# RelEx: Visualization for Actively Changing Overlay Network Specifications

Michael Sedlmair, *Member, IEEE*, Annika Frank, Tamara Munzner, *Member, IEEE*, and Andreas Butz

**Abstract**—We present a network visualization design study focused on supporting automotive engineers who need to specify and optimize traffic patterns for in-car communication networks. The task and data abstractions that we derived support actively making changes to an overlay network, where logical communication specifications must be mapped to an underlying physical network. These abstractions are very different from the dominant use case in visual network analysis, namely identifying clusters and central nodes, that stems from the domain of social network analysis. Our visualization tool RelEx was created and iteratively refined through a full user-centered design process that included a full problem characterization phase before tool design began, paper prototyping, iterative refinement in close collaboration with expert users for formative evaluation, deployment in the field with real analysts using their own data, usability testing with non-expert users, and summative evaluation at the end of the deployment. In the summative post-deployment study, which entailed domain experts using the tool over several weeks in their daily practice, we documented many examples where the use of RelEx simplified or sped up their work compared to previous practices.

**Index Terms**—Network visualization, change management, traffic routing, traffic optimization, automotive, design study.

## 1 Introduction

This design study is the result of a 9-month collaboration with German automotive engineers at BMW tasked with specifying and optimizing the communication within *in-car networks*, which connect the vehicle's electronic control units (ECUs) and enable the exchange of information between them via several communication bus subsystems. Their work is part of the challenging process of developing and testing a network for a new series of cars [4] and at any given time there are up to five such networks in active development at BMW. The difficulty of this data-intensive process and the need for better tools is widely acknowledged [7, 14, 18, 37]. The limited previous work on visual network analysis in this domain has focused on analyzing network traces in a late phase of the development cycle [40]. Here we present *RelEx*, short for Relation Explorer, a system that supports development engineers in an earlier phase of the automotive design cycle. RelEx focuses on the mapping of logical communication specifications to a physical network of ECUs and bus systems. We designed RelEx to better support our target users in adapting in-car networks according to change requests from other engineers and in optimizing network traffic.

Analyzing network data is a large and important application area for visualization, and a multitude of approaches and techniques for network visualization have been proposed [46]. Despite some initial work towards understanding low-level, generic tasks in visual network analysis [13, 25], the visualization community's understanding of what people are doing in real-world settings remains incomplete. Most of the previous design studies investigating real-world applications of network visualization have been conducted on the domain of social network analysis. There the central problem for data analysts is to identify and describe communities and actors [17, 20, 34, 45]. Abstractly, these domain problems map to the generic tasks of identifying and describing clusters and specific types of nodes including high-degree nodes, bridge nodes, and outliers. These generic tasks also occur in the visual analysis of email communication networks [24] and academic co-authorship [30].

- *M. Sedlmair and T. Munzner are with the University of British Columbia, Vancouver, e-mail: {msedl, tmm}@cs.ubc.ca.*
- *A. Frank is with the Bertrand AG, Munich, e-mail: annika.frank@partner.bmw.de.*
- *A. Butz is with the University of Munich (LMU), e-mail: butz@ifi.lmu.de.*

*Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.*

However, network analysis problems arise in a very broad set of application domains, and in many cases these problems map to very different abstract tasks. In particular, we focus on building engineered networks [1, 49], rather than on discovering evolved networks. This design study adds diversity to the set of carefully documented use cases that arise from the real-world practices and problems in visual network analysis. Such diversity will help the visualization research community avoid the pitfall of focusing too narrowly on use cases and tasks that do not cover the full scope of user needs. An example of such a pitfall in another field is discussed by Borgatti in his typology of network flows [3], where he argues that the implicit assumptions underlying many off-the-shelf network centrality measures only hold for a small and specific set of network analysis use cases and are not appropriate for others.

RelEx was developed through a collaborative user-centered process that began with a problem characterization phase. We conducted a grounded investigation of the current practices, needs, and challenges of our target group, including an analysis of the success and failure factors of previously proposed tools. We then developed abstractions of the data and tasks and derived design requirements for visual analysis tools. We chose a coordinated multiple view approach featuring a custom matrix representation for an overview of communication patterns and a filtered detail view to support path tracing, in addition to traditional domain-specific topological network views and lists. We evaluated RelEx with domain experts in a longitudinal field study over 5 weeks with 7 domain experts, and a qualitative lab study with 10 domain experts. We also conducted a heuristic evaluation with HCI graduate students for further usability refinement. The resulting tool for the visual exploration of complex relations sped up some tasks that were difficult with the previous workflow of our target users, and allowed them to conduct some tasks that were completely unsupported before. After the collaborative development and evaluation phase was completed, our domain experts continued using RelEx for their daily activities.

Following recent guidelines on design study research [41], this paper makes two contributions. The first is the characterization and abstraction of tasks and data in this problem domain, and their relationship to abstractions used in other network visualization application areas. Our abstractions add a new and very different perspective on the problems and tasks that people face in visual network analysis. The second contribution is the carefully justified and validated design of RelEx, created with a full user-centered design process that included close collaboration with expert users at every step of the way. The validation included a post-deployment study showing the effectiveness of the matrix view, the first study documenting their utility in the field.

## 2 Related Work

Most network visualization design studies focus on the abstract tasks of cluster analysis and the identification of central nodes and outliers. Many of these tasks come from the application domain of social network analysis, where the domain problem is framed as the identification and description of communities, key actors, and atypical actor roles. Henry and Fekete documented that this abstraction matches the needs of social science researchers as part of the MatrixExplorer design process [20]. The Honeycomb system from van Ham et al. [45] and the work of Perer et al. [35, 34] also fit this mold. Design studies focused on email communication networks [24] and academic paper co-authorship [30] also use similar abstractions.

The abstract tasks and data abstraction of our target users, network engineers, are fundamentally different. Our data abstraction is built around multiple related networks that are explicitly changed by the user, with a logical specification of communication paths layered on top of a physically instantiated bearer network, and the abstract tasks arise from the higher-level goals of specifying and optimizing traffic flow, as described below.

Some design studies in network visualization have indeed addressed a broader set of abstract tasks. In computational linguistics, Munzner et al. proposed the Constellation system for visualizing semantic correlation between words, with a focus on path tracing tasks [29]. Van Ham uses a matrix-based visualization to support software architects in identifying outliers – unwanted calls between software subsystems – and the communication across different subsystems [44]. Another major domain for network visualization stems from biological data. Cerebral, for instance, was designed to support system biologists in understanding changes over time and clusters in interaction graphs of biomolecules [2]. ABySS-Explorer offers a graph-based visualization of a sequence assembly with the goal of disambiguating paths [31].

Closest to our work are design studies from system analysis and network trace visualizations. Pretorius and van Wijk presented a multivariate graph visualization tool for state transition analysis. They explicitly outline that while most visualization techniques focus on showing the structure of a graph, state transition analysis necessitates incorporating additional information that is associated with nodes and edges [36]. Our work presents another use case along these lines, where the user needs to go beyond showing and understanding the topology of a network. Finally, visual network analysis has also been proposed for monitoring and testing computer networks. Fischer et al. discuss a design study about visualizing network communication to detect and analyze attacks on large computer networks [12]. Sedlmair et al. introduced Cardiogram, a visual network analysis approach for debugging in-car communication networks [40]. Both of these deal with traces of actual communication in computer networks, whereas we instead focus on the specification and optimization of communication routing in such networks.

## 3 Process and Methods

This design study was conducted in nine months. Our user-centered design approach was motivated by guidelines advocated in a wide variety of methodologies, ranging from Action Research [15] to Multidimensional In-depth Long-term Case Studies (MILCs) [42]. The overall process was organized in three stages – problem characterization, tool design and formative evaluation, and summative evaluation – each of them using a mix of different data gathering methods to allow for data triangulation. The target users were engineers within the large automotive company BMW, working under high time pressure. All activities involving domain expert participation were therefore carefully chosen and planned, considering their time as a restricted resource. The two researchers actively involved in planning and conducting these studies were embedded within the company, with their daily workspace close to those of the users.

**Problem Characterization:** The goal of the problem characterization phase was to get a good understanding of our target group's needs and challenges. We did so by:

1. Reading about target domain background, from both internal company documents and general background literature [4, 5, 26].
2. Understanding their tasks, previous practices, and problems by conducting field work.
3. Identifying failure and success factors for tool use in the domain.
4. Abstracting the data and tasks from domain-specific to generic forms that can be addressed through visualization methods.
5. Deriving design requirements for RelEx, and considering how these might generalize to other problem contexts.

A major method used in our field work was contextual inquiry, namely the combination of in-place user observation with the discussion of what is observed [22]; we also used semi-structured interviews and a focus group. We undertook contextual inquires with five domain experts, all working in the area of network architecture, and conducted between two and ten sessions per participant where each session took between five minutes and one hour. During the sessions we asked our participants to demonstrate typical and important tasks, and we interrupted to request clarification of activities that were not obvious from silent observation. We also asked questions regarding their field of activity, other tools they use for their daily activities, and challenges they are confronted with. Exploiting our workspace proximity, we also requested that they spontaneously call us over to observe them when their daily work led to relevant tasks that might interest us. These opportunistic interviews were a rich source of information about their true daily work practices, versus the idealized recollected set of tasks discussed at the pre-scheduled interviews. We also conducted a focus group study with four domain experts. Finally, we carefully selected three lead users from all of these engineers, and conducted guided interviews with them in order to focus and evaluate our findings. In all studies we took notes to log information and statements. This phase lasted roughly three months. Section 4 presents its results.

**Design and Formative Evaluation:** Based on the implications from our problem characterization, we designed and implemented a prototype called RelEx, short for Relation Explorer. During this phase, we continued the collaboration with the three lead users that we identified before. Our process was divided into four stages, following the design principles of parallel prototyping [10]. First, we engaged in a diverging phase considering a broad set of alternative data abstraction and visual design ideas. Appreciating the time restrictions of our collaborators, we ruled out many ideas by evaluating them against our problem analysis without further user contact. Second, we created paper mockups of the set of pre-selected ideas and discussed them with our lead users to obtain feedback. Third, we converged to one final design concept that we again paper prototyped and discussed with the lead users. Fourth, we implemented the refined concept as a software system using an agile development process with six major tool releases over a period of three months. These prototypes were actively used by the three lead users for their daily work, and we constantly elicited feedback from them and refined the tool accordingly. In parallel, we conducted a heuristic evaluation with five HCI graduate students to detect generic usability issues that could be found by non-experts and further refined the tool to fix these problems as well. Sections 5 and 6 report the results of this phase.

**Summative Evaluation:** To validate RelEx's domain usefulness, we conducted two studies. First, we conducted a field study in which we deployed RelEx to a group of 7 automotive network architects who used the tool for a period of 5 weeks. Our initial plan was to have 5 pre-scheduled hour-long weekly meetings. In practice, we met opportunistically at the instigation of the participants, in a similar spirit to the problem characterization phase. These spontaneous meetings lasted between ten minutes and one hour, and ranged in number from 3 to 9 with each participant. In these meetings, we interviewed the participants about their experiences with the tool and elicited both formative suggestions for improvement and summative feedback in terms of usage examples. While we also gave them logbooks, following standard recommendations [42], we found that participants almost never used them due to their high workload. Based on the formative feedback, we iteratively refined, adapted, or added additional features to

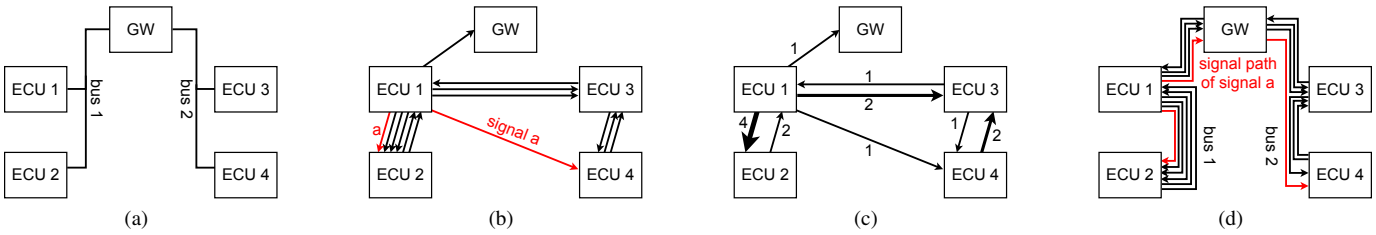



Fig. 1. Network abstractions: **(a)** The *physical* network describes how ECUs and bus systems are connected; **(b)** the *logical* network depicts signals and their sending and receiving ECUs; **(c)** the *logicalcount* network collapses the logical multigraph into a weighted simple graph, with weights corresponding to the number of signals sent between a pair of ECUs; **(d)** the *signalpath* network results from mapping the logical onto the physical network, showing signal paths.

RelEx, and provided our participants with tool updates. The summative results included experts' estimations, comments and opinions, as well as many usage examples where our participants successfully used RelEx for their daily tasks. Second, we conducted a think-aloud study with 10 domain experts, 5 of whom also participated in the field study. These sessions lasted roughly one hour. This study focused on the qualitative comparison of RelEx with the four previous tools discussed in Section 4.3. We focused on gathering their opinions of benefits and drawbacks and deliberately abstained from gathering quantitative performance measurements. We did gather quantitative subjective ratings using a questionnaire about tool use for a set of four representative and open tasks, where the suitability of the tools for each task was rated using 5-point Likert scales. In both studies, we logged verbal feedback with written notes. The summative evaluation results are discussed in Section 6.

**CHALLENGES:** During our investigations, we heard many statements that we consider misconceptions about the role and value of visualization, including *"visualization is 3d"*, *"visualization is doing graphics in Powerpoint"*, *"visualization is human-car interface design"*, *"visualization is UI design"* and *"visualization is painting nice pictures"*. We chose to combat these misconceptions both explicitly through dialog, and implicitly through creating an effective visualization tool.

## 4 PROBLEM CHARACTERIZATION

We discuss data description and abstraction, results from our task analysis, previous practices and problems including successful and failed domain tools, and design requirements we derived.

### 4.1 Data Description and Abstraction

The data our collaborators worked with was mainly stored in a database called NetDB, a company-wide resource for all information related to in-car network development. A first challenge was to filter the full information available within the database into the subset that was important to our target users, and then to transform it into an understandable data abstraction that is useful for visualization. Our contribution includes the crisp abstract description of existing network perspectives of engineers, and the explicit identification of new elements and abstractions as described below. We therefore distinguish between the *basic perspective*, which more or less reflects and abstracts how engineers currently think about in-car networks [4], and *further aggregations* which we and our domain collaborators introduced on top of this understanding.

**BASIC PERSPECTIVE:** Three types of elements found within in-car networks were the main focus of attention for our users:

- The roughly 100 Electronic Control Units, or **ECUs**, are the nodes of the network. There are three node roles: **send**, emit a signal; **recv**, receive a signal; and **gateway**, repack and forward a specific signal from one bus system to another.
- The roughly 10–15 bus systems, or **buses**, connect the ECUs. Abstractly, they are undirected hyperedges – meaning that they can connect multiple nodes – in the *physical* network discussed below. ECUs with gateway roles are connected to multiple buses.
- The roughly 10,000 signal specifications, or **signals**, are information containers used to transport a particular kind of information in the network. Examples include 'speed', 'temperature', and 'window state'. These signal specifications describe the types of information flow that can occur between ECUs, as opposed to the actual signal instances that might appear in a trace of an active network, such as 'speed = 25mph'. Abstractly, a *signal* is a set of directed edges between ECU nodes in the *logical* network defined below. Signals begin from a set of $m$ send ECUs and end at a set of $n$ recv ECUs, but the common case is a single sender with multiple receivers. The same physical ECU might have different roles – send, recv, or gateway – for different signal specifications.

Based on these element groups, engineers distinguish between two types of networks, which we abstract as following:

- The **physical** network specifies how ECUs are connected by buses; signals are not involved. It can be described by a set of $\{ECU, bus\}$ pairs resulting in an undirected hypergraph, as in Figure 1(a). This topological network is sparse, with a small number of ECU nodes and bus edges.
- The **logical** network specifies the communication of signals between ECUs; buses are not involved. It can be described by a set of $\{send, signal, recv\}$ triples resulting in a directed multigraph – meaning that there can be multiple edges between nodes – as shown in Figure 1(b). This network is very dense, typically with two orders of magnitude more signal edges than ECU nodes.

**FURTHER AGGREGATIONS:** We found that the needs and interests of our users were more complex than these basic elements supported by their previous tools, and thus derived further data abstractions and aggregations. We identified two more elements:

- **signalcount** elements, which are the result of aggregating together all signals between a particular pair of ECUs into a single element that includes a count of how many signals it includes;
- **signal paths** which specify the routing of a specific signal from the logical network over the physical network.

which led us to two new network abstractions:

- The **logicalcount** network is a weighted version of the logical network transformed from an unweighted multigraph to a weighted simple graph, as shown in Figure 1(c). That is, there is only a single edge between any pair of nodes, and the edge weights are the number of signals between those two ECUs in the logical network. Each edge in this network is a single signalcount element. While this network has up to an order of magnitude fewer edges than the logical one, it remains very dense.
- The **signalpath** network specifies the mapping of the logical network onto the physical network. In other words, the logically specified signals are mapped to signal paths that adhere to the physical network structure, taking into account the bus connectivity. It can be described as a set of $\{src, signal, bus, dest\}$ 4-tuples resulting in a directed multigraph, as in Figure 1(d). The *src* ECU can be either a *send* or a *gateway*, and the *dest* can be either a *gateway* or a *recv*. A single signal triple from the logical network is mapped to a set of multiple 4-tuples in the physical

network; this 4-tuple is the signal path defined above. This graph is even more dense than the logical network, since a single signal on it between two ECUs on different buses is mapped into several segments, one for each gateway that must be traversed.

**NOVEL ASPECTS:** This data abstraction differs from those in previous work in several ways.

In terms of scalability requirements, nearly all previous work has focused on the scalability of the topology itself, with systems designed to handle large numbers of nodes and edges. In our case, the physical network is quite small, with around 100 ECU nodes and 10-15 sets of bus hyperedges. Straightforward static representations are sufficient to show this topological structure alone, as supported by the Topology Maps discussed in Section 4.3. Instead, we face a **path scalability** problem: there are 10,000 signals, specifying communication paths that are overlaid onto this topology.

The network specification data is also **actively** changing due to explicit actions by the target users as described below. In contrast, the previous work on dynamic networks has been in support of passively changing datasets, where the users seek to understand what changes occurred; the Honeycomb system is one example [45].

Finally, this dataset can be described as an **overlay communication network** where a logical layer expressing communication paths is mapped onto a physical layer with physical edges. This set of related networks is a more complex data abstraction than the networks typically found in domains such as social network analysis, where there is a single directed graph with actors as nodes and friendship relations as edges [17].

### 4.2 Task Analysis

Our final task abstraction begins with a summary of the problem of traffic optimization, as articulated by the automotive engineers. We broke this high-level task down into a set of mid-level subtasks, and developed a low-level characterization of all user activities in terms of queries about relations.

The central concern of the engineers is to actively change the network. A change request, particularly one from external colleagues, will directly address only the elements of a **basic** physical or logical network specification. These constitute adding, removing, or changing ECUs, buses or signals. However, changes to individual elements in these two networks often require additional changes to the **signalpath routing specification**: the set of signal paths that specify the mapping between them. Examples include connecting an ECU to an additional or different bus, sending a signal via another bus, or sending a signal to other ECUs. Thus, the signalpath network is usually affected by a change request, even though it is not directly mentioned.

**TRAFFIC OPTIMIZATION:** The high-level goal of traffic optimization on the signalpath network is thus the ultimate consideration that underlies all change requests. The process of specifying an efficient and effective signal routing requires accounting for different costs and constraints. Costs stem from traversing edges or nodes and from adding additional elements. The engineers use many cost metrics including available bandwidth, delay and real-time requirements, path length, ECU load, reliability, and money: adding new elements might increase the financial cost of vehicle production. The engineers must take into account the hard constraints that the physical bus edges and the ECU nodes have a capacity limit and cannot be overloaded, and soft constraints such as the goal of load balancing the physical network. Unbalanced communication can lead to communication problems and does not efficiently use the available resources.

The target users heavily relied on this implicit knowledge about costs and constraints during the process of proposing, implementing, and validating changes. If complete and full information about the cost structure of the problem were to become digitally available, then the tasks would be fully automatable, and there would be no need for a visualization tool. However, such knowledge elicitation would be tricky; the set of considerations at play is sufficiently complex that there is no current hope of fully automating the process any time soon.

**MID-LEVEL SUBTASKS:** We identified four specific subtasks that were carried out as part of traffic optimization:

- locating **communication hotspots**, heavily travelled edges on the logical and signalpath networks indicating high traffic.
- finding the number of communication partners for a specific ECU, by distinguishing between **local** communicators that only exchanged signals with a few other ECUs, versus **global** communicators that exchanged signals with nearly all ECUs in the network.
- characterizing the communication structures between entire bus systems on the signalpath network, by understanding whether a bus system is an **introvert** where most of the traffic is between elements directly attached to it with little or no communication with other network sub-systems, or an **extrovert** that forwards and receives communication from other buses.
- determining whether a signal is **well-directed** between just two ECUs, or **wide-spread** across many ECUs.

**LOW-LEVEL QUERIES AND RELATIONS:** We found a way to express the high- and mid-level tasks as low-level operations: they are all **queries** about **relations** between the elements and networks defined in Section 4.1. That is, the answer to a query is one or more relations. All of these queries can be expressed as the operation of filtering based on a selected element, as outlined in Table 1.

Example queries include: Which ECU is communicating with which ECU? Which signals do they exchange? Is a specific signal available on a bus system? What is the path a signal takes? How many signals does a ECU receive? What signals are sent over a bus system? What pairs of ECUs do exchange many, little, or no signals? What signals are exchanged across bus systems?

One factor in query complexity is the number of interacting elements in the relation: some questions can be answered as a two-way relation between elements, for example those about the physical network of ECUs and buses. Others require three-way relationships, for example the triples of the logical network, and four-way relationships, for example the 4-tuples of the signalpath network. A second factor is the number of relations returned: while a simple query might return only a single relation, more complex queries return multiple relations. The last factor is whether the query encompasses only relations **within** a set, such as a particular network, or **between** two sets, such as mappings from one network to another.

**NOVEL ASPECTS:** These tasks are for the most part very different from those found in previous network analysis design studies. Many previously identified tasks are not relevant for this domain, for example the cluster identification task common in social network analysis [17, 20, 21, 45]. The hotspot detection task is interestingly different from finding key actors in social network analysis [20, 21, 45], because the focus is on edges rather than nodes: that is, finding high-traffic edges rather than finding highly central nodes.

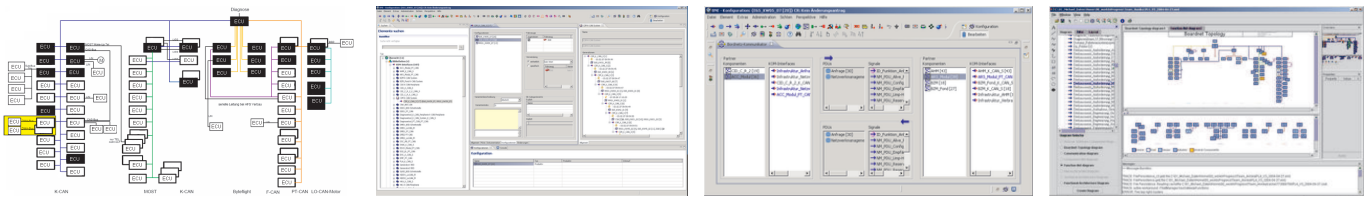
### 4.3 Previous Tool Analysis

At the time the design study began, the engineers had access to a set of four previously created tools that supported visual data exploration. All were proprietary tools either developed in-house or made by third parties according to specifications developed in-house. We found that two were frequently used in daily practice, one was occasionally used, and one had completely fallen out of use.[1]

**TopoMaps**, as shown in Figure 2(a), are manually created static node-link representations of the physical network that show all ECU and bus systems. They were used to obtain an overview of the entire physical network and to understand the wiring of ECUs to bus systems and reachability between ECUs. TopoMaps were used ubiquitously and provided a strong mental model: the engineers thought, discussed, and sketched based on this representation.

**NetDB**, the large, company-wide available database for all data related to in-car networks, comes with a text-based interface, shown in Figure 2(b). This interface was heavily used for exploring what we

[1]The tool names have been sanitized to remove proprietary information.



(a) (b) (c) (d)

Fig. 2. Miniature screenshots of previous tools: **(a)** *Topology map*; **(b)** *NetDB* frontend allowing for simple queries; **(c)** *Signaller* showing signals between two specified ECUs; **(d)** The former visualization tool *AutoTopo* for automatic generation of static visualizations such as topology maps.

call 'simple connections': a query for a single element with the results filtered to show all of its connections to elements of another type. Example usage included: all signals a specific ECU sends, all ECUs connected to a specific bus system, and which signals are transported over a specific bus system. At the time of our collaboration, the NetDB interface did not allow for direct queries of signal paths; these had to be pieced together from multiple queries.

**Signaller** is a plugin for the NetDB interface that some engineers used occasionally, as shown in Figure 2(c). It allows the user to query for all signals that flow between two selected ECUs in the logical network. The user can browse a list of send ECUs on the left and recv ECUs on the right. Selecting a specific sender-receiver pair then shows all signals that pass between them in the center of the view.

**AutoTopo** is a previously developed tool for in-car network visualization, shown in Figure 2(d). It showed a semi-automatically created static view with the same information as the manually created TopoMaps described above. The tool was no longer in use, and we were curious as to why. The proximate cause was a change in NetDB data export formats, but when we inquired further the ultimate cause was that AutoTopo had only been rarely used and thus was not missed in daily working practice after the format change. Although engineers first volunteered positive feedback about AutoTopo, especially about the automatic topology map creation, further elicitation yielded four reasons for its abandonment. First, the semi-automatic layout approach yielded only minor benefits over manual refinement of the drawings because most diagram changes were minor. Second, there were many usability issues, most importantly that the manual layout refinement was lost when data was reloaded. Third, the requirement to export data from NetDB before using AutoTopo imposed a significant time cost before the tool could be used. Fourth, the tool did not provide any novel data exploration capabilities, supporting only the same usage as the static maps.

The top half of Table 1 summarizes the capabilities of these tools using the data and task abstractions proposed above.

**Needs and Challenges:** Understanding connections between the physical, logical and signalpath networks was the most critical need. However, most of the engineers felt restricted in exploring these connections with previous tools. The main limitations stated by the participants were the restriction to explore only simple connections between the network elements, the lack of overview for signals, and the absence of suitable representation techniques for signal paths. One participant said: *"A good solution for better showing signal communication is most urgent"*.[2] Another engineer mentioned that *"technically, you can get all information [...] but if you need to understand more than 1:n connections, you have to manually add up all information you get from [NetDB]"*. He referred to challenges in understanding more complex connections such as signal paths. In such cases, engineers had to sequentially conduct multiple queries while either forming a mental picture of what was going on or using paper and pen to sketch interactions and connections.

Both TopoMaps and NetDB were invaluable and necessary tools for our participants, but were not sufficient for their data analysis needs. Their intrinsic goals were often at a high level, such as *"which other elements will be influenced when I change this function"* or *"what are the dependencies of this ECU within the entire network"*. These goals had to be translated by the engineers to the set of simple queries supported by these views, resulting in information snippets that needed to be manually combined.

**Visual Data Exploration Needs:** The high-level tasks presented above require major components of manual analysis and decision-making by experts. Our target users were frequently engaged in exploratory data analysis of the logical, the physical, and the signal-path network, particularly in the initial phases of inspecting the current network configuration as part of creating or evaluating a change proposal. They then used a combination of automatic verification and manual inspection methods to validate the proposed changes before releasing a new version of the network specifications.

### 4.4 Design Requirements

Our problem characterization phase resulted in six top-level design requirements. The first three are very general, whereas the last three are specific to this problem.

- Integrate with existing tools, to avoid any time costs of data export and formatting. Our analysis of the AutoTopo failure factors showed that even small technical hurdles on this front could result in the neglect of a tool. Close integration also allows for a more realistic evaluation in a large company environment [39].
- Extend rather than displace current practices. We carefully identified the strengths of current practice: the strong mental model of the predominant topology map representation, and the capability for simple queries. Our goal was to preserve, augment, and extend them with further data exploration capabilities.
- Interaction as a first-class citizen: the tool should support some aspects of the tasks through its visual encoding, and others through interaction. While the value of interactive data exploration to amplify cognition is not a new idea for the visualization community [8, 9, 47], the reliance of the failed AutoTopo project on static images showed that the potential of interactivity for visual data analysis was not apparent to this audience.
- Rich exploration of complex relations, not just simple ones, needs to be supported. The fundamental problem with previous tools is that the investigation of 2-way relations were well supported, but there was only limited support for 3-way relations and no support at all for 4-way relations. This limitation led to two gulfs: a gulf of execution where users must mentally break down a complex query into the simpler ones supported by the tools, and a gulf of evaluation from the need to combine the multiple query results into a big picture using manual or internal resources [32]. Our tool should support complex, multi-way relations, as suggested by the name *Relation Explorer*.
- Signal paths need an explicit visual representation. Previous tools force the users to manually construct these through multiple simple queries shown with textual output.
- A logical overview is needed, showing signals and their connection to other elements, just as TopoMaps provide an overview over the physical network. We derived this requirement from the desires of our participants for an *"overview over signals"* but

[2] All quotations translated from German.

Table 1. Visual encoding comparison between previous tools and RelEx. In the Scope column, normal text indicates selected items, italics indicates items filtered according to the selections, and bold indicates that all items in the set are shown.

| PREVIOUS TOOLS | | | |
|---|---|---|---|
| Name | Network | Scope | Description |
| **TopoMap/AutoTopo** | Physical | **ECU**, **bus** | 2-way full (both overview and detail) |
| **NetDB** | any | { A, *B* } | 2-way detail: filtered by selection |
| (NetDB example) | Physical | { ECU, *bus* } | buses connected to selected ECU |
| (NetDB example) | Logical | { ECU, *signal* } | signals sent from selected ECU |
| (NetDB example) | SignalPath | { bus, *signal* } | signals sent over selected bus |
| **Signaller** | Logical | { ECU, *signal*, ECU } | 3-way detail: filter signal set by send/receive ECU selection |
| RELEX | | | |
| View | Network | Scope | Description |
| **Topology** | Physical | **ECU**, **bus** | 2-way full; same as TopoMap above |
| **List** | any | { A, *B* } | 2-way detail: filtered by selection; same as NetDB above |
| **Matrix** | LogicalCount | **ECU**, **signalcount**, **ECU** | 3-way overview |
| **SignalPath** | SignalPath | { *ECU*, signal, *bus*, *ECU* } | 4-way detail: filtered by selected signal |

also by considering the gaps in the classification of data exploration capabilities of previous tools, as summarized in the top half of Table 1.

## 5 RELEX

RelEx uses multiple coordinated views with linked selection and highlighting [33], as is standard practice [48]. Following the INTERACTION requirement, we carefully considered which aspects of the tasks should be supported by directly visually encoding information, versus through interaction via view coordination. RelEx was designed for use with large, high-resolution displays of 1920x1080 pixels.

There are four main views, as shown in Figure 3. The **Topology View** shows the physical network of all ECUs connected by buses with an automatically computed layout designed to be as similar as possible to the previous TopoMaps. Two **List Views** show text lists of all ECUs and signals respectively, featuring carefully designed coordination with other views to support both simple and complex queries. Following the EXTEND requirement, the Topology View and the List Views are very similar to heavily used previous views in terms of visual representation, but are augmented with full interaction capabilities as integrated linked views. The **Matrix View** is designed to provide an overview of signal communication using an adjacency matrix visual representation. The **SignalPath View** acts as a node-link detail view for a filtered set of signal paths traversed by the currently selected signals. Additional minor views are available for filtering the data at hand and showing the visual encoding legend.

By default, the Topology View, the Matrix View, and the ECU List View are shown simultaneously; the Signal List and the minor views are sequentially available via tabs in the ECU List, and the SignalPath View can be accessed on demand.

Figure 3 and all subsequent figures show the system with a sanitized real-world example dataset of roughly 8000 signals specified between 93 ECUs and 9 buses, yielding 3000 unidirectional logicalcount edges. Additional screenshots as well as a video showing the look and feel of RelEx on the same dataset are available as supplemental material.

All visual encoding choices in all three of the network-centric views were informed by the empirical work on the strengths and weaknesses of matrix versus node-link visual encodings [13]. Three central findings pertain to our design problem. First, node-link encoding is a poor choice for large dense graphs because they become extremely cluttered. Second, matrix encoding does scale well, even for dense graphs. Third, path tracing is poorly supported by matrix encoding.

The bottom half of Table 1 summarizes the capabilities of each main view in terms of the data abstractions. We now discuss the capabilities of and design decisions underlying each of these views in more detail.

### 5.1 Topology View

The Topology View uses a node-link visual encoding, a perceptually appropriate choice because the physical network is small and sparse. The Topology View incorporates an automatic layout algorithm designed to mimic as closely as possible the heavily used manually generated views that were familiar to the engineers, following the EXTEND requirement, in order to maximize our tool's usability, learnability, and user acceptance. Details about the layout algorithm can be found in the first author's dissertation [38].

This choice of automation also follows the INTEGRATE requirement by ensuring that the drawings reflect the latest database updates, avoiding the barrier of manual drawing updates. Also, the INTERACTION requirement is more easily met with an automatically constructed view than with an approach where dynamic information is overlaid upon a static drawing.

ECUs are represented as labeled rectangles, with a thick black outline around gateway ECUs. The central gateway ECU is placed in the top center, with other ECUs connected to it sorted into vertical stacks according to their bus connectivity. Buses are drawn with orthogonal lines, color-coded according to domain conventions, with labels at the bottom. Although manual drawings sometimes use double stacks with ECU rectangles on both sides of bus lines, our system creates only a single stack per group for algorithmic simplicity. The algorithm reorders stacks and lines to avoid crossings of bus edges on top of ECU nodes, and to minimize edge crossings. The view supports navigation through zooming and panning.

### 5.2 Matrix View

The Matrix View was designed to fulfill the OVERVIEW requirement by providing a big-picture view of the logicalcount network, showing the possible communication between send and recv ECUs across all signals. We chose an adjacency matrix representation because the logicalcount network is large and dense.

The signal matrix positions send ECUs along the top row and recv ECUs down the left column. The visual representation of the ECUs along the outside is with labeled rectangles for consistency with the Topology View, with the addition of small arrows to clarify the direction of communication. At each crossing of a *send column* and a *recv row* the total number of signals exchanged between that pair of ECUs are represented by a square whose size encodes the logicalcount; we call these **communication-boxes**. The largest box size showing counts of over 100 signals also has a thick black outline to support preattentive search for communication hotspots [43, 16]. The size coding is documented in a legend at the bottom of the view.

Although the users' main interest is in overviewing the logical network, we intentionally decided to use the logicalcount aggregation resulting in a simple directed graph for visual overview purposes rather than trying to directly show the more complex logical multigraph. By using the logicalcount aggregation, the user can quickly detect whether edges exist between nodes in the logical multigraph; the individual signal edges can then be resolved interactively as described below. Our lead users found this to be an adequate and highly valuable approach for their data and tasks.

The default ordering of the ECUs within the matrix is alphabetical by name. The user can instead choose **bus-sorted** mode, where the ECUs are ordered by bus connectivity, and alternating groups of bus

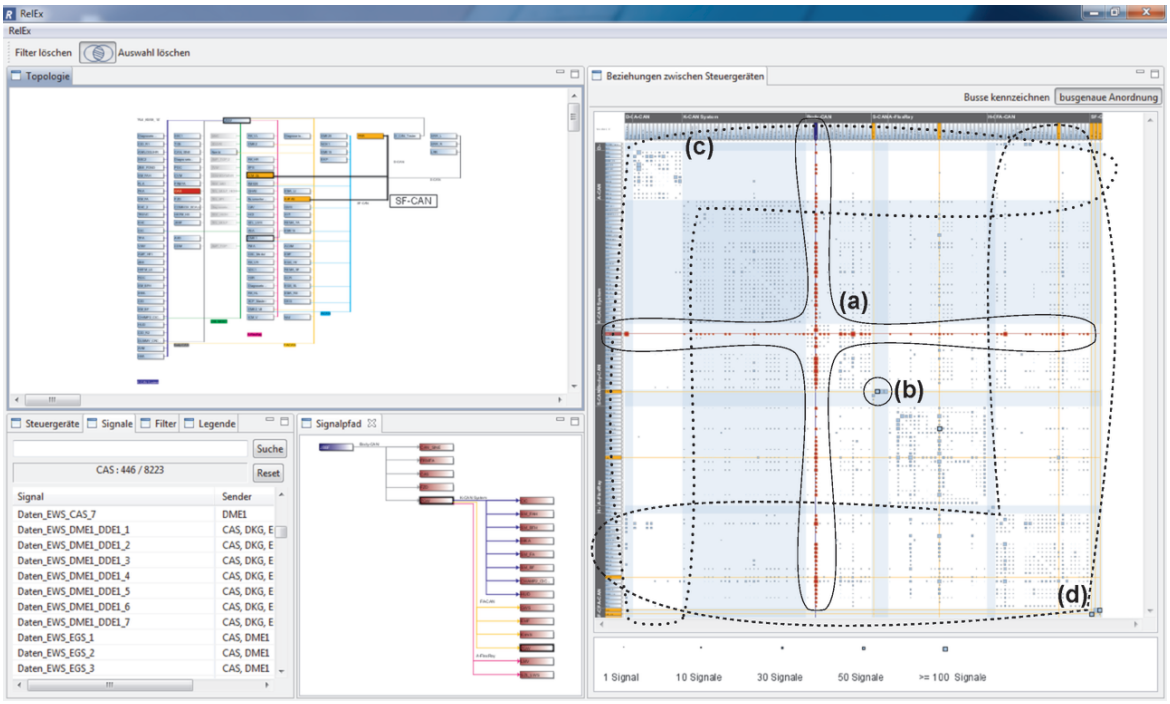



Fig. 3. RelEx with the four main views opened: Topology View (upper left), List View with the Signal tab active (lower left), SignalPath View (lower middle), and the Matrix View (right side). The annotations in the matrix sorted by bus systems show (a) a *crossbeam*, (b) communication *hotspots*, (c) the bus system communication *introvert* pattern – most of the communication-boxes are within the white square on the diagonal indicating communication within a bus, (d) the *extrovert* pattern – many boxes are outside the diagonal square indicating communication with other buses.

systems have light blue backgrounds to create visually distinguishable stripes. Because gateway ECUs are connected to more than one bus, in this mode we had to decide between representing them redundantly or leaving out all but one bus connection, a decision also faced by the designers of Constellation [29] and NodeTrix [19]. We chose to show them redundantly based on feedback from the lead users. We maintain awareness of the redundancy by simultaneously highlighting all instances of the same node on hover or selection.

This view supports navigation through panning and constrained zooming to inspect regions in more detail. Zooming is constrained to be a single-step increase of magnification to a fixed level, and the relevant labels on the periphery are pinned to stay visible at all times as the user pans. These interaction choices were influenced and refined by the heuristic usability study we conducted and by feedback from our lead users.

### 5.3 SignalPath View

Following the PATH requirement, the SignalPath View, as the name implies, supports path tracing tasks, and thus a node-link visual encoding is the most appropriate choice. Because the signalpath network is very dense, we do not attempt to show an overview representation of the whole network. This view is a detail view, showing only the signal paths corresponding to a selected signal. It can be considered to be a view of the signalpath network filtered by the selected signal, containing only the edges and nodes traversed by it.

The style is visually consistent with the Topology View in terms of shapes, colors, and labels, and the layout shares many features such as orthogonal edge routing and vertical stacks. However, it is important to note that it shows a completely different data abstraction: a filtered version of the signalpath network rather than the physical network.

We observed many participants sketching this kind of view for themselves on paper, after internally synthesizing the results of a sequence of simple queries. They typically created a left-right orthogonal node-link view with vertical stacking of elements according to bus systems, following the visual conventions of the Topology View. We developed an automatic layout algorithm that is faithful to this style, following the EXTEND requirement.

### 5.4 List Views and Other Views

The List View can show one of its four tabs at any time. The main two tabs are text-based detail browsers, the ECU List and the Signal List. The lists are alphabetically ordered by the name of the ECU and signal respectively and provide detailed information as well as related elements such as, for instance, sending or receiving ECUs of a particular signal in the Signal List. They therefore allow for conducting traditional simple queries supporting the EXTEND requirement.

In addition, each List View includes a search box that can be used to filter the lists. Selecting a list item results in highlighting not only that item, but also related elements in all other views, as described in more detail below. In the other direction, selection in any of the other views automatically initiates a search request in the List Views.

A third tab holds the Filter View that allows users to interactively filter elements out so that they are completely excluded from the visual exploration and representation in all views – as opposed to List View filtering. The fourth tab holds the Legend View that provides a complete legend of all visual encodings used, most importantly all color codings. These two views were created in response to direct requests from the lead users.

### 5.5 View Coordination

The brushing and linking between the views was carefully designed to support rich exploration of connections in the network (RICH). The core operations for this purpose are hover and select operations, as lighter and heavier weight ways to indicate choice.

The visual encoding for highlighting ECU and signal elements is orange for hover and light red for selection in all visualization views. Where send and recv ECUs need to be distinguished, they are colored with saturated blue and red respectively. These colors follow domain conventions and were determined based on feedback from the engineers. Highlighting bus system elements is done by increasing the linewidth in order to retain their color information.

Hover and selection of an element not only highlights the element itself in all views, but also all elements directly connected to the chosen element, in a way that depends on the element type:

- **ECU:** *Hover/select in Topology, SignalPath, Matrix, List Views.* In the Matrix View, horizontal and vertical lines "beam" out from the location of the chosen ECU along the periphery at the top and left, where all communication-boxes along these lines are highlighted. The resulting **crossbeam**, which spans the matrix, will be centered on the diagonal, as shown in Figure 3(a). These lines draw attention to all the signals sent or received by the chosen ECU. If the chosen element appears more than once because the matrix is bus-sorted, multiple crossbeams will appear.
- **Bus:** *Hover/select in Topology, SignalPath Views.* In the Topology View, all ECUs connected to a chosen bus are also highlighted, as shown with the orange ECUs connected to the thick black line of the `SF-CAN` bus in the upper left of Figure 3. This highlighting supports faster detection of all connected ECUs, especially if the ECUs of a bus are connected to more than one bus and therefore located in different columns.
- **Signal:** *Select in List View.* Selecting a signal from the Signal List View highlights all elements directly involved in the corresponding signal paths for its routing in all views, and optionally also opens a SignalPath View showing those signal paths explicitly. The highlighted elements include all ECUs that send or receive the signal and all bus systems that transport it; thus, the highlighting in the Topology View provides a quick way to see what subset of the familiar physical network is affected by the chosen signal. The highlighted elements also include all communication-boxes that incorporate the chosen signal in the Matrix View, in addition to the affected ECUs that are connected with vertical blue (send) and horizontal red (recv) crossbeam lines. Figure 4 shows the Matrix View highlighting a single *wide-spread* signal that touches many ECUs on the network.
- **Communication-box:** *Hover/select in Matrix View.* Choosing a communication-box in the Matrix View, representing signal exchange between a pair of ECUs, creates a crossbeam centered at that chosen box as described for signals. Sending and receiving ECU and transporting bus are highlighted in the Topology View, and the Signal List is filtered to show the loading of the selected signalcount edge.

Thus, although the visual encoding in some of the views is straightforward, the interaction design through view coordination allows users to quickly explore with both simple and more complex connection queries following the RICH requirement. For example, selecting a bus system in the Topology Map automatically shows all signals that are sent via this bus in the Signal List and all connected ECUs in the ECU List. Selecting a communication-box in the Matrix View updates the lists with sending and receiving ECUs, as well as all signals that are sent between these ECUs.

We added many other features in response to requests from the users. Most notably, we support multiple selections and allow the user to choose between combining them into AND or OR filtering operations; we also added a simple wizard to support for editing elements directly within RelEx without the need to change to another tool.

### 5.6 Implementation

We implemented RelEx in Java using Eclipse's Plug-in Development Environments [11]. The sophisticated window management framework of Eclipse works well for building a multiple coordinated view systems. We further integrated RelEx as an plugin into NetDB-2, the experimental new NetDB version. This close integration, following the INTEGRATE requirement, allowed everybody involved in testing the new database, including all of our collaborators, to use RelEx without any additional time overhead. For users without access to this test version and for users at supplier companies, we also provided a standalone version that works with exported data.

## 6 RESULTS AND VALIDATION

Our particular efforts to validate the utility of RelEx focused on two aspects: first, determining whether the carefully designed coordination of our multiple views provided benefits to the engineers, compared to their previous tools; and second, whether the novel Matrix View overview was helpful for engineers' tasks. Following previously suggested guidelines for validating design studies [27, 28], we present usage examples we found where our participants successfully used RelEx for their daily work, feedback and subjective ratings, and discuss the adoption of RelEx in daily engineering practice both during and after the longitudinal studies. These results summarize all formative and summative studies discussed in Section 3.

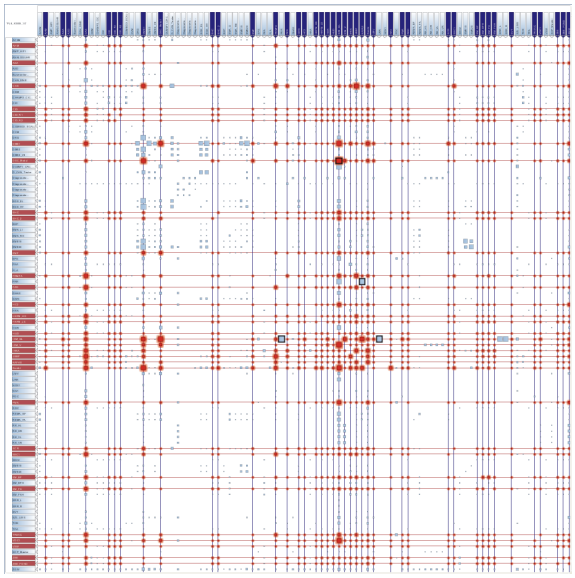


Fig. 4. The Matrix View reveals a pattern of a *wide-spread* signal that touches many of the ECUs in the network.

### 6.1 Usage Examples

From our longitudinal field studies, we derived several usage examples that show how RelEx helped our participants in conducting daily tasks, and how they gained novel insights into the data by using it.

In terms of how our multiple view approach extended previous exploration practices, we found that all participants were able to quickly learn all interactions and to productively use them to simplify or speed up some of their tasks. For instance, three engineers used RelEx for replacing ECUs with other ECUs in four ways. They used it to identify and adapt all communication partners. They also used it to decide whether two ECUs might be consolidated or not by investigating differences and connections between the ECUs in question. Furthermore, they used RelEx for finding out to which bus to connect a novel ECU, and for deciding how to re-route signals over the network. Another engineer who was relatively new in the area of in-car network engineering, with only two months of experience, used RelEx frequently to familiarize himself with the network topology and communication processes in the network, and especially to understand the intensity of communication between particular ECU pairs. Furthermore, RelEx was used for exploring the current network specification and to understand how to improve it by gaining new insights into the data, as we discuss below. Finally, our prototype was used for ordinary tasks such as to identify all communication partners of an ECU, to investigate gateway communication, to explain decisions and facts to colleagues, or simply to refer to the topology map if a printed version was not near to hand.

We documented four mid-level subtasks of complex traffic optimization, as described in our task analysis, in which the matrix overview provided benefits to the engineers. In the first three cases, the task of finding these configurations was at least partially supported by previous tools, but carrying it out was slow, tedious, and error-prone. In the last case, the task was not supported at all by previous tools.

**Communication hotspots:** The Matrix View supported fast detection of *communication hotspots*, namely heavily travelled pathways

between ECUs, because of the size coding of the communication boxes and highly visible black frame around the largest ones, as shown in Figure 3(b). Hotspot detection was known to be an important problem, but using previous tools users had either to know a priori where hotspots might be, or to identify them via sequential browsing using simple textual queries in a slow and error-prone process. In contrast, the matrix allowed for a robust and fast identification of these hotspots.

**Local/global communicator ECUs:** The combination of the Matrix View and hover/select highlighting capabilities made it easy to understand the number of communication partners for a specific ECU. It was easy to spot *local* communicators that only exchanged signals with a few other ECUs, versus *global* communicators that exchanged signals with nearly all ECUs in the network, as shown by the red highlighted cross in Figure 3. The fast detection of global communicators helped our participants to estimate potential impacts of changing or replacing this ECU. Just as with the above example, identifying these patterns was far faster with RelEx than with the simple queries supported by previous tools. The benefits of matrix views for distinguishing local from global were previously noted in a use case of analyzing call graph matrices of large software projects [44].

**Well-directed/wide-spread signals:** The Matrix View also helped the engineers to quickly get an impression of the importance of a selected signal. Well-directed signals exchanging information between two dedicated ECUs result in only one or two communication-boxes being highlighted in the Matrix View when that signal is selected. In contrast, wide-spread signals jump out because many communication-boxes are highlighted, as shown by the several dozen crossbeams in Figure 4; the red horizontal beams emanate from the send ECUs, and the blue vertical beams from the recv ones. Our participants preferred to use the much faster RelEx than the previous tools for this purpose.

**Introvert/extrovert bus systems:** The Matrix View also helped engineers understand communication structures between entire bus systems. Sorting the Matrix View by bus helped engineers to distinguish between **introvert** bus systems that have little or no communication with other network sub-systems, as shown in Figure 3(c), and **extrovert** bus systems that communicate with many others, as shown in Figure 3(d). The easy access to all signals that are exchanged across different bus systems turned out to be particularly helpful in network optimization undertakings. All participants mentioned that communication across bus systems is expensive and that the Matrix View provided a good starting point for hypothesizing about traffic optimization in terms of reducing gateway communication costs. Previous tools did not support this kind of analysis at all.

### 6.2 Feedback and Subjective Ratings

During our think-aloud studies, we explicitly focused on comparing RelEx with the heavily used previous tools TopoMaps and NetDB. In doing so, we triggered more discussions about room for future improvement and possible extensions of the RelEx approach. When interpreting these results we were careful to keep in mind experimental demand characteristics, namely that study participants are often eager to please researchers [6, 23].

We were particularly interested in comparing our automatic Topology View layout to manually drawn TopoMaps. The general feedback was more positive than we expected given the known limitations of automatic layout algorithms, with the mean similarity ranked as 1.8 and clear arrangement as 2.1, on a scale where 1 is best and 5 worst. We were also given many suggestions for improvement. Some participants complained that the position of our ECUs is not exactly the same as the position in the original maps. Others suggested that all gateways should be positioned more centrally, and that the layout should support the representation of mutually exclusive ECUs just as manually drawn layouts do.

When comparing RelEx to NetDB, the feedback was highly positive, with better ratings for 95% of all tasks tested. This result is heartening but not surprising, given that RelEx was designed to support complex connection exploration while the NetDB interface was developed for data management purposes and simple exploration tasks. The major benefits of RelEx most frequently stated by our participants were the overview RelEx provides, the rich and usable exploration opportunities, its comprehensibility, the ability to easily communicate with colleagues using RelEx, and the ability to speed up and simplify a variety of their daily tasks. Four participants also particularly stressed the value of the SignalPath View as it shows all relevant elements of signal paths in an easily understandable and usable way, while with previous tools this information had to be derived with a manual step-by-step process. Several participants even advocated replacing Signaller with RelEx as soon as possible for company-wide use.

The main limitation of RelEx was the lack of support for aspects of communication that are not currently addressed in the tool at all, most notably a networking layer in which signals are packed and unpacked into combined message units. We intentionally abstained from adding this additional layer of information based on feedback from our lead users during the user-centered design process. Our lead users argued that a solution for signals, ECUs and bus systems was most urgent and should be implemented and validated before extending it to further elements.

### 6.3 Adoption

It is usually difficult to convince users to spend time learning novel solutions in domains with strong time and cost restrictions, especially when they are strongly accustomed to and effective with previous tools [39]. Given the strong restrictions on the available time of engineers in our company setting, we were pleasantly surprised by how frequently our participants used RelEx in their routine work practice during the study: 5 participants used it weekly, and 2 participants used it daily. Most of our participants also continued to use RelEx after the study and recommended it to colleagues for usage, resulting in a user base of approximately 15 engineers a few months after the study's end.

## 7 Conclusion and Future Work

The RelEx visualization system was successful in speeding up the work carried out by its target users, engineers actively changing a complex combination of overlay communication networks. It was created as part of a full user-centered design cycle that featured close contact with the target users at every step, including pre-design observation in the field, formative feedback during iterative development, and summative post-deployment field studies. The latter is the first post-deployment study of a matrix-based visualization and provides direct evidence of their benefits in the field for real-world tasks.

A further major contribution of this design study is the data and task abstraction that is fundamentally different from previous design studies in network visualization, leading to different implications for network visualization design. While previous work mainly focused on cluster and node centrality analysis, our work supports the tasks of actively changing the data, routing communication, and optimizing traffic. Rather than focusing on a single network, our data abstraction is an overlay communication network where a logical layer expressing communication paths is mapped onto a physical layer with physical edges. In our application the true problem lies in exploring path scalability. In contrast, most of the previous work has concentrated on node scalability. We argue that such differences can be found in many other application domains beyond in-car network development.

We provide this design study as one step towards broadening the visualization community's knowledge about the tasks and data involved in visual network analysis, but there is still far to go. We call for more such studies in the future across a disparate set of application domains, in order to build up a more complete understanding of the space of task and data abstractions, and their implications for visualization design. Eventually, a solid understanding of the diversity of tasks in network analysis will help us to develop broadly applicable network visualization techniques with widespread impact.


## Acknowledgments

We thank the BMW Group and all participants for making this work possible, especially Ben Krebs and Martin Schlaugath who actively helped shape these ideas. We also thank Matthew Brehmer and Jessica Dawson for comments on paper drafts.

## REFERENCES


[1] D. L. Alderson and J. C. Doyle. Catching the "network science" bug: Insight and opportunity for the operations researcher. *Operations Research*, 56(5):1047–1065, 2008.

[2] A. Barsky, T. Munzner, J. Gardy, and R. Kincaid. Cerebral: visualizing multiple experimental conditions on a graph with biological context. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 2008)*, 14(6):1253–60, 2008.

[3] S. Borgatti. Centrality and network flow. *Social Networks*, 27(1):55–71, 2005.

[4] R. Bosch GmbH. *Autoelektrik/Autoelektronik, 5. Auflage*. Vieweg Verlag, Braunschweig, Germany, 2007.

[5] R. Bosch GmbH. *Kraftfahrzeugtechnisches Taschenbuch. 26. Auflage*. Vieweg Verlag, Braunschweig, Germany, 2007.

[6] B. Brown, S. Reeves, and S. Sherwood. Into the wild: Challenges and opportunities for field trial methods. In *Proc. ACM Conf. Human Factors in Computing Systems (CHI)*, pages 1657–1666. ACM, 2011.

[7] M. Broy. Challenges in automotive software engineering. In *Proc. ACM Intl. Conf. Software Engineering (ICSE)*, pages 33–42, 2006.

[8] S. Card, J. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[9] A. Dix and G. Ellis. Starting simple - adding value to static visualisation through simple interaction. In *Proc. Advanced Visual Interfaces (AVI)*, pages 124–134, 1998.

[10] S. P. Dow, A. Glassco, J. Kass, M. Schwarz, D. L. Schwartz, and S. R. Klemmer. Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. *ACM Trans. Computer-Human Interaction (ToCHI)*, 17(4):1–24, 2010.

[11] Eclipse Foundation. Eclipse. http://www.eclipse.org/, last visited: 03/12.

[12] F. Fischer, F. Mansmann, D. Keim, and S. Pietzko. Large-scale network monitoring for visual analysis of attacks. In *Proc. Intl. Workshop Visualization for Computer Security (VizSec)*, pages 111–118, 2008.

[13] M. Ghoniem, J. Fekete, and P. Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization*, 4(2):114, 2005.

[14] K. Grimm. Software technology in an automotive company: major challenges. In *Proc. Intl. Conf. Software Engineering (ICSE)*, pages 498–503, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[15] G. Hayes. The relationship of action research to human-computer interaction. *ACM Trans. Computer-Human Interaction (ToCHI)*, 18(3):15, 2011.

[16] C. Healey, K. Booth, and J. Enns. Harnessing Preattentive Processes for Multivariate Data Visualization. In *Proc. Graphics Interface (GI)*, pages 107–117, 1993.

[17] J. Heer and d. boyd. Vizster: Visualizing online social networks. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 32–39, 2005.

[18] H. Heinecke. Automotive System Design – Challenges and Potential. In *Proc. IEEE Conf. Design, Automation and Test in Europe (DATE)*, pages 656–657, 2005.

[19] N. Henry, A. Bezerianos, and J. Fekete. Improving the Readability of Clustered Social Networks using Node Duplication. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 2008)*, 14(6):1317–1324, 2008.

[20] N. Henry and J. Fekete. MatrixExplorer: a dual-representation system to explore social networks. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 2006)*, 12(5):677–684, 2006.

[21] N. Henry, J. Fekete, and M. McGuffin. NodeTrix: a hybrid visualization of social networks. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 2007)*, 13(6):1302–1309, 2007.

[22] K. Holtzblatt and S. Jones. Contextual inquiry: A participatory technique for system design. In *D. Schuler, A. Namioka (eds.): Participatory Design: Principles and Practices*, pages 177–210. Lawrence Erlbaum Associates, 1993.

[23] M. J. Intons-Peterson. Imagery paradigms: How vulnerable are they to experimenters' expectations? *Journ. Experimental Psychology - Human Perception and Performance*, 9:394–412, 1983.

[24] H. Kang, C. Plaisant, B. Lee, and B. Bederson. NetLens: iterative exploration of content-actor network data. *Information Visualization*, 6(1):18–31, 2007.

[25] B. Lee, C. Plaisant, C. Parr, J. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proc. AVI Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization (BELIV)*. ACM, 2006. Article 14.

[26] G. Leen, D. Heffernan, and A. Dunne. Digital networks in the automotive vehicle. *Computing & Control Engineering Journ.*, 10(6):257–266, 1999.

[27] T. Munzner. Process and pitfalls in writing information visualization research papers. In *Information Visualization: Human-Centered Issues and Perspectives*, pages 134–153. Springer LNCS 4950, 2008.

[28] T. Munzner. A nested model for visualization design and validation. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 2009)*, 15(6):921–928, 2009.

[29] T. Munzner, F. Guimbretière, and G. Robertson. Constellation: a visualization tool for linguistic queries from MindNet. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 132–135, 1999.

[30] G. Namata, B. Staats, L. Getoor, and B. Shneiderman. A dual-view approach to interactive network visualization. In *Proc. ACM Conf. Information and Knowledge Management (CIKM)*, pages 939–942, 2007.

[31] C. B. Nielsen, S. D. Jackman, I. Birol, and S. J. M. Jones. ABySS-Explorer: visualizing genome sequence assemblies. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 2009)*, 15(6):881–8, 2009.

[32] D. A. Norman. *The Design of Everyday Things*. Doubleday, 1988.

[33] C. North and B. Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *Proc. ACM Conf. Advanced Visual Interfaces (AVI)*, pages 128–135, 2000.

[34] A. Perer, I. Guy, E. Uziel, I. Ronen, and M. Jacovi. Visual social network analytics for relationship discovery in the enterprise. In *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)*, pages 71–79, 2011.

[35] A. Perer and B. Shneiderman. Integrating Statistics and Visualization: Case Studies of Gaining Clarity during Exploratory Data Analysis. *Proc. ACM Conf. Human Factors in Computing Systems (CHI)*, pages 265–274, 2008.

[36] A. Pretorius and J. Van Wijk. Visual inspection of multivariate graphs. *Computer Graphics Forum (Proc. Eurovis 2008)*, 27(3):967–974, 2008.

[37] A. Pretschner, M. Broy, I. Kruger, and T. Stauner. Software engineering for automotive systems: A roadmap. In *Proc. ICSE Workshop on Future of Software Engineering (FOSE'07)*, pages 55–71. IEEE Computer Society, 2007.

[38] M. Sedlmair. *Visual Analysis of In-Car Communication Networks*. PhD thesis, Faculty of Mathematics, Computer Science and Statistics, University of Munich, 2010.

[39] M. Sedlmair, P. Isenberg, D. Baur, and A. Butz. Information Visualization Evaluation in Large Companies: Challenges, Experiences and Recommendations. *Information Visualization*, 10(3):248–266, 2011.

[40] M. Sedlmair, P. Isenberg, D. Baur, M. Mauerer, C. Pigorsch, and A. Butz. Cardiogram: Visual analytics for automotive engineers. In *Proc. ACM Conf. Human Factors in Computing Systems (CHI)*, 2011.

[41] M. Sedlmair, M. Meyer, and T. Munzner. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 2012)*, 2012. In press.

[42] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proc. AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV)*, 2006.

[43] A. Treisman. Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing*, 31(2):156–177, 1985.

[44] F. van Ham. Using multilevel call matrices in large software projects. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 227–232, 2003.

[45] F. van Ham, H. Schulz, and J. Dimicco. Honeycomb: Visual Analysis of Large Scale Social Networks. In *Proc. Intl. Conf. Human-Computer Interaction (INTERACT)*, pages 420–442. Springer-Verlag, 2009.

[46] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual Analysis of Large Graphs. *Proc. Eurographics*, 2010.

[47] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2004.

[48] C. Weaver. Building highly-coordinated visualizations in Improvise. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 159–166, 2004.

[49] W. Willinger, D. Alderson, and J. C. Doyle. Mathematics and the Internet: A source of enormous confusion and great potential. *Notices of the American Mathemematical Society*, 56(5):586–599, 2009.