A Dual-View Visualization of In-Car Communication Processes

Michael Sedlmair, Wolfgang Hintermaier, Konrad Stocker, Thorsten Büring, Andreas Butz
BMW Group Research and Technology, University of Munich, Germany
{Michael.Sedlmair, Wolfgang.Hintermaier, Konrad.Stocker}@bmw.de
{Thorsten.Buering, Andreas.Butz}@ifi.lmu.de

Abstract

With the increasing complexity of in-car communication architectures, their diagnostics have become essential for automotive development and maintenance. In order to help engineers to detect and analyze the potential sources and consequences of errors, it is crucial to provide both comprehensive and detailed insight into the communication processes and their contexts. Two important aspects of these are the dependencies and correlations between onboard functions. In this paper we present a dual-view visualization for exploring the functional dependency chains of in-car communication processes. One view presents the dependencies of hardware components using a space filling approach similar to a treemap, whereas the other view displays the functional correlations as an interactive sequence chart. The views are coupled via color coding and show the dependencies of an interactively selectable functional unit. In an expert evaluation, we assessed the benefits of using this visualization technique for in-car communication diagnostics with very positive results.

Keywords: Multiple Views, Information Visualization, In-Car Communication, Automotive, Diagnostics

1. Introduction

Modern automobiles use complex hard- and software architectures to meet today's customer requirements, particularly in the premium automobile sector. Vehicles presently contain up to 70 *electronic control units* (ECUs), which are linked via several bus systems and protocols, including *controller area network* (CAN), *FlexRay* or *media oriented systems transport* (MOST), as well as gateways, which connect the different bus systems to form one large *physical network*. The enormous growth of these automobile communication architectures in the past was mainly caused by innovations in driver assistance and safety applications, which demand highly networked functions within the automobile. The ECUs in turn contain numerous (logical) *functional blocks* (FBs) in the form of software components to provide the desired functionality. Several FBs together can realize more complex functions, such as electric window lifting or climate control. On the other hand, complex functions are not necessarily implemented within a single ECU but can also be realized by several FBs distributed over different ECUs. This logical network of FBs is called the *functional network*. In summary, the communication architecture of modern premium automobiles consists of a complex physical network, which is used by a complex functional network for message and data exchange. This communication architecture is shown in figure 1.

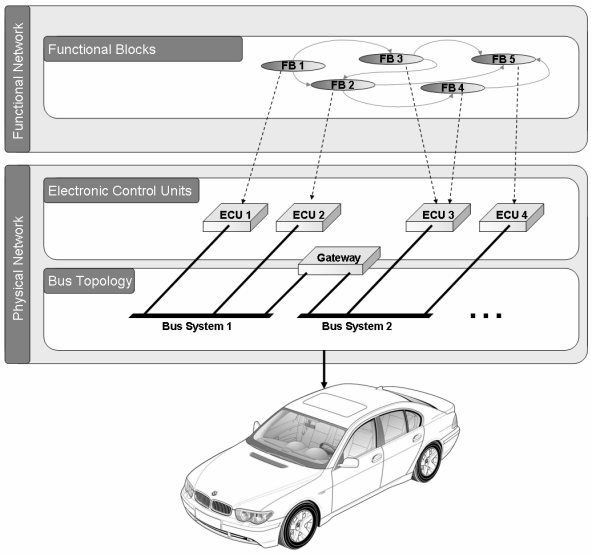


Figure 1: Concept of a modern automobile communication architecture.

Traditionally, this complex architecture was described in a static table. With increasing size, this presentation became unreadable for most practical matters and currently requires a very high degree of expert knowledge to decode. We therefore designed a visualization tool to reestablish an understanding of in-car communication processes and make them accessible to a wider audience, based on the insight that operations within an automobile can only be understood by understanding the communication relations behind them.

2. Technical motivation

During automotive system development, integration and maintenance processes, fault diagnosis and root cause analysis are continuously used in order to find and correct problems. They become particularly challenging when dealing with a large number of different variants of a car. For current cars, there are usually many different variants due to optional equipment and country-specific configurations. The physical architecture (i.e., number of ECUs and hence number of bus subscribers) can, for example, substantially vary between a car equipped with an adaptive cruise control and one without. As a consequence, the number FBs varies heavily. Additional ECUs, e.g., for optional equipment, bring along new FBs, and in already existing ECUs, further FBs are activated to provide the desired functionality. The theoretical "maximum car" containing all available optional equipment and the country-specific components of all countries at once provides an upper estimate for the available FBs within a specific existing car variant to be analyzed. In addition to *active FBs* the ECUs in any customer's car will presumably also contain *inactive FBs*, because some FBs are only activated with certain optional equipment.

Analyzing a specific vehicle variant means identifying all possible functional *dependency chains*, i.e., the transitive FB communication over several involved ECUs. The functional dependency chain with focus on a *Trigger-FB* A consists of all successor FBs reachable by a communication from A and their successors in turn (FBs of the transitive closure forward), plus all predecessors communicating with A and their predecessors in turn (FBs of the transitive closure backward). Figure 2 shows an example of a communication situation in which the Trigger-FB has a cloud of successors as well as predecessors.

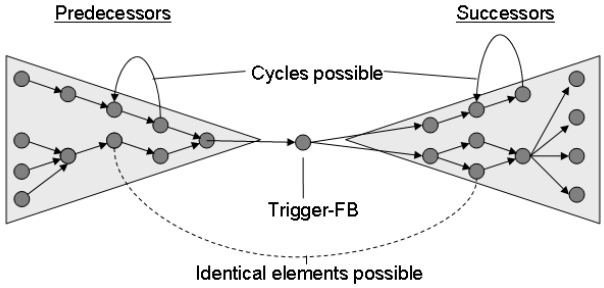


Figure 2: Schematic representation of a dependency chain around a Trigger-FB.

To see why all possible dependency chains are needed, let's consider the following scenario: If a problem occurs in a specific component (FB), this problem may lead to faulty messages sent to successor FBs which may produce problems themselves in turn, or just pass on the faulty message. One of the successor FBs (not necessarily the next in line) will eventually indicate the problem. Starting at this point, the developer has to identify the root cause. Without the set of all predecessors it would be necessary to test all components of the car in a brute force manner to find the actual root. Traditionally, developers used a sophisticated system involving traces and logfile data to analyze the situation. This approach eventually comes to a limit with more and more variants of cars produced. It is time-consuming and error-prone. The proposed new approach visualizes the communication between FBs in a much more understandable manner without sacrificing detail.

3. Related Work

The application presented in this work is based on several information visualization concepts:

Multiple coordinated views (MCV) use two or more distinct views to support the exploration of complex data. Each view focuses on different aspects of the information space, while coordination between the windows allows discovering relationships based on user interaction [10]. For instance, a typical configuration for an analysis tool may be to present data as a scatterplot along with two separate list and histogram views. Selecting items in one view then causes the items to be highlighted in all other views. This technique is called brushing [2]. Other common forms of coordination are the synchronization of navigation (e.g., between a map browser and an overview window) and item selection from an overview for presenting details in an adjacent window (e.g., selecting an email from a header list). MCV systems have been found to improve user performance in a variety of scenarios (e.g., [10, 4, 9]). However, drawbacks of the MCV approach include the increased display requirements and the effort for the users to constantly switch between different views [3]. Moreover, the effectiveness of the system depends heavily on the layout and interaction design. A guideline to support MCV developers is given in [1].

Treemaps [7, 12] are a representation technique for hierarchical data. Each node of the data is mapped to a rectangular display area by recursively partitioning the screen into 2D boxes. In comparison to conventional node-link diagrams, treemaps utilize the entire screen and thus are considered to provide superior scalability. At the same time the box representation of nodes facilitates additional encoding via color and shading. While non-nested treemaps are less effective for conveying the topology of a tree, they are particularly useful when the focus of the visualization is on the leaf nodes and their attributes [17]. Various algorithms have been developed to further improve the map layout, for instance, to allow for easier comparison of node sizes [13].

Focus and context approaches such as fisheye views [5] have been invented to support user orientation when navigating large information spaces. One of the first techniques proposed was the bifocal display [14], which shows a virtual workspace with color-coded documents on a horizontal strip. The screen is split into a centered detail region, and two context regions in which the documents are visually compressed to small-sized vertical bars. The users can view the documents in full size by moving the corresponding portion of the workspace into the detail region. Due to the distortion applied, the

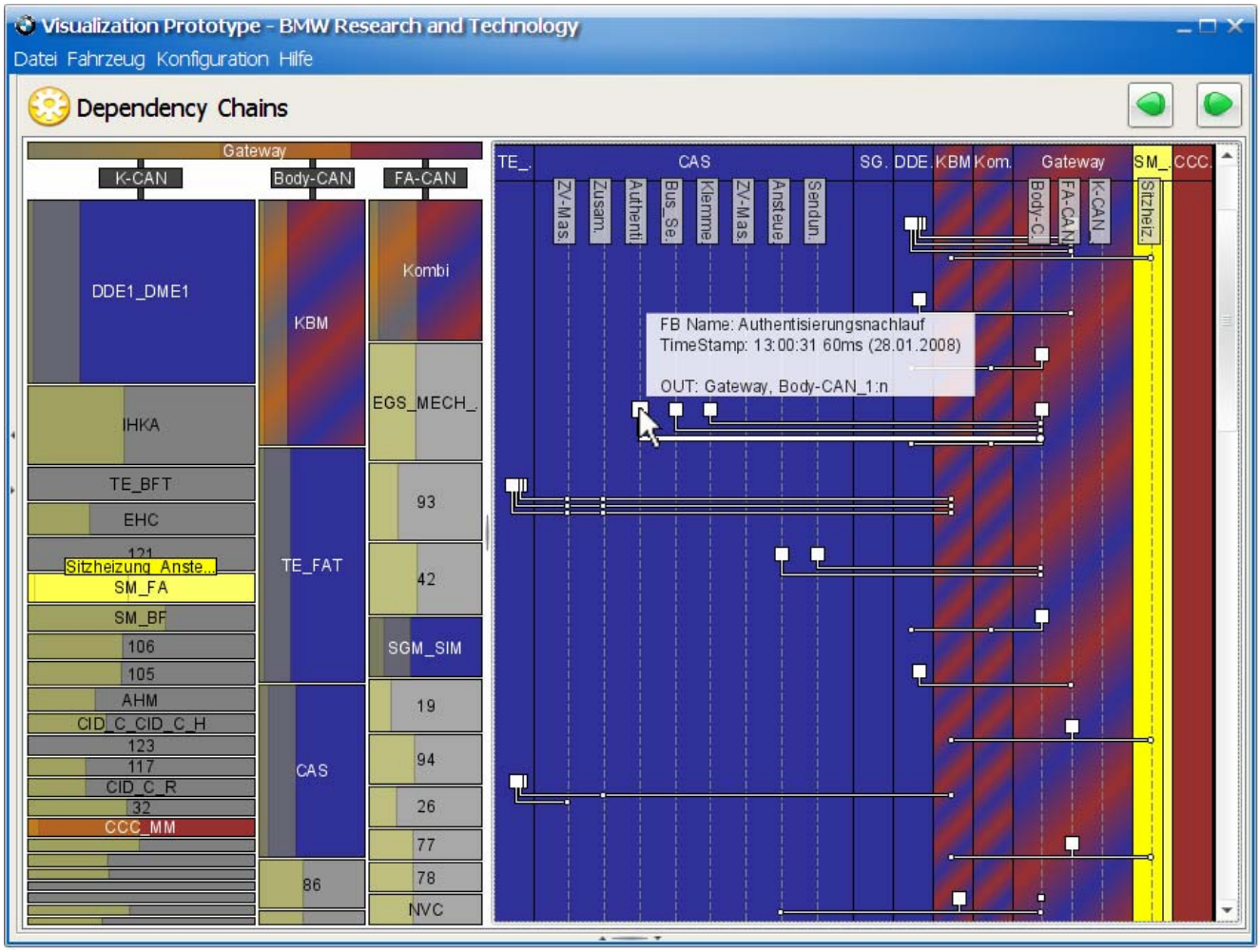


Figure 3: A screenshot of the entire visualization showing the ECU-Map view on the left and the Sequence Chart view on the right. Within the Sequence Chart view, one connection is highlighted via mouse-over, and additional detail is shown in a semitransparent tooltip.

bifocal display can present an increased number of documents on a single screen. Well-known variations of this approach include the perspective wall [8], and the document lens [11].

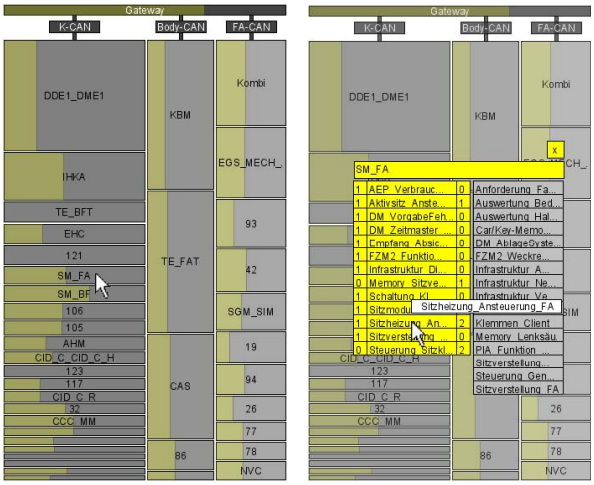
Sequence diagrams such as the Message Sequence Chart [6] or the Sequence Diagram as defined by the Unified Modeling Language [15] show ordered communication processes among objects. Objects are represented as parallel lifelines, while communication messages are displayed as arrows connecting the sender and receiver objects along the vertical time dimension. Other sequence visualization techniques such as Arc Diagrams [16] may be easier to understand for novice users, but are hardly suitable for conveying the temporal relations of object communication. Instead, most sequence visualizations focus on highlighting patterns in the data.

4. Visualization Design and Implementation

Our goal was to create an interactive visualization to support automotive developers in the process of analyzing and diagnosing dependency chains within erroneous in-car communication processes. A major challenge addressed by several measures was the scalability to real world data sets. A general guideline of our visualization approach was to use common automotive representation forms and enrich them with interactivity and information visualization techniques. In bridging traditional workflows with new technologies we expected to address the users' existing mental models and thereby achieve a better acceptance and understanding. We therefore designed a vertically separated dual-view approach, reflecting in the left view the entire physical layer of the in-car communication network and in the right view a dynamically adapted and situation specific extract of the functional network. By selecting a FB – the Trigger-FB – the user initiates a color-coded presentation of the views and can interactively explore on the physical as well as the functional layer the dependency chain around this Trigger-FB (schematically presented in figure 2).

4.1. The ECU-Map view

The left view is visualized as a mixture of an automotive ECU topology diagram and a treemap [7, 12] and is called the *ECU-Map*. The underlying hierarchy of the



(a) (b)

Figure 4: Basic display of the ECU-Map (a) and selection of an initial Trigger-FB in a popup (b).

treemap reflects the correlations between the entire network on top, the specific bus systems in the columns and the ECUs on the leaf layer (see figure 1, showing the underlying hierarchy of the in-car communication architecture). Due to its special interconnecting role, the gateway "ECU" is allocated a spanning position on top of the ECUs and is connected to each "bus-column" via a labeled link. The labels of the links show the names of the different bus systems the subjacent ECUs are connected to. To further support the visual allocation of the ECUs to their respective bus system, we used different shades of gray as base colors and a wider horizontal space separation. To discriminate the particular ECUs their representing rectangles are labeled with the ECU's name. The ECU-Map is created depending on the number of FBs in the "maximum car". The widths of the columns therefore represent the number of FBs per bus compared to the overall number of FBs. The heights of the individual ECU rectangles correlates to the number of FBs per ECU compared to the FBs per bus system. Hence the area of an ECU rectangle codes the percentage of its "maximum car"-FBs to the overall number of "maximum car"-FBs. This kind of coding allows the observer to directly get an impression of the power of an ECU and the potential impact of errors in this ECU on the overall system. Additionally the ECU rectangles obtain a "status bar" to indicate the number of active FBs within an ECU – these are the FBs used by one particular customer car variant. The status bar, which is represented as a semitransparent yellow overlay on the gray ECU rectangles resulting in an olive greenish sector of this ECU, is fully included in the treemap approach so that the area directly correlates with the entire system (Figure 4a). We had two major reasons for choosing the treemap approach to visualize the physical network. On the one hand it allowed us the special layout related to an ECU topology diagram. On the other hand the primary focus is set to the leaf nodes, the ECUs, which is very useful in diagnostics.

To start exploring, the user has to select an initial Trigger-FB within the ECU-Map. For this purpose, she or he can click on the ECU rectangles, which in turn display an alphabetical list of the included FBs. The list is split into two sections. A yellow section shows the active FBs and a gray section shows the inactive FBs in order not to lose the context information of the entire ECU functionality. Only the active FBs can then be selected to trigger a dependency chain. The active FB entries additionally contain the number of direct predecessor FBs (left number) and direct successor FBs (right number) to provide an initial indication of their dependency situation (Figure 4b). After selecting a Trigger-FB, both views are updated and the ECU-Map now highlights all ECUs involved in the triggered dependency chain by changing their color. The triggered ECU turns a light yellow and contains a header in a more saturated yellow to label the Trigger-FB. The predecessor ECUs turn blue and the successor ECUs red. If an ECU contains both predecessor FBs and successor FBs, it is drawn in blue and red stripes. In addition to the status bar, all ECUs involved obtain another yellowish overlay. This overlay represents the percentage of FBs involved in the triggered dependency chain, which obviously is a subset of the active FBs of the respective ECU. The overlay is also fully included in the treemap approach and can hence be correlated to the overall system (Figure 3, left view).

4.2. The Sequence Chart view

The *Sequence Chart* view is first shown after an initial trigger in the ECU-Map view and presents the respective functional dependency chain of the Trigger-FB. In order to save space, only the ECUs involved in the current chain are visualized. The Sequence Chart is constructed in the following way: The ECUs are allocated to horizontally arranged rectangles in the same yellow/blue/red color coding as in the ECU-Map. Within these ECU rectangles, vertical dashed lines represent the FBs involved within the respective ECU. The horizontal distance between these FB lines is constant, hence the width of each ECU rectangle depends on the number of FBs involved. Assuming a time line from the top to the bottom, the single communication connections between the FBs are presented. Each communication connection consists of three graphical elements. A small rectangle indicating a sending action is attached to the corresponding FB-Line. A communication line shows the path of the message sent. To distinguish irrelevant line-crossings and real sink FBs, small circles are added to mark the "docking point" on sink FB-Lines. Logically, this means that the information sent is now available at a sink FB. The resulting visualization is a color-coded sequence chart with a representation of the triggered dependency chain ordered by causality (Figure 3, right view).

4.3. Interaction

All interaction takes place through simple mouse operations. The most important user interaction is to find and to select the correct Trigger-FB. The initial Trigger-FB is selected in a two step interaction within the ECU-

Map (see section 4.1). Furthermore, each FB element in the Sequence Chart can be selected and thereby initiate the presentation of a new dependency chain. This is very important, because in many cases not only one FB is erroneous and diagnostic experts have to consider several different dependency chains. To provide an easy undo functionality, we integrated a browser-like history, which allows the user to navigate through the dependency chains explored so far. More flexibility is provided by integrating both views in a split pane with a movable boundary. The user can thereby dynamically scale the views according to her or his actual focus requirements. More detail, e.g., the full version of abbreviated component names, is provided in several places upon mouse-over. Also, the communication connections highlight the respective communication path upon mouse-over in order to provide a better orientation in highly complex dependency chains. More detail is added in a semi-transparent text box, providing information about outgoing communications, which are not part of the actual triggered dependency chain (Figure 3, left view at the mouse cursor).

In addition to the basic interaction of selecting an FB-Trigger, detailed information via mouse-over, scaling and history navigation, we added a *semantic fisheye zoom* to the Sequence Chart view. This solved the problem that some dependency chains are extremely complex and far too large for the horizontal space of the Sequence Chart. This complexity is controlled by a vertically acting semantic fisheye zoom, which is activated if the triggered dependency chain exceeds a certain length. All ECU rectangles in the Sequence chart except the ECU, which contains the FB-Trigger, are semantically downscaled. In downscale mode connections between the FBs are reduced to links between the downscaled ECUs instead of the incorporated FBs. To explore the downscaled ECUs on the basis of their FBs, the user can interactively apply a semantic fisheye zoom and expand the downscaled ECU back to the entire functional view (Figure 5). The interaction is also available in the other direction to collapse ECUs represented in detail.

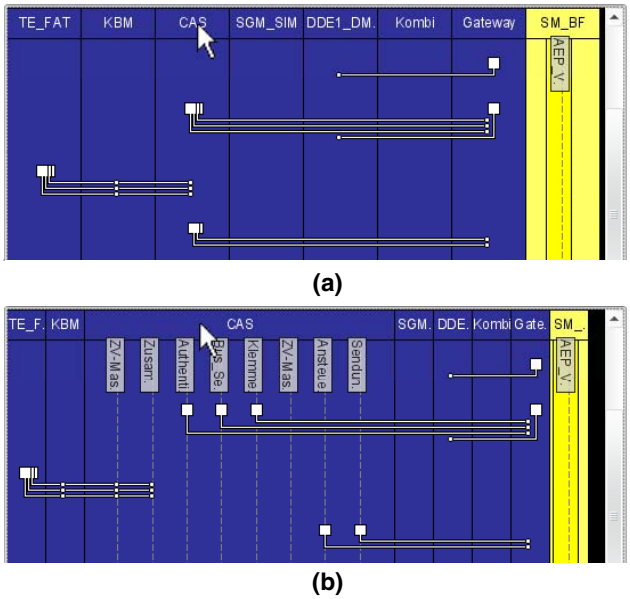


Figure 5: The Sequence Chart view before (a) and after a semantic fisheye zoom interaction (b).

4.4. Implementation

The visualization prototype is written in Java using the Piccolo framework to build a structured graphical application. It is integrated into a proprietary tool framework for the evaluation of diagnostic modules. The application is based on the Model-View-Controller paradigm, which allowed a clear and coherent coordination strategy for the dual-view visualization.

5. Evaluation

To test the understandability and the usability of our visualization approach, we conducted a qualitative expert user evaluation with six participants. It was organized as a think aloud protocol and took one hour for each user. Our prototype visualized an example data set of a customer car variant, containing information on three buses. It was presented on a 1920x1200 pixel screen. We started each session by giving the participants a short introduction of ten minutes. In the following 30 minutes they had to solve different tasks by interacting with the visualization. At the end of the study, the participants were asked a number of questions regarding the information encoding, interaction design and the dual-view concept and were encouraged to express feedback and criticism. Our participants were three automotive diagnostic experts and three students with a usability background. So we actually had two different “sub-studies”, which allowed us on the one hand to receive meaningful results for the target group of diagnostic specialists, i.e., the domain experts. On the other hand it also provided us with an independent group of usability experts to evaluate the usability and compare to the diagnostic expert group. In addition to the basic test, the diagnostic experts were given additional diagnostic tasks, as well as specialized questions regarding their current workflows in comparison to the application of the new visualization tool.

The overall feedback to the visualization was positive. The users were able to solve most tasks without further help by the instructor. The encoding of the information was judged to be easily understandable and the usability of the interaction design was ranked highly. The dual-view concept was also judged positively. All users liked the simultaneous views and used them in an overview-and-detail as well as in a navigation-and-exploration manner. They all felt that a multiple view approach was appropriate and helpful for the visualization of dependency chains. Although the student group had minor problems with context switching between the views, they all preferred a simultaneous presentation to a sequential one. What subjects liked most about the multiple views, was the permanent presence of the physical in-car communication overview in the ECU-Map. This provided a good understanding of the situation in the

overall system context at every point of the visualization. The Sequence Chart, on the other hand, was the region mainly used for exploring and interacting. The users closely examined dependency chains of different complexity and used all the interaction alternatives to explore them. The mouse-over was used to highlight the passes and to easily find appropriate linkages. It was ranked by the users as an important and usable feature. The semantic fisheye zoom was also very well accepted and was frequently used by each user. The "opening and closing of the ECUs is enormously helpful and understandable" said one of the users while handling the tasks. The participants, especially the diagnostic expert group used the semantic fisheye zoom to dynamically expand two ECUs and explore their dependencies.

The encoding and its meaning were understood very well in general. After hearing it once the users did not have problems with the color coding of successor and predecessor elements. The treemap size coding as well as the integrated status bar (yellow overlay) was appreciated. Both was used to get an impression of the importance of each ECU and – in the case of the status bar – of the activity of the ECU in a particular customer car variant. Only the second overlay onto the status bar caused some confusion in the students group based on the complex underlying data and structures. The diagnostic expert group did not use this feature very often either, but they valued it as "useful additional information with little optical noise".

Another very interesting point of the study was the appreciation of our tool by the diagnostic expert group and their comparison to current workflows. Above all, the three participants underlined its potential for saving time. One user formulated it as follows: "The task I now solved in a few minutes, would have taken at least 15-20 minutes with the traditional methods".

6. Conclusions

We have presented the design of an interactive multiple view visualization for in-car communication processes and described the measures we took to make it scaleable to real world data sets and understandable to its target group. In an expert evaluation, we received very positive feedback about our design and our subjects particularly emphasized the potential of this approach for saving time in diagnostic processes. These results encourage us to further investigate the application of information visualization approaches to this kind of data sets and to present our work so far as a best practice example for the automotive sector.

References

[1] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for using multiple views in information visualization. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 110–119, New York, NY, USA, 2000. ACM.

[2] Richard A. Becker and William S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[3] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[4] Robert M. Edsall, Alan M. MacEachren, and Linda Pickle. Case study: Design and assessment of an enhanced geographic information system for exploration of multivariate health statistics. In *INFOVIS '01: Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, page 159, Washington, DC, USA, 2001. IEEE Computer Society.

[5] G. W. Furnas. Generalized fisheye views. In *CHI '86: Conference on Human Factors in Computing Systems*, pages 16–23, 1986. ACM.

[6] Ekkart Rudolph Jens Grabowski, Peter Graubmann. The standardization of message sequence charts. In *Proceedings of the Software Engineering Standards Symposium 1993*. IEEE Computer Society, 1993.

[7] Brian Johnson and Ben Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 284–291, IEEE Computer Society Press, 1991.

[8] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: detail and context smoothly integrated. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 173–176, New York, NY, USA, 1991. ACM.

[9] Galileo Mark Namata, Brian Staats, Lise Getoor, and Ben Shneiderman. A dual-view approach to interactive network visualization. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 939–942, New York, NY, USA, 2007. ACM.

[10] Chris North and Ben Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 128–135, New York, NY, USA, 2000. ACM.

[11] George G. Robertson and Jock D. Mackinlay. The Document Lens. In *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 101–108, 1993, ACM.

[12] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1): 92–99, 1992.

[13] Ben Shneiderman and Martin Wattenberg. Ordered treemap layouts. In *INFOVIS '01: Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, page 73, Washington, DC, USA, 2001. IEEE Computer Society.

[14] Robert Spence and Mark Apperley. Data base navigation: an office environment for the professional. Behaviour and Information Technology, 1(1): 43-54, 1999.

[15] UML url: http://www.uml.org.

[16] Martin Wattenberg. Arc diagrams: Visualizing structure in strings. In *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, page 110, Washington, DC, USA, 2002, IEEE.

[17] Jarke J. Van Wijk and Huub van de Wetering. Cushion treemaps: Visualization of hierarchical information. In *INFOVIS '99: Proceedings of the 1999 IEEE Symposium on Information Visualization*, pages 73-78, Washington, DC, USA, 1999. IEEE Computer Society.