collaborative requirements of the large company context and helps engineers to avoid repetitive work.

Convenience Features: Based on engineers' requests during our user-centered design process, we integrated a variety of other interactive features, including: keyboard shortcuts for all features (based on familiar tools), unrestricted vertical scaling of state machine plots to provide additional overviews, minimization and closing of state machine plots, drag&drop positioning of the state machine plots and state lines to allow side-by-side comparison and new perspectives on the data, dynamic adding or subtractions of state machines to or from the overview bar, and the free configuration of nearly all system features and settings.

Integration

We spent considerable effort to closely integrate our modules with a comprehensive in-house analysis software environment, Carmen, which supports valuable back-end features such as data storage, interpretation, filtering and traditional views (e. g., message lists). During our current-practice analysis we found that many of the tools used by engineers were already integrated into this platform and therefore were powerful in terms of flexibility, compatibility, scalability, and in providing specific features for specific problems. Re-implementing all of these features for our tool would not have been possible within realistic time and budget requirements. Instead, we opted for a closely integrated solution which could be immediately used by engineers in the context of their familiar work environments and would allow them to take advantage of already supported data formats, and be combined with conventional solutions without any extra costs. Using this approach, Cardiogram has achieved wide use by our target users as unnecessary data conversions and repetitive tasks to use our tool were avoided. Integrating Cardiogram with Carmen, on the other hand, also helped us to study it under realistic conditions.

CARDIOGRAM EVALUATION

We evaluated Cardiogram in a long-term, qualitative study with expert test users rather than using quantitative measures and statistical analysis that fail to provide insights into the exploratory nature of analysis tasks [4, 23, 32]. Analysis of Cardiogram was conducted in two phases for (1) the data preprocessing and storage and (2) the visualization component. To test our novel automation and abstraction approach using state machine data preparation, we first implemented the state machine editor, database and evaluation engine (cf. Data Preprocessing and Storage components in Figure 2) and integrated it as a module in Carmen. Together with a textual representation of statistical results (number of errors/warnings

occurred) and transition lists—similar to current trace practices of engineers—for detailed inspection on demand. This approach was validated in a 12-month field study with 15 domain experts—which is comparatively long by visual analytics standards. Over these 12 months, experts used the tool during their daily activities, created state machines on their own, and included the tool in their data analysis procedure. The usage length of such a single, situational session varied from several minutes up to three hours. During bi-weekly meetings with these experts we discussed their experiences with our tool and elicited feedback on their benefits and areas for improvement. As a second step, we qualitatively evaluated the Cardiogram visualization with six domain experts during a one-hour session in which they used the visualization on their own data and/or on test data sets we provided. Additionally, we received feedback from two test users who used Cardiogram for an eight week period. The results of the study uncovered several main categories of benefits for our new approach which aided in adoption and acceptance of the tool:

Results of Evaluating the Data Preprocessing Approach

Externalization of Expert Knowledge: Analysis experts created state machines to capture their expertise for verification and abstraction of complex behavior. Many of the state machines were specified to reproduce highly distributed procedures such as booting a car, starting the motor, or shutting down the vehicle. These state machines included up to 25 different states as well as clusters of sub-state machines. Each state in turn abstracted up to 15 signals in order to form combined and interpreted information about specific vehicle behavior. Externalizing this knowledge into state machines made it widely available for other engineers who benefited even without specific knowledge about this particular behavior.

Mass Analysis Instead of Sample-Tests: Our abstraction and automation techniques facilitated a broad analysis of a great number of traces. One engineer used the core components to automatically analyze 12,000 traces with 50,000 messages on average within one day. Based on the global result tags of state machines, he could isolate three important traces which he examined in more depth to verify a specific hypothesis. Previously, testing of this data relied on analyzing and debugging samples of the data, our approach however allowed the analysis of hundreds to tens of thousands of traces, and to test or verify hypotheses on a broad testing basis.

Results of Evaluating the Visualization

Correlation Between State Machines: All of our participants stated that the Cardiogram visualization was enormously helpful to understand and explore correlations between dependent state machines. For example, we saw the Cardiogram visualization being used to explore several parallel procedures involved in shutting down a car. Correct shutdowns are of high importance as errors can lead to high consumption of electricity and load on the car's battery. Shutting down a car first involves the shutdown of all relevant subsystems. One of our participants used a set of 12 state machines together with our visualization and verified the shut-down behavior of all sub-systems as well as that of the entire car. Using the visualization allowed him to compare timings, to verify correctness of temporal order, and to correlate the transitions of the state machines. This in-depth analysis was not possible previously or with the core components alone.

Trace-Related Overview of State Machine Activities: Four of our participants mentioned that the Cardiogram visualization provided a good overview over all trace-related, logical, and temporal activities. The list of all state machines showed valuable information to them on which tests had been conducted and whether they had been successful or erroneous. Freely selecting, combining, and repositioning of state machines helped them to further explore erroneous state machines and to derive correlations between them. Additionally, three participants mentioned that when zooming out, the time line provided a valuable overview over a state machine's global activities and helped to quickly detect transition peaks.

Verification and Re-Engineering of State Machines: Two of our participants used the Cardiogram visualization to verify state machines currently under construction. Loading these state machines together with a known test trace helped them to validate the correctness of transitions and to estimate possible interaction with other state machines.

Summary of Results

In summary, the analysis of the Cardiogram core components and visualization showed that the tool successfully supported the engineer's analysis requirements and addressed known challenges, most importantly it provided novel overviews and abstractions of the data, semi-automated analysis and collaborative features and, thus, allowed the engineers to avoid repetitive work. Cardiogram's abstraction and automation techniques were valuable tools for reducing and preprocessing the data and Cardiogram's visualization allowed for the analysis of comprehensive and global aspects. At the time of writing, the tool has spread within the company and is now used by more than 30 engineers on a daily and weekly basis. We recently transferred our software to the Carmen tool developers who are now extending our solutions. Extensions will include: fine-tunig of some visual encodings (e. g., larger symbols for errors and warnings in state/time plots), increasing the scalability of the visualization component (currently it can handle files with up to 100.000 transitions), and direct embedding of the tool into Carmen's core components.

DISCUSSION: RECOMMENDATIONS AND ADOPTION

While working with our domain experts and on Cardiogram we gained an in-depth understanding of the field and its visual analytics requirements. In particular, we uncovered several recommendations for developing visual analytics tools in this domain and gained insight into making them accepted and used with experts who may have had no previous experience with visual analytics tools. In the following we summarize both, the design implications that we have derived from our exploratory field analysis and from Cardiograms's design and evaluation, as well as the main aspects that we think have led to the successful adoption of Cardiogram. Some of these aspects have also been shown in other papers and domains and therefore underline their general importance (we cite some below). Others are more unique and specific for our domain. In the following we list all findings in order to provide clear and comprehensive guidance for other VA researchers in this or similar areas.

Design Implications

The following recommendations were collected for and used in the design of Cardiogram.

New Perspectives on Complex Errors: The detection of complex errors required dedicated data representations to show correlations between error sources. In particular, we uncovered a need for:

Visual overview techniques: to allow for better analysis of traces without journals and gaining insight into comprehensive aspects along traces (see also, e. g., [31]).

Perspectives beyond raw data and signal plots: No current technique supported the analysis of timings, the detection of outliers, correlations between messages, mechanical behavior, electronic data, and message propagation. Especially understanding correlations between the temporal (when has a message been sent) and the logical layer (who sent it, who received it, what software components were involved, etc.) was highly important for engineers. Not properly supported yet, all these aspects are invaluable to get a profound understanding about a trace and eventually to detect errors.

Multiple, modular, and coordinated solutions: Our group of engineers required multiple different perspectives on the data to detect complex errors (see also, e. g., [9]). Which perspectives were most relevant relied on an engineer's knowledge, preferences, as well as the underlying problem. Therefore, an unrestricted and modular combination of perspectives is instrumental. Perspectives should support coordination over time and data linking according to known techniques, but also the opportunity to work without coordination (e.g., for comparing behavior at different time stamps).

Fast access to raw data: Engineers were used to working with hexadecimals and regularly checked single bytes and bits during their work. Fast access to the familiar raw data representation to check, prove, or discard hypotheses on the sources of errors was of highest importance for them.

Handling the Masses of Data: Handling the masses of data produced in automobile testing is a huge practical challenge (see also [15]). The following approaches were successfully applied by us in Cardiogram:

Data abstraction and automated filtering: Understanding comprehensive aspects in traces is essential for complex error detection but difficult to retrieve directly from raw data. Novel data abstraction techniques were required for reproducing behavioral aspects, comprehensive correlations and functional dependencies in the data.

Support for automated error detection: Current analysis procedures rely on sample testing and a lot of recorded traces are never analyzed. Automation of this process helped engineers (a) to rapidly test a set of hypotheses, (b) to speed up the detection of common errors, and (c) to allow analyzing much more data than is achievable via manual inspection.

Avoid repetitive work and unnecessary iterations: Due to the size and complexity of recorded trace data, engineers used a large array of tools that each only supported parts of the analysis. Converting data manually is tedious and annoying and hinders the acceptance of novel solutions.

Engineer-centered Solutions: Our understanding of the work environment helped us to design solutions which closely matched the work process of our engineers.

Familiarity: Our target users had a broad background and knowledge of various data representation techniques, most importantly statistical graphics such as line plots. Supporting these well-known mental models helped to support communication with and between engineers and increasing the acceptance of new tools.

Support collaboration: Our target group was located in a large company setting with thousands of specialized employees. To form a greater understanding based on individual expertise and to master comprehensive challenges collaboration is indispensable and, therefore, should be actively supported in data analysis tools.

Adoption

Overall, the recommendations from our exploratory field analysis together with a user-centered design process helped us to design a tool that found its way into everyday routines of our analysis experts. We learned that the core ingredients to a successful deployment of our visual analytics technique in this industrial setting were (a) the approach's simplicity, (b) strong user integration throughout the entire design process, and (c) most importantly a tight integration into existing tools and workflows. Especially, with a focus on simple solutions and tight integration we deviated from our earlier visualization prototypes which tended to be feature-rich, but were never properly connected to support real-world problem solving in real environments with real data. In the following, we summarize these three main findings:

Simplicity: Our target users had demanded tools that *"simplify [my] work, not complicate it with intricate visualizations that have to be learned upfront."* Preferred were solutions with high automation and simple, easy to understand representations with an immediately apparent benefit that were explicitly tailored to their needs.

User Integration: Engineering is a complex area that requires expertise and background knowledge. As outsiders to the area we found it invaluable to counterbalance our little domain knowledge through an exploratory study of analysis practices and a user-centered design processes. This helped us to implement tailored, well-directed, and valuable solutions (see also, e. g., [20] or [21]).

Tool Integration: We also learned that in our domain tight integration of final tools with domain data and process is a crucial factor to success, adoption and an essential requirement for us to evaluate the value of our tools under real circumstances. In our case, integrating Cardiogram as a module into Carmen—a traditional and accepted tool—helped engineers to easily access the data at hand, use other common modules and draw upon multiple perspectives (see above). Especially in large companies, often a set of traditional tools already exist for a data analysis problem at hand. These tools are usually integrated into a well-defined process and re-implementing them for a research project is too expensive. However, neglecting these practices can pose additional costs for employees, for instance, due to additional data transformation efforts or tool-flipping costs. In our case, it turned out that such additional time costs were not acceptable for engineers and poorly integrated tools were disapproved of in daily work.

CONCLUSION AND FUTURE WORK

In 2006, Broy [3] wrote: "The increase of software and functionality in cars is not close to an end. In the contrary, we can expect a substantial growth in the future". This holds unchanged even four years later and is probably the strongest motivation behind the work presented. In this paper, we focused on an in-depth analysis of this application area from an VA researchers point of view. We provided results from an exploratory field study, and a novel VA tool, Cardiogram, combining data pre-processing and visualization techniques, and its evaluation in the field. We also explained and discussed our design approach and showed how it finally led to the acceptance of Cardiogram in our target domain. While our experience so far is solely based on working in our target domain, we hope that our findings can serve as reference for others who are planning to closer integrate their VA tools with real-world application domains, especially in large companies and technical environments. We encourage other researchers to report their own experience in order to broaden our general understanding about such projects and to help VA "move from research to practice" [36].

REFERENCES

1. R. Bosch GmbH. *Kraftfahrzeugtechnisches Taschenbuch*. Vieweg&Sohn Verlagsgesellschaft, 26 edition, 2007.
2. E. Bringmann and A. Krämer. Model-based Testing of Automotive Systems. In *Proc. Software Testing, Verification, and Validation*, pages 485–493. IEEE, 2008.
3. M. Broy. Challenges in automotive software engineering. In *Proc. Software Engineering (ICSE)*, pages 33–42. ACM, 2006.
4. S. Carpendale. Evaluating information visualizations. In A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, editors, *Information Visualization: Human-Centered Issues and Perspectives*, pages 19–45. Springer LNCS, 2007.
5. K. Cook. From research to reality: Visual analytics technology transition. In *VAC Views*. PNNL, Nov. 2008.
6. H. Doleisch, M. Mayer, M. Gasser, R. Wanker, and H. Hauser. Case study: Visual analysis of complex, time-dependent simulation results of a diesel exhaust system. In *Proc. EuroVis*, pages 91–96, 2004.
7. J. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *Proc. VAST*, pages 167–174. IEEE, 2006.
8. H. Heinecke. Automotive system design-challenges and potential. In *Proc. Design, Automation and Test (DATE)*, pages 656–657. IEEE, 2005.
9. N. Henry and J. Fekete. Matrixexplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):677, 2006.
10. D. Holten, B. Cornelissen, and J. van Wijk. Trace Visualization Using Hierarchical Edge Bundles and Massive Sequence Views. In *Proc. of Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 47–54, 2007.
11. K. Holtzblatt and S. Jones. Contextual inquiry: A participatory technique for system design. In *Participatory Design: Principles and Practices*, pages 177–210. Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1993.
12. P. Isenberg, A. Tang, and S. Carpendale. An exploratory study of visual information analysis. In *Proc. CHI*, pages 1217–1226. ACM, 2008.
13. P. Isenberg, T. Zuk, C. Collins, and S. Carpendale. Grounded evaluation of information visualizations. In *Proc. BELIV*, pages 56–63. ACM, 2008.
14. C. Johnson, R. Moorhead, T. Munzner, H. Pfister, P. Rheingans, and T. Yoo. *NIH-NSF Visualization Research Challenges Report*. IEEE Press, 2006.
15. D. Keim, E. Bak, P.and Bertini, D. Oelke, D. Spretke, and H. Ziegler. Advanced visual analytics interfaces. In *Proc. AVI*, pages 3–10. ACM, 2010.
16. F. Kensing and J. Simonsen. MUST: A method for participatory design. *Human-Computer Interaction*, 13(2):167–198, 1998.
17. R. Kincaid. Signallens: Focus+context applied to electronic time series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):900–907, 2010.
18. S. Koch, H. Bosch, M. Giereth, and T. Ertl. Iterative Integration of Visual Insights during Patent Search and Analysis. In *Proc. VAST*, pages 203–210. IEEE, 2009.
19. Z. Konyha, K. Matkovix, D. Graxanin, M. Duras, J. Juric, and H. Hauser. Interactive visual analysis of a timing chain drive using segmented curve view and other coordinated views. In *Proc. CMV*, pages 3–15. IEEE, 2007.
20. P. McLachlan, T. Munzner, E. Koutsofios, and S. North. LiveRAC: Interactive visual exploration of system management time-series data. In *Proc. CHI*, pages 1483–1492. ACM, 2008.
21. M. Meyer, T. Munzner, A. DePace, and H. Pfister. Multeesum: A tool for comparative spatial and temporal gene expression data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):908–917, 2010.
22. T. Munzner. A nested process model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009.
23. C. Plaisant. The Challenge of Information Visualization Evaluation. In *Proc. AVI*, pages 109–116. ACM, 2004.
24. A. Pretorius and J. Van Wijk. Multiple views on system traces. In *Proc. PacificVis*, pages 95–102. IEEE, 2008.
25. A. Pretschner, M. Broy, I. Kruger, and T. Stauner. Software engineering for automotive systems: A roadmap. In *Future of Software Engineering*, pages 55–71. IEEE, 2007.
26. M. Schulz, F. Reck, W. Bartelheimer, and T. Ertl. Interactive visualization of fluid dynamics simulations in locally refined cartesian grids. In *Proc. VIS*, pages 413–416. IEEE, 1999.
27. M. Sedlmair, C. Bernhold, D. Herrscher, S. Boring, and A. Butz. Mostvis: An interactive visualization supporting automotive engineers in most catalog exploration. In *Proc. Information Visualisation (IV)*, pages 173–182. IEEE, 2009.
28. M. Sedlmair, W. Hintermaier, K. Stocker, T. Büring, and A. Butz. A dual-view visualization of in-car communication processes. In *Proc. Information Visualization (IV)*, pages 157–162. IEEE, 2008.
29. M. Sedlmair, P. Isenberg, D. Baur, and A. Butz. Evaluating Information Visualization in Large Companies: Challenges, Experiences and Recommendations. In *Proc. BELIV*. ACM, 2010.
30. M. Sedlmair, B. Kunze, W. Hintermaier, and A. Butz. User-centered Development of a Visual Exploration System for In-Car Communication. In *Proc. Smart Graphics*, pages 105–116. Springer, 2009.
31. B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proc. Visual Languages (VL)*, pages 336–343. IEEE, 1996.
32. B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proc. BELIV*. ACM, 2006.
33. J. Stevens. Visualization of Complex Automotive Data. *IEEE Computer Graphics and Applications*, 27(6):80–86, 2007.
34. L. A. Suchman. *Plans and Situated Actions*. Cambridge University Press, 1987.
35. T. Tekušová, G. Darmstadt, and T. Schreck. Visualizing Time-Dependent Data in Multivariate Hierarchic Plots-Design and Evaluation of an Economic Application. In *Proc. Information Visualization (IV)*, pages 143–150, 2008.
36. J. Thomas and K. Cook. *Illuminating the path: The research and development agenda for visual analytics*. National Visualization and Analytics Center, 2005.
37. M. Tory, S. Staub-French, B. Po, and F. Wu. Physical and Digital Artifact-Mediated Coordination in Building Design. *Proc. CSCW*, 17(4):311–351, 2008.
38. Vector Informatik GmbH. http://www.vector.com/, 03/10.