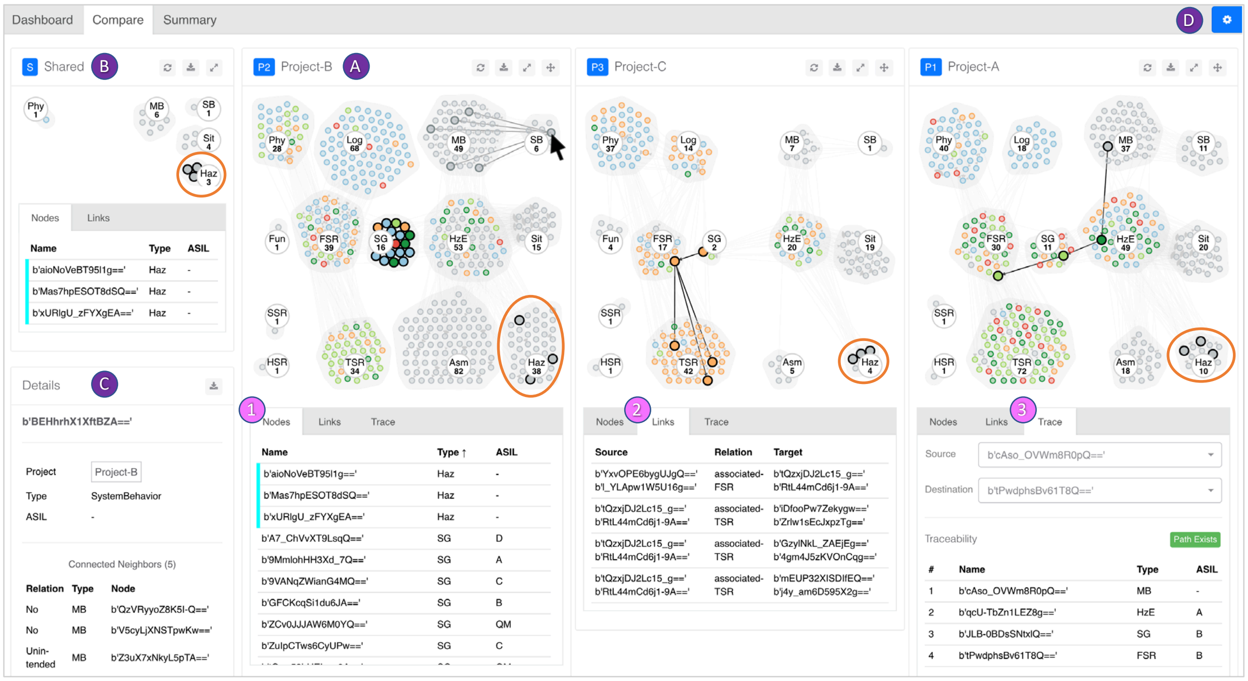
SafetyLens: Visual Data Analysis of Functional Safety of Vehicles

Arpit Narechania* — Georgia Institute of Technology
Ahsan Qamar† — Ford Motor Company
Alex Endert‡ — Georgia Institute of Technology



Shared — Nodes

Name	Type	ASIL
b'aioNoVeBT95l1g=='	Haz	-
b'Mas7hpESOT8dSQ=='	Haz	-
b'xURlgU_zFYXgEA=='	Haz	-

Details

b'BEHhrhX1XftBZA=='

Project	Project-B
Type	SystemBehavior
ASIL	-

Connected Neighbors (5)

Relation	Type	Node
No	MB	b'QzVRyyoZ8K5I-Q=='
No	MB	b'V5cyLjXNSTpwKw=='
Unin-tended	MB	b'Z3uX7xNkyL5pTA=='

Project-B — Nodes

Name	Type ↑	ASIL
b'aioNoVeBT95l1g=='	Haz	-
b'Mas7hpESOT8dSQ=='	Haz	-
b'xURlgU_zFYXgEA=='	Haz	-
b'A7_ChVvXT9LsqQ=='	SG	D
b'9MmlohHH3Xd_7Q=='	SG	A
b'9VANqZWianG4MQ=='	SG	C
b'GFCKcqSi1du6JA=='	SG	B
b'ZCv0JJJAW6M0YQ=='	SG	QM
b'ZuIpCTws6CyUPw=='	SG	C

Project-C — Links

Source	Relation	Target
b'YxvOPE6bygUJgQ==' b'I_YLApw1W5U16g=='	associated-FSR	b'tQzxjDJ2Lc15_g==' b'RtL44mCd6j1-9A=='
b'tQzxjDJ2Lc15_g==' b'RtL44mCd6j1-9A=='	associated-TSR	b'iDfooPw7Zekygw==' b'Zrlw1sEcJxpzTg=='
b'tQzxjDJ2Lc15_g==' b'RtL44mCd6j1-9A=='	associated-TSR	b'GzylNkL_ZAEjEg==' b'4gm4J5zKVOnCqg=='
b'tQzxjDJ2Lc15_g==' b'RtL44mCd6j1-9A=='	associated-TSR	b'mEUP32XISDIfEQ==' b'j4y_am6D595X2g=='

Project-A — Trace

Source: b'cAso_OVWm8R0pQ=='
Destination: b'tPwdphsBv61T8Q=='

Traceability — Path Exists

#	Name	Type	ASIL
1	b'cAso_OVWm8R0pQ=='	MB	-
2	b'qcU-TbZn1LEZ8g=='	HzE	A
3	b'JLB-0BDsSNtxlQ=='	SG	B
4	b'tPwdphsBv61T8Q=='	FSR	B

Fig. 1. The SafetyLens Compare view: **(A)** Project Panel comprising the visualization canvas and three tabs: (1) Nodes, (2) Links, and (3) Trace, **(B)** Shared Nodes Panel with three nodes (*Hazards*) selected and highlighted across all three projects (ellipses), **(C)** Detail View Panel showing details of a node (*System Behavior*) hovered in *Project-B* (↖), and **(D)** opens into a Control Panel comprising search and filter controls.

Abstract— Modern automobiles have evolved from just being mechanical machines to having full-fledged electronics systems that enhance vehicle dynamics and driver experience. However, these complex hardware and software systems, if not properly designed, can experience failures that can compromise the safety of the vehicle, its occupants, and the surrounding environment. For example, a system to activate the brakes to avoid a collision saves lives when it functions properly, but could lead to tragic outcomes if the brakes were applied in a way that's inconsistent with the design. Broadly speaking, the analysis performed to minimize such risks falls into a systems engineering domain called Functional Safety. In this paper, we present SafetyLens, a visual data analysis tool to assist engineers and analysts in analyzing automotive Functional Safety datasets. SafetyLens combines techniques including network exploration and visual comparison to help analysts perform domain-specific tasks. This paper presents the design study with domain experts that resulted in the design guidelines, the tool, and user feedback.

Index Terms—Visual data analysis, Design study, Network visualization, Functional safety, Automotive engineering

1 Introduction

Modern automobiles come integrated with a complex suite of systems (e.g. Anti-lock Braking System) that deploy hardware components integrated with millions of lines of software code. In a system that is not designed properly, the sheer number and complexity of components can result in failures posing risks which if not appropriately handled can be detrimental to the safety of the vehicle, its occupants, and the surrounding environment. Recalls can tarnish a manufacturers brand, straining customer trust and loyalty. It is important to design and manufacture vehicles in a way that may assist in reducing the risk. To analyze a given design, several systems engineering analysis techniques exist, which include but are not limited to Failure Mode Effect Analysis [52], Fault Tree Analysis [35], etc.

Our work here is focused on Functional Safety in the automotive sector. To understand what Functional Safety is and how domain experts use it to ensure vehicle safety, consider an example scenario of the **"Adaptive Cruise Control"** system whose objective is to accelerate (or decelerate) the vehicle to maintain a reference speed as set by the driver. However, an improperly designed system could malfunction causing the vehicle to continuously accelerate (or decelerate). For example, when the vehicle is **overtaking**, if acceleration was to continue, and not be controlled by the driver, it can lead to a collision. To mini-

*e-mail: arpitnarechania@gatech.edu
†e-mail: aqamar2@ford.com
‡e-mail: endert@cc.gatech.edu

Manuscript received 30 Apr. 2020; revised 31 July 2020; accepted 14 Aug. 2020. Date of publication 13 Oct. 2020; date of current version 15 Jan. 2021. Digital Object Identifier no. 10.1109/TVCG.2020.3030382

mize risk associated with this mishap, a goal to **not accelerate more than required** is set. To achieve this goal, a requirement to be able to **override acceleration when brake pedal is pressed** is defined which in turn requires **monitoring of the brake pedal sensor**. Furthermore, this may require software code to **compare brake pedal sensor outputs for confidence** and possibly **additional sensors**. The process of setting these goals, testing the conditions, and analyzing the results are all part of Functional Safety.

The above example showcases one scenario of the *"Adaptive Cruise Control"* system. There are other vehicle systems (e.g., Anti-lock Braking System, Parking Assist System) with their potential scenarios (similar or different sets of expected functions, associated risks, and safety goals). Finally, car companies have multiple car models, with updates from year to year, each with potentially different systems. This makes Functional Safety datasets large. Figure 2 illustrates the *"Adaptive Cruise Control"* scenario in the form of a network with each node representing an entity. The purple boxes indicate the corresponding Functional Safety terminologies for the entities in the example (described in Section 2.1).

Functional Safety domain experts are engineers and analysts who can identify and assess hazards, set goals to minimize risks associated with these hazards, and define safety requirements to achieve these goals. They must also ensure smooth implementation of these goals by asking questions such as *"Are there any scenarios where there is an unreasonable risk for which there is no safety goal?"*, *"Are there hazards in the current project that were also addressed in another project? If so, what safety goals were identified for them?"*, and others.

We designed and developed SafetyLens, a visual data analysis tool to assist domain experts to better understand functional safety datasets. Through a three-month-long user-centered design process comprising interviews, demos, and discussions with domain experts from a multinational automobile company, we derived design goals and user tasks that drove our work. After a final phase of user feedback on the prototype, SafetyLens is currently being deployed.

The primary contributions of this paper include (i) design goals and challenges in supporting visual analysis of Functional Safety datasets, (ii) SafetyLens, a visual data analysis tool that helps domain experts visualize and interact with functional safety datasets, (iii) usage scenarios that illustrate how SafetyLens can aid the current analysis processes of domain experts, and (iv) feedback from domain experts about the design, functionality, and impact of SafetyLens.

2 Industrial Background

Functional Safety analyses are performed in several domains such as Military Aviation [14], Space [49], Medical [32], Automotive [31], etc. Our work here is focused on Functional Safety of vehicles within the automotive systems engineering domain. ISO 26262 titled "Functional Safety - Road Vehicles" [31] is the international standard for the design and development of automotive electrical and electronic systems that makes Functional Safety a part of the automotive product development life-cycle. It is a risk-based safety standard, where the risk from hazardous situations is qualitatively assessed leading to the definition of safety measures that minimize these risks.

2.1 Concepts

This section introduces concepts and terminology associated with Functional Safety from the perspective of ISO 26262. These also describe the Functional Safety datasets used in SafetyLens. When modeled as a network, they make up the nodes, links, and attributes.

SYSTEM refers to a classification of automotive electrical and electronic systems such as "Adaptive Cruise Control" and "Chassis" on which Functional Safety is performed.

PROJECT refers to an instance of a *System*, e.g., *"Chassis for Model Z"*. This is also an instance of a Functional Safety dataset.

ELEMENT refers to an entity within a *Project* (e.g., *"Vehicle should not accelerate more than required."*). Elements constitute nodes in a Functional Safety network.

Element Type	Abb.	Description
System Behavior	SB	An expected behavior of an electrical/electronic *System* in a vehicle.
Malfunctioning Behavior	MB	A failure or unintended behaviour of an item with respect to its design intent.
Situation	Sit	A vehicular operational scenario.
Hazard	Haz	A potential source of harm.
Hazardous Event	HzE	A dangerous condition arising out of a *Situation* and a *Hazard*.
Safety Goal	SG	A requirement to eliminate (or minimize) *Hazards*.
Functional Safety Requirement	FSR	A requirement to achieve the functional or behavioral aspects of a *Safety Goal*.
Technical Safety Requirement	TSR	A technical requirement to meet a *Functional Safety Requirement*.
Software Safety Requirement	SSR	A software-level requirement.
Hardware Safety Requirement	HSR	A hardware-level requirement.

Table 1. A description of the *Types* of *Elements* in the context of automotive Functional Safety. Other Element *Types* such as *Physical*, *Logical*, and *Assumptions* exist but are beyond the scope of this paper.

TYPE refers to a group of similar *Elements* within a *Project*, e.g. *Malfunctioning Behaviors (MB)*, described in Table 1. Type is an attribute of the nodes in Functional Safety networks.

RELATION refers to the relationship between two *Elements*. For example, a *Relation "associatedSG"* indicates that a *Safety Goal (SG)* has been assigned to a *Hazardous Event (HzE)*. Relations constitute the links in Functional Safety networks. Some of these Relations are shown in the Links column in Figure 5.

ASIL Automotive Safety Integrity Level, or ASIL, is a nominal attribute assigned to an *Element* (node) indicating its relative risk level. It can take one of four values: {A, B, C, D}. An *Element* that is assigned an *ASIL=D* indicates the highest risk. *ASIL=A* indicates the lowest risk. A special value *QM* indicates that quality management processes are sufficient to manage the identified risk [31].

2.2 Workflow

Functional Safety is a broad process ranging from the design through to the manufacturing and quality management of vehicles. The processes relevant to our work here comprise the following:

HAZARD ANALYSIS AND RISK ASSESSMENT (HARA): Domain experts first determine potential hazards that can be triggered by the malfunctioning of one or more system components and/or processes. These hazards are classified based on their *Severity (S)*, *Exposure (E)*, and *Controllability (C)*. Severity is an estimate of the extent of harm to one or more individuals that can occur in a potentially hazardous situation. Exposure is a measure of the probability of being in a hazardous situation. Controllability is an estimate of the driver's ability to avoid harm or damage through timely reactions. These are qualitative ratings with Severity={*S1, S2, S3*}, Exposure={*E0, E1, E2, E3, E4*}, and Controllability={*C0, C1, C2, C3*}. Based on these, an *Automotive Safety Integrity Level (ASIL)* is calculated and assigned to the corresponding *Hazardous Event (HzE)*. To minimize the risk associated with this hazard, a *Safety Goal (SG)* is formulated.

FUNCTIONAL SAFETY CONCEPT (FSC): From the formulated *SG*, *Functional Safety Requirements (FSR)* are derived. These define a system architecture to achieve the *SGs*.

TECHNICAL SAFETY CONCEPT (TSC): The technical safety concept comprises all *Technical Safety Requirements (TSRs)* along with their allocations to system, hardware or software elements. These are refinements of the corresponding *FSRs*.

In practice, Functional Safety analysts and engineers can use a combination of commercial and open source tools to perform their day-to-day

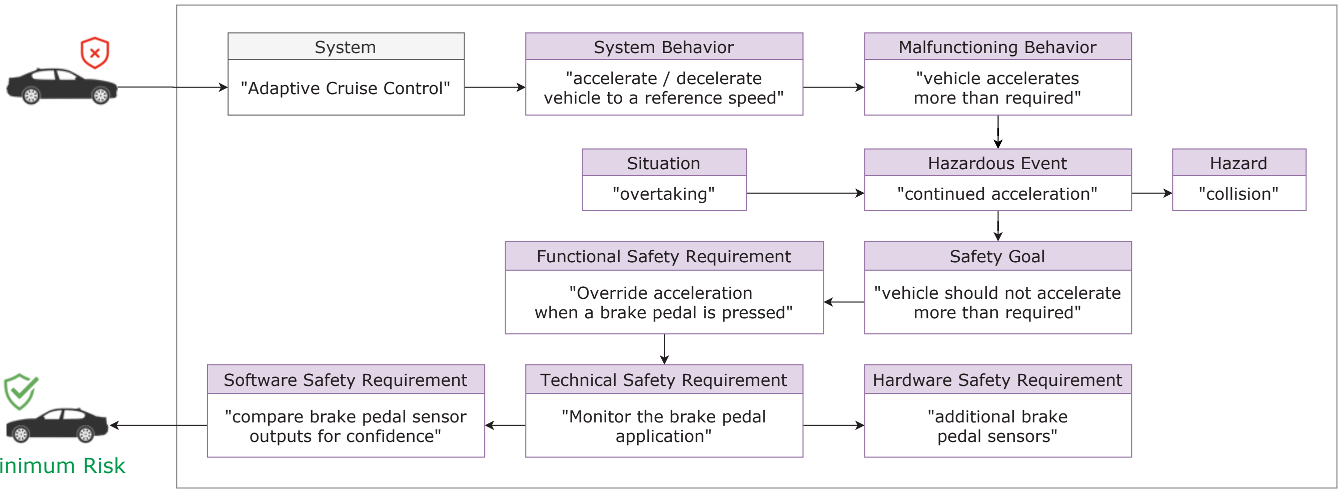

System
"Adaptive Cruise Control"
System Behavior
"accelerate / decelerate vehicle to a reference speed"
Malfunctioning Behavior
"vehicle accelerates more than required"
Situation
"overtaking"
Hazardous Event
"continued acceleration"
Hazard
"collision"
Functional Safety Requirement
"Override acceleration when a brake pedal is pressed"
Safety Goal
"vehicle should not accelerate more than required"
Software Safety Requirement
"compare brake pedal sensor outputs for confidence"
Technical Safety Requirement
"Monitor the brake pedal application"
Hardware Safety Requirement
"additional brake pedal sensors"
Minimum Risk

Fig. 2. A hypothetical scenario of an *"Adaptive Cruise Control"* system on which Functional Safety is performed. Purple boxes indicate the *Element Types* described in Table 1.

tasks. The Functional Safety processes can either be performed and documented in such tools or using traditional approaches such as Excel sheets and Word documents. For instance, our domain collaborators have utilized a domain-specific language (DSL) for ISO 26262 that enables them to import artifacts from HARA, FSC, TSC into a model. They use this model to generate Word documents with information on functional safety that was performed on a system.

There can be several challenges in the above workflow, especially in the translation of an assigned ASIL into a Technical Safety Concept. Assigning an ASIL is a complex process that can involve discussions among multiple teams. Further, ASIL assignments can be modified at different functional safety stages and points in time. The introduction of model-based systems engineering (MBSE) approaches may solve part of the problem as collaborators can now exchange information using a common language like SysML [41]. However, this may still have limitations when the scale and complexity of data increases. Finally, analyzing multiple projects to detect shared components and problems is not supported, and hence, a core focus of SafetyLens.

2.3 Data

Each *Project* constitutes an instance of a Functional Safety dataset. As seen in Figure 2, this can be modeled as a network with nodes and links. Each node represents a functional safety *Element* with seven attributes: $\{ID, Name, Type, ASIL, Severity, Exposure, Controllability\}$. A link between two nodes represents the *Relation* between two *Elements* and has three attributes: $\{Source, Target, Relation\}$.

The dataset has several properties. First, a node of a particular *Type* can be connected to zero (orphan), one, or more nodes of some other *Type*. Second, *Functional Safety Requirement (FSR)* and *Technical Safety Requirement (TSR)* nodes can be connected to nodes of the same *Type* as themselves. Third, a dataset can comprise nodes and links that are common to other projects (e.g., the same hazard (*HzE*) can arise due to different malfunctioning system behaviors (*MB*)). Fourth, these common nodes may be assigned different ASIL values. Finally, a standard Functional Safety network may consist of multiple nodes and links with the vehicle itself having multiple systems. This scale and complexity makes Functional Safety datasets large, and make the tasks of our domain experts challenging.

3 Related Work

Network Visualizations Since a functional safety dataset can be modeled as a network, we explored existing literature in network visualization systems for inspiration. There are a number of open source, free, or commercially available software [5,8,9,13,50], toolkits [27,39], and research prototypes [33,48,54]. Multiple survey reports [11,28,55] have not only summarized the state-of-the-art of network visualizations and techniques but also discussed their general evolution. We review Saket et al.'s evaluation study that indicated node-link-group visualizations as more "enjoyable" than node-link visualizations [43] as a potential technique to represent the functional safety network. A core functional safety task is to compare projects (networks) hence we review Gove, R.'s V3SPA, a visual analysis tool (for security policy workers) to explore and find differences between large complex networks (SELinux and SEAndroid security policies) [26]. Another core functional safety task is to find and visualize connections and/or paths between elements (nodes) hence we also review existing work in route tracing techniques. Candela et al. have developed Radian, an internet probe that helps visualize trace route paths [17]. Fischer et al. have developed Vistracer to investigate routing anomalies in traceroutes [21]. Zhao et al have developed MissBin which infers the existence of unseen (missing) links based on currently observed ones by involving the user to sensemake the predicted results [57].

Set Visualizations Another functional safety task involves comparing graphs to find common nodes and links. For this, we explored existing literature in set visualization. Euler and Venn diagrams are the most common methods to visualize sets and their intersections. Euler diagrams represent each set as a geometric shape and show the intersections by overlapping the shapes. Venn diagrams are like Euler diagrams but show all intersections, including empty ones. Sadana et al. [42] developed Onset to represent large-scale binary set data. Lex et al.'s UpSet technique used a *set view* and *element view* to visualize intersections in a matrix layout introducing aggregates based on groupings and queries [36]. Alsallakh et al.'s survey paper discusses the state-of-the-art of set visualization systems and techniques [2].

Visual Analytics in the Automotive Domain Perhaps most relevant to this paper is previous work on visualization applied to the automotive industry. These had been mostly in the context of scientific visualization, computer aided design (CAD), and virtual reality [53]. Considerably less work was dedicated to the systems engineering domain until recently. Basole et al. developed visual analytics tools to help users perform Complex Engineered System (CES) design analysis tasks helping stakeholders with visualizations of complex design models [7] and understand Failure Mode Effect Analysis (FMEA) data by modelling and visualizing it as a network [6]. Sedlmair et al. have developed multiple tools leveraging visual analytics in the electronic engineering domain for vehicle development and testing. One such tool is Cardiogram, that helps engineers debug millions of recorded messages from safety-critical in-car communication networks and ensure that they are error-free [47]. RelEx helps engineers specify and optimize traffic patterns for in-car communication networks [45]. MostVis facilitates exploration of MOST function catalogs which was otherwise infeasible using paper and existing database interfaces [44]. Another tool was a dual-view visualization system that helped diagnostics of

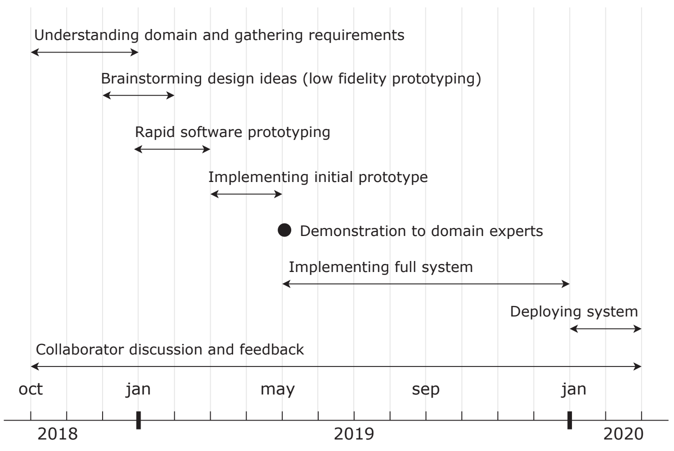


Fig. 3. The high-level phases and milestones of the design process.

in-car communication networks [46].

Functional Safety is a critical operation in automotive systems engineering whose methodologies [15, 30, 31] and applications [12, 18, 29, 56] have been extensively researched in literature. Several processes in systems engineering such as design reviews are often still conducted using unstructured documents (e.g., Excel sheets [37]) which is non-standard, inconsistent, and can cause miscommunication [22]. Part of the problem was solved by the introduction of model-based systems engineering (MBSE) approaches that encourage information exchange using a common language [41] (e.g., SysML [24]). There exist several tools now, both commercial [1, 3] as well as open-source [23, 51] that support commonly performed tasks in systems engineering. However, these advances have resulted in more complex models and user tasks.

In general, these existing tools focus their user experience and functionality on the authoring of the information. These are valuable for creating the information, but lack user interface affordances for communicating and exploring the data for various tasks described in subsequent sections. Moreover, these applications cannot be customized to support newer or more advanced types of tasks. For example, comparing multiple projects or visualizing the ASIL trace and decomposition cannot be easily accomplished in these tools. The potential application of viewing functional safety data in a visual analysis tool is under-explored, motivating the need for this work.

4 Design Process and Domain Requirements

We adopted a user-centered design methodology and conducted a series of design activities to learn about Functional Safety, the requirements and tasks of the domain experts, and distill design goals that ultimately ground SafetyLens (see Figure 3). This process included formative evaluation, design sessions, and iterative prototype development.

First, we conducted interviews with functional safety engineers, analysts, and supervisors to understand the domain requirements and user tasks. We conducted these interviews remotely (using teleconference applications with video and screensharing capabilities) over the course of three months, with one session per week. Generally, the sessions lasted 1 hour, with 1-4 domain experts on the calls. These sessions consisted of semi-structured interviews that started with questions to learn about their tasks, roles, and responsibilities. We gained an understanding of the current design ecosystem of tools, tasks, and datasets. We learned about the questions they have to answer as part of their daily activities, and specifically how those relate to the different datasets, databases, and other sources of information (e.g., manuals, regulations, etc.). We learned about the current workflow and identified 25 commonly performed tasks.

Second, we began design exercises to explore potential visualization and interaction techniques. We sketched ideas on paper and shared them with our collaborators during subsequent sessions. These sketches included potential visualization techniques (e.g., tree maps, force directed networks, hive plots), UI widgets (e.g., sliders, dropdowns, buttons), UI layouts (e.g., panels, grids), workflows, and interactions. These were low-fidelity designs sketched quickly but detailed enough to catch errors and slips that could surface later. During these sessions, we collaboratively brainstormed on the pros and cons of each design which eventually resulted in multiple changes and refinements. We also realized these designs by developing rapid software prototypes with a dual purpose of exploring potential technologies for the tool (such as software libraries and packages) and evaluating the feasibility of the designs. We deployed these prototypes for the domain experts to interact and evaluate them based on their usability and visual look and feel, with small subsets of their data integrated into the prototypes. This helped discard less-useful designs, fine-tune our initial set of tasks, and derive 6 design goals.

Third, we designed and developed an initial prototype of SafetyLens and conducted an in-person demo and feedback session. This consisted of two half-day sessions with domain experts comprising functional safety analysts, project managers, and personnel from application-specific teams (e.g., chassis, power-train) who were split into two groups. The first session included 15 participants (7 in-person, 8 online), and the second one included 20 participants (6 in-person, 14 online). Each session consisted of a presentation of the design goals and supported user tasks, followed by a demonstration of the prototype. One researcher presented the prototype and led the discussion, while a second took notes on participants' feedback. The participants engaged in thoughtful discussions about the demonstrated usage scenarios. They had access to sticky notes to keep track of comments throughout the presentation (with remote participants being able to use chat and other messaging services to give their feedback). Also, screen-shots were given to each participant to annotate with comments, changes, or other visual cues about how to improve the interface and interaction design.

Finally, a whiteboard was used by the in-person participants to illustrate how attributes from different databases can be integrated into SafetyLens. These discussions led to refinements to existing features, as well as new tasks that the tool could support. The user tasks and design goals presented below are those that resulted from this second round of user feedback.

4.1 User Tasks

User tasks which SafetyLens should support are:

- Discover patterns within projects; e.g., a project has elements that are mostly assigned an ASIL=D (high risk).
- Find missing (otherwise *must-have*) links between nodes; e.g., $\{MB \rightarrow HzE\}$, $\{HzE \rightarrow SG\}$, $\{SG \rightarrow FSR\}$, $\{FSR \rightarrow TSR\}$.
- Look up and analyze a node's end-to-end traceability (i.e., the extent to which functional safety elements have been defined); e.g. $\{MB \rightarrow HzE \rightarrow SG \rightarrow FSR \rightarrow TSR\}$.
- Compare ASILs by analyzing their decomposition into their *S-E-C* (Severity, Exposure, and Controllability) constituents.
- Find common nodes and links among projects; (e.g., *Hazards (Haz)* that exist in multiple projects).
- Discover anomalies across projects; (e.g., the same *Hazardous Event (HzE)* is assigned a different ASIL across projects).
- Compare key metrics across projects; (e.g., counts and distribution of nodes and links within projects).
- Get an overview of current and past projects to monitor status.

4.2 Design Goals

From the design exercises and requirements gathering activities described above we identified the following design goals.

DG1: Facilitate Exploration of a Project User tasks such as "Find missing links", "Lookup and analyze a node's end-to-end traceability" are network exploration tasks. Hence, we modeled Functional Safety data as a network and derived this design goal to support tasks like fetching node details on hover, finding adjacent nodes, and finding paths from one node to another based on the taxonomies by Lee et al. [34] and Pretorius et al. [40].

DG2: Facilitate Comparison among Projects Teams within functional safety can be system-specific and may not always be aware of the day to day progress made by other teams. This may result in duplicate work (e.g., a team may re-implement an artifact from scratch instead of re-using the one already implemented by another team, or for another vehicle). A core goal for SafetyLens, thus, was to provide a unified interface where users can explore and compare multiple projects to find shared (common) nodes, links, or even sub-graphs (combination of nodes and links). This unified interface can foster better collaboration among teams while also saving time and resources for the organization.

DG3: Discover Patterns and Anomalies The domain experts we spoke to make use of existing commercial and open source tools. These tools support basic exploration and comparison of Functional Safety projects but may fall short when the scale and complexity of data increases. Thus, SafetyLens should support discovering interesting patterns within and across projects.

DG4: Facilitate Traceability and Decomposition of ASILs An important task for domain experts is to trace the ASIL from one node to another (e.g., $\{MB \rightarrow HzE \rightarrow SG \rightarrow FSR \rightarrow TSR\}$). Since ASILs determine the extent of safety mechanisms for elements, any discrepancy such as an element assigned an ASIL=A instead of ASIL=D is an important concern. To diagnose the problem, users should be able to decompose the ASIL into its *Severity (S), Exposure (E), and Controllability (C)*. Thus, it is an important design goal for SafetyLens to unify tasks allowing users to efficiently detect, diagnose, and fix ASIL assignment issues.

DG5: Support different User Groups Functional Safety consists of engineers, analysts, and managers all of whom perform analysis and decision-making tasks at different levels. While engineers and analysts are concerned with lower element-level tasks within projects, managers are concerned with higher project-level tasks. SafetyLens should support both along the functional safety organization hierarchy.

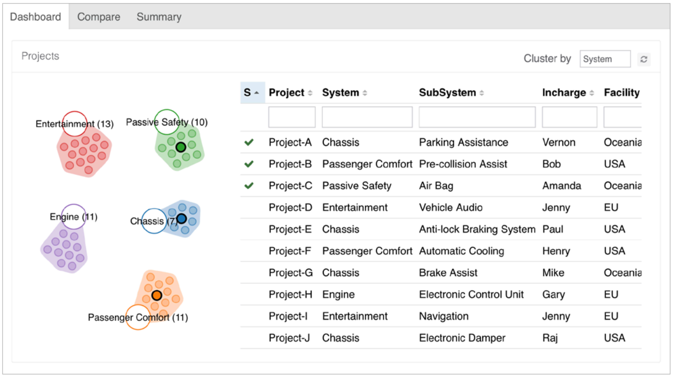
DG6: Provide a Summary of Key Metrics The user tasks suggested that key metrics should be readily available (e.g., total number of nodes, total number of nodes with ASIL=D, etc).

In addition to these design goals, we had to consider other factors. Since Functional Safety datasets can get large, we had to design for scalability and performance. We utilized a graph database to persist the data and execute queries to offload computation on the browser. Further, we had to ensure our tool is easy-to-deploy and integrates well with existing systems and workflows (e.g., the use of linked spreadsheets that are local to groups and users as well as shared databases). Finally, our users are from the automotive domain and not visualization practitioners, hence SafetyLens needs a simple user experience.

4.3 Design Considerations

Since we modeled Functional Safety data as a network we considered several network visualization techniques. We implemented rapid prototypes of standard visualization techniques such as hive-plots, dot-matrix plots, and parallel coordinate charts. These had a number of shortcomings. First, due to the number of links between nodes, edge-crossings can get messy. Second, positioning the nodes linearly in hive-plots made it difficult to discover patterns (e.g., distribution of ASILs). The dot-matrix plot enabled discovering patterns but give an impression that the {x,y} spatial coordinates of each node are of importance which was not the case. Third, parallel coordinates charts failed because the nodes had connections with more than two types of nodes.

We implemented a force-directed network with a multi-body force algorithm generating node positions at random. However, similar nodes (e.g., *Elements* with the same *Type*) were scattered throughout the canvas making pattern discovery difficult. Inspired by Saket et al's study that found node-link-group diagrams more "enjoyable" than node-link diagrams [43], we augmented our visualization with circle packing (to group similar nodes together within a cluster), and collision detection (to prevent overlaps between clusters and nodes) algorithms to achieve our goal. To facilitate interactive exploration and "fluidity" [20], we provide affordances to drag the clusters within the canvas, brush and link between projects by a lasso operation, and a control panel with search and filter direct manipulation widgets.



S	Project	System	SubSystem	Incharge	Facility
✓	Project-A	Chassis	Parking Assistance	Vernon	Oceania
✓	Project-B	Passenger Comfort	Pre-collision Assist	Bob	USA
✓	Project-C	Passive Safety	Air Bag	Amanda	Oceania
	Project-D	Entertainment	Vehicle Audio	Jenny	EU
	Project-E	Chassis	Anti-lock Braking System	Paul	USA
	Project-F	Passenger Comfort	Automatic Cooling	Henry	USA
	Project-G	Chassis	Brake Assist	Mike	Oceania
	Project-H	Engine	Electronic Control Unit	Gary	EU
	Project-I	Entertainment	Navigation	Jenny	EU
	Project-J	Chassis	Electronic Damper	Raj	USA

Fig. 4. The **Dashboard View** shows available projects. The projects in the node-link group visualization on the left are clustered as well as colored based on their "System" (configured from the *Cluster by* dropdown to the top-right of the Table View.

To support comparison between projects, we show multiple node-link-group diagrams aligned spatially. One idea was to have a single visualization with all projects on a shared canvas. However, this had a major shortcoming. Since projects comprise shared nodes with attributes that are specific to a project (e.g., Node A in Project A has an ASIL=B but ASIL=D in Project B), it was important to keep the nodes and hence the projects separate. Our users are familiar with tabular representations of data (e.g., MS Excel). Keeping their visualization literacy in mind, we complemented each node-link-diagram with a data table. To facilitate comparison between projects, we considered juxtaposing the node-link-group diagram and the datatable within each project panel and vertically stacking the project panels themselves. Since comparing vertically is challenging, we flipped this approach and provided drag-and-drop affordances to reorder the project panels.

5 SafetyLens

This section describes the main views and functionality of SafetyLens.

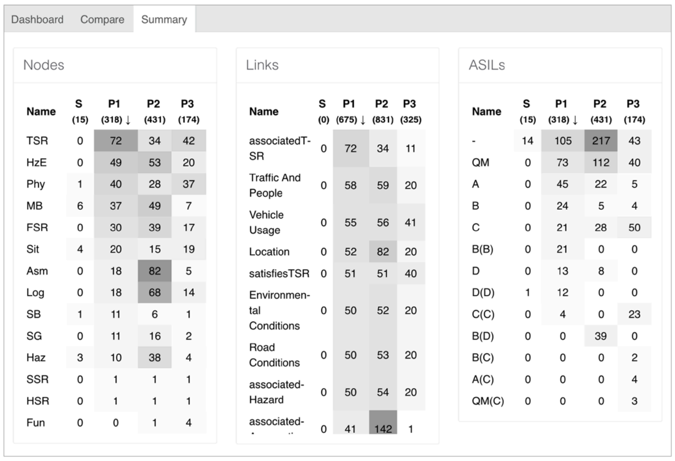
5.1 User Interface

SafetyLens has three primary views: 1) Dashboard View, 2) Summary View, and 3) Compare View.

Dashboard View, shown in Figure 4, is the landing page and provides an overview of the Functional Safety ecosystem. It consists of a clustered node visualization with each node representing a project. We complemented the visualization with a table to provide a familiar interface to users who are more comfortable with data in spreadsheets. The table consists of project-level information such as {Name, Department, Project In-Charge, and Location}. SafetyLens supports multiple starting points for users to analyze functional safety data. By providing access to all projects within the organization, SafetyLens facilitates collaboration **(DG2)** among domain experts with different roles **(DG5)**.

Summary View, shown in Figure 5, provides a summary of the projects selected in the Dashboard View **(DG6)**. There are juxtaposed heatmap visualizations for *Node Type*, *Link Relation*, and *ASIL* respectively positioned next to each other. For each attribute table, projects are along the column axis and the corresponding attribute values are along the row axis. An additional column titled "S" is added to show the number of entities (nodes and links) that are shared among these projects. The cells show the per-project entity counts for the corresponding attribute, colored using a continuous color scale (*white-to-gray*) to help the user discover patterns within as well as across projects **(DG3)**.

Compare View, shown in Figure 1, is the main view of the SafetyLens interface that allows users to simultaneously perform exploratory analysis of an individual project as well as a comparative analysis across multiple projects. It has four subviews (alphanumeric list enumerations match with those in Figure 1):



Nodes

Name	S (15)	P1 (318) ↓	P2 (431)	P3 (174)
TSR	0	72	34	42
HzE	0	49	53	20
Phy	1	40	28	37
MB	6	37	49	7
FSR	0	30	39	17
Sit	4	20	15	19
Asm	0	18	82	5
Log	0	18	68	14
SB	1	11	6	1
SG	0	11	16	2
Haz	3	10	38	4
SSR	0	1	1	1
HSR	0	1	1	1
Fun	0	0	1	4

Links

Name	S (0)	P1 (675) ↓	P2 (831)	P3 (325)
associatedT-SR	0	72	34	11
Traffic And People	0	58	59	20
Vehicle Usage	0	55	56	41
Location	0	52	82	20
satisfiesTSR	0	51	51	40
Environmen-tal Conditions	0	50	52	20
Road Conditions	0	50	53	20
associated-Hazard	0	50	54	20
associated-	0	41	142	1

ASILs

Name	S (15)	P1 (318) ↓	P2 (431)	P3 (174)
-	14	105	217	43
QM	0	73	112	40
A	0	45	22	5
B	0	24	5	4
C	0	21	28	50
B(B)	0	21	0	0
D	0	13	8	0
D(D)	1	12	0	0
C(C)	0	4	0	23
B(D)	0	0	39	0
B(C)	0	0	0	2
A(C)	0	0	0	4
QM(C)	0	0	0	3

Fig. 5. The **Summary View** shows three projects **P1**, **P2**, and **P3** and the distribution of nodes by their *Types*, links by their *Relations*, and *ASILs* by their values. The column ***S*** is a count of shared entities (nodes and links) between the projects. Project **P1** has **318** nodes and **675** links. The projects share **15** nodes and **0** links among them.

(A) **PROJECT PANEL**. Each imported project is rendered as a panel that shows the project title and utilities to (i) reset the panel state, (ii) export the panel as an image, (iii) toggle full-screen mode, and (iv) horizontally reorder the panel using drag and drop.
VISUALIZATION CANVAS shows a node-link group visualization. Each node (small circle) is an *Element* of a functional safety project. These nodes are clustered together based on attributes (e.g., *Type*). The largest circles represent the group nodes and are labeled with the *Type* and the number of nodes that are part of it. The boundary marking the extent of the groups (convex hull) is highlighted. The nodes are positioned in each others' vicinity using an implementation of the circle packing algorithm. The node size and color can be mapped to attributes such as {ASIL, Type, Degree (number of edges to a node)} which in the presence of multiple nodes across multiple projects will create visual clusters leading to discoveries of patterns and anomalies **(DG3)**. By default, the groups are positioned relative to each other based on the multi-body force and collision detection algorithms. However, these groups are fluid and can be re-positioned by dragging them around. Further, a control panel toggle button enables the users to position and fix these clusters to similar relative locations across all projects to aid comparison (e.g., the *SB* cluster can be positioned to the top-right of the canvas across all projects). Section 5.2 describes all the interactions supported by this visualization.
TAB LAYOUT has three options: Nodes, Links, and Trace.
  (1) **NODES TAB** shows a datatable with the nodes selected by the user with their {*Type, ASIL, Name, ID*}.
  (2) **LINKS TAB** shows a datatable with the Links data with their {*Source, Relation and Target*}.
  (3) **TRACE TAB** (shown in Figure 11) allows the user to find and visualize if a path exists between two nodes as well as trace their ASILs **(DG4)** (e.g., tracing the ASIL from a *System Behavior* to a *Technical Software Requirement*). The user can set a node as *Source* and another as *Destination*. SafetyLens first checks if a path exists between the two nodes and overlays it onto the node-link group visualization. It also returns a linearized node-link diagram showing the entire route from source to destination. Below the node-link diagram is a heatmap showing the S-E-C (Severity-Exposure-Controllability) break up of the ASILs.
Since the visualization canvas and the tab layout are vertically stacked within a project panel, we positioned each panel side by side. This way, SafetyLens would facilitate exploratory analyses within and comparative analyses across projects **(DG1, DG2)**.

(B) **SHARED VIEW** shows the nodes and links that are common to / shared by all loaded projects **(DG2)**. These are computed by performing a set intersection operation across project graphs based on unique node identifiers. We show it in a separate view instead of overlaying or highlighting in the same view to make it easier for domain experts to begin their analysis.

(C) **DETAILS VIEW** shows all attributes of a node as well as a table containing all nodes that are connected to this node.

(D) **CONTROL PANEL** has three tabs: (i) Nodes, (ii) Links, and (iii) Config with various operations that can be performed (e.g., lookups, encodings, layout, preferences for tooltips, and more).

5.2 Interactions

The primary design goals for the user interactions are to facilitate brushing and linking between views while maintaining a simple and usable interface. These interaction include:

(A) **HOVER** The visualization nodes and the datatable rows can be hovered to highlight neighbours as well as show more information in the Details View.
(B) **CLICK** The visualization nodes and the datatable rows can be (left) clicked to toggle their selections. In addition, right-clicking on a visualization node opens a context-menu that supports operations such as *Select*, *Set as Source/Destination*, and others.
(C) **DRAG** The group nodes can be dragged to move the clusters within the visualization canvas. Similarly, the project panels themselves can be reordered to aid comparison.
(D) **SELECTION** The visualization canvas can be used to draw free-form shapes via a lasso operation to select nodes and links.
(E) **SEARCH** The Search input field in the Control Panel can be used to "lookup" and "select" nodes by their names.
(F) **FILTER** The Control Panel has UI controls (e.g., radio buttons, checkboxes, dropdowns, interactive legends) that allow the user to "filter" and "select" the nodes and links in the visualization.
(G) **SORT** The column headers of datatables in the Tab layout can be used to sort the selected nodes and links.

5.3 Implementation

SafetyLens is implemented as a web application using Flask (a Python-based microframework) [4], AngularJS [25], and D3.js [38] . The Functional Safety data is stored in OrientDB [16].

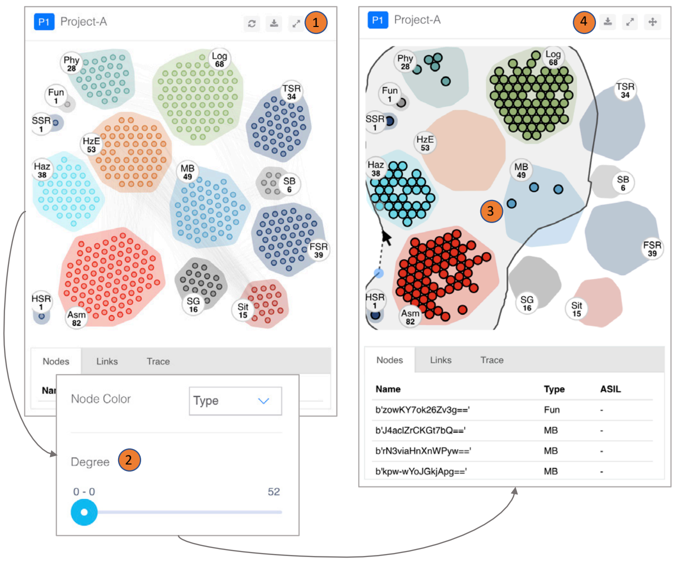
6 USAGE SCENARIOS

We illustrate how SafetyLens can help domain experts visualize and interact with Functional Safety datasets. Consider three usage scenarios comprising several subtasks as performed by three hypothetical users - Chris (analyst), Parker (Chris's supervisor), and Kyle (analyst) respectively. These usage scenarios were developed in collaboration with the domain experts from our interview studies to ensure domain relevance. Due to confidentiality concerns, project, node, and link names have been obfuscated. These scenarios are also illustrated in the accompanying video.

6.1 General Exploratory Tasks

Chris and their colleagues have been actively working on Project-A (the "Adaptive Cruise Control" System) for a week. They have (i) identified several *Malfunctioning Behaviors* and the potential *Hazards* that could arise, (ii) assigned a risk level (ASIL) to these hazards and (iii) defined *Safety Goals* and requirements (*Functional Safety Requirements*) to minimize these risks. Chris wishes to perform a status check to determine if the risks associated with the project components have been reduced to acceptable levels. This includes the following steps (or tasks).

Find and address Orphan Nodes It is important to ensure that the defined nodes are connected to other nodes (that is, are not orphan). By being aware of these orphan nodes, Chris can ensure that these are not implemented from scratch but instead reused from the same node's implementation in another project.



Name	Type	ASIL
b'zowKY7ok26Zv3g=='	Fun	-
b'J4aclZrCKGt7bQ=='	MB	-
b'rN3viaHnXnWPyw=='	MB	-
b'kpw-wYoJGkjApg=='	MB	-

Fig. 6. **Report Orphan Nodes and their Types**. **(1)** Default State where nodes are colored by their *Type*, **(2)** Drag the degree slider to zero, **(3)** Select the remaining nodes with a lasso operation; these are highlighted with a black stroke and also added to the nodes datatable below, and **(4)** Export the view as an image.

To accomplish this task, Chris first opens the Control Panel and drags the Degree slider to zero. This filters out the nodes that have at least one connection. Using a lasso operation, they select these nodes and export a snapshot of the visualization for future reference. Figure 6 summarizes this task.

Find and report nodes with an unassigned ASIL. After addressing orphan nodes, Chris shifts their focus to assigning an ASIL to each node. The ASIL determines the quality and quantity of safety measures that are undertaken to minimize associated risks. During this assignment process, it is possible that a few nodes are either missed or skipped for later. In both cases, the node is left without a valid ASIL.

To verify this for *Hazardous Events*, Chris opens the Control Panel and hides all node types except *Hazardous Event*. From the ASIL legend, they select the nodes with an ASIL "-" (unassigned). Several nodes (gray) that do not have an ASIL are selected and added to the datatable below. Chris shares the selected nodes with their team to discuss and plan next steps. Chris's workflow is summarized in Figure 7.

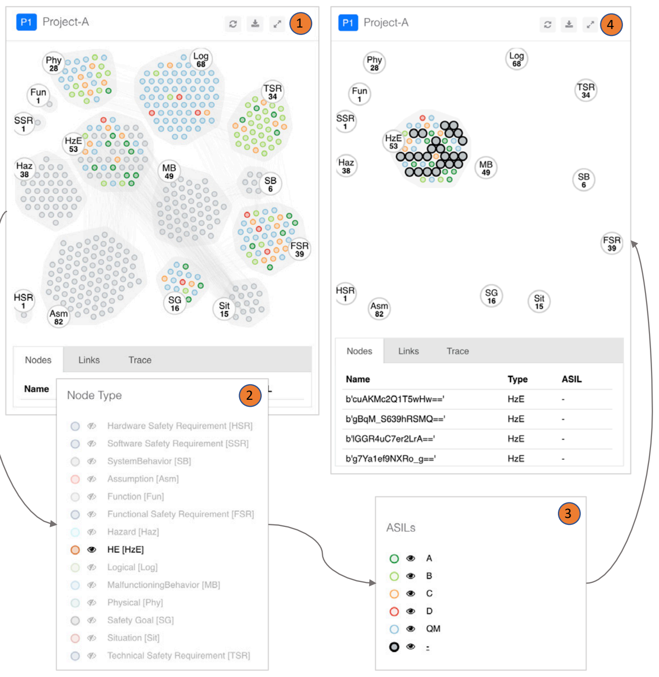
Find Missing Links. After analyzing orphan and ASIL-unassigned nodes, Chris funnels their attention to analyze specific connections within the Functional Safety network to ensure that:

1. Each *Malfunctioning Behavior (MB)* should have a *Hazardous Event (HzE)* identified for it.
2. Each *HzE* should have a *Safety Goal (SG)* assigned to it.
3. Each *SG* should define a *Functional Safety Requirement (FSR)*.
4. Each *FSR* should define a *Technical Safety Requirement (TSR)*.

A domain expert's role is to address such missing connections at the earliest. The links in the dataset comprise the *Relation* attribute that determines the type of connection between two nodes, e.g. *associatedHE* connects a *Malfunctioning Behavior* with a *Hazardous Event*. A unique Relation is defined for a connection between nodes of different Types. To find these missing connections, Chris opens the Control Panel and switches to the Links tab. As illustrated in Figure 8, they select the corresponding Relations from the legend that are highlighted in the visualization. Chris observes that:

1. ***associatedHE***: *HzEs* for more than half of *MBs* are not identified.
2. ***associatedSafetyGoal***: A few *HzEs* do not have a *SG* assigned.
3. ***associatedFSR***: All *SGs* have at least one *FSR* defined.
4. ***associatedTSR***: Very few *FSRs* have no *TSRs*.

From the above usage scenario, Chris found several nodes without an ASIL, several nodes with zero as well as missing links. They conclude that even though their Project-A is under active development, there are gaps in their process that are hampering their efficiency. They take these results to their team to take required actions.



Name	Type	ASIL
b'cuAKMc2Q1T5wHw=='	HzE	-
b'gBqM_S639hRSMQ=='	HzE	-
b'lGGR4uC7er2LrA=='	HzE	-
b'g7Ya1ef9NXRo_g=='	HzE	-

Fig. 7. **Find elements that are not yet assigned an ASIL. (1)** Default State where nodes are colored by their *ASIL*, **(2)** Deselect all Node Types except *Hazardous Events (HzE)*, **(3)** Select all nodes with unassigned ASILs, and **(4)** The selected ASILs are highlighted with a **black** border.

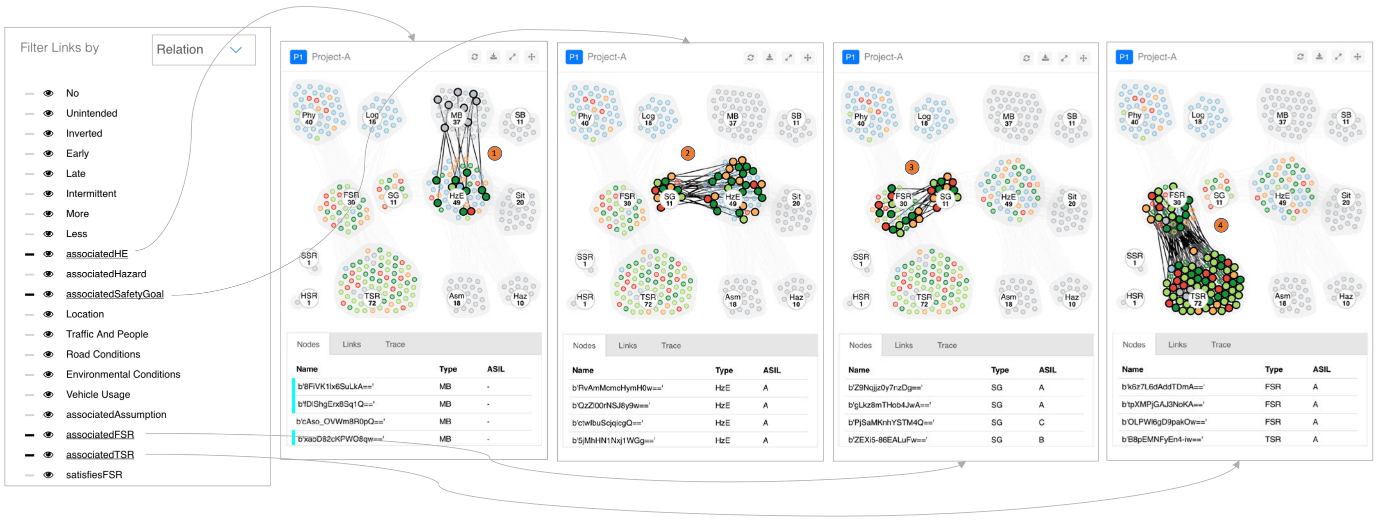
6.2 Trace ASIL

Another aspect of an analyst's role is to analyze the ASIL of a node in comparison with the ASILs of its connections (neighbor, neighbor's neighbor, and so on). Consider a *Hazardous Event (HzE)* that is assigned the highest risk classification ASIL=D. It requires *Safety Goals (SG)* that can sufficiently minimize the associated risk. By default, $\{SGs\}$ inherit the ASIL of the $\{HzE\}$ ($\{FSRs\}$ inherit the ASIL of the $\{SG\}$, and $\{TSRs\}$ inherit the ASIL of the $\{FSR\}$) but the analyst may choose to override it to achieve acceptable risk levels, making visual analysis of the ASIL trace an important task.

Kyle's role is to assess risks associated with *HzEs* and qualitatively assign an ASIL to them. Their responsibilities also include analyzing the extent to which Functional Safety is achieved for a node, that is, finding paths between nodes that are not directly connected via a link. This way, a project's maturity ("Are there sufficient nodes and links implemented for this project?") as well as completeness ("Most *SBs* have a *TSR* defined now; the project is nearing its completion.") can be assessed. Kyle performs the following tasks:

Manually Trace paths. As illustrated in Figure 9, Kyle chooses a known *SB* for further analysis. (1) On hovering, they see several links connecting it with multiple *MBs*. (2) They select the *SB* and hover on one of the *MB* which highlights the links with *HzEs*. (3,4) They continue this process until they reach an *FSR*. (5) They find that the *FSR* does not have a link with an *TSR*. Since this is a requirement, this means Functional Safety is still incomplete for the *SB* in (1).

Find if a Path exists between two Nodes. In the previous task, Kyle traversed the network manually by following the nodes' connections. SafetyLens can also check if a path exists between non-adjacent nodes, that is, nodes that are not directly connected. Consider a scenario to check if a *(Technical Safety Requirement (TSR)* exists for a *System Behavior (SB)* as illustrated in Figure 11. Kyle imports Project-A into the Compare View and switches to the Trace Tab in the Project View. (1) They right click on the *SB* and set it as Source from the context menu. (2) They look up the *TSR* from the search field and set it as Destination. (3) SafetyLens calculates and finds that a path does exist between the two nodes. In this way, SafetyLens helps analysts find and



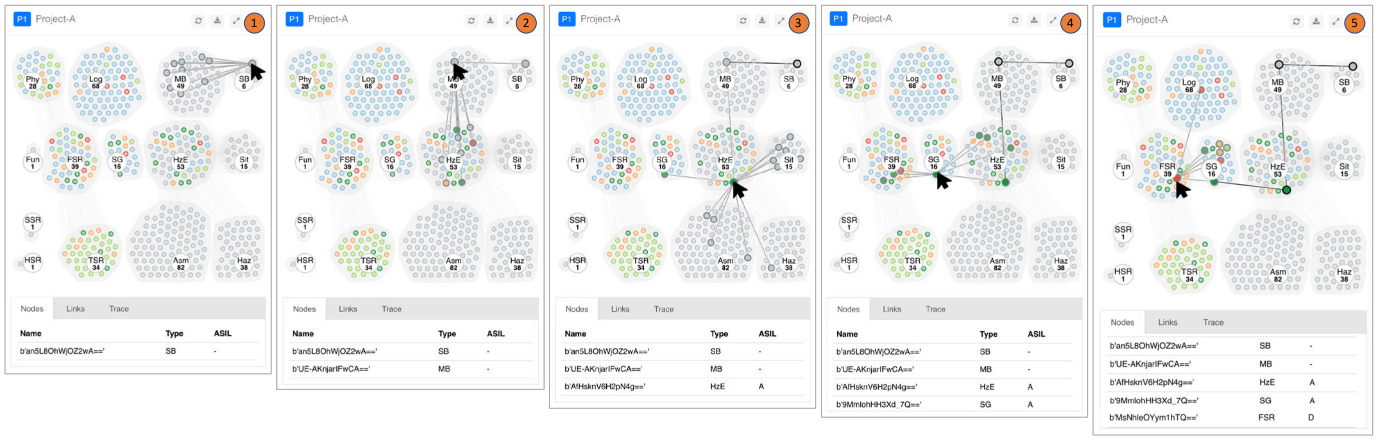
Name	Type	ASIL
b'8FiVK1Ix6SuLkA=='	MB	-
b'fDIShgErx8Sq1Q=='	MB	-
b'cAso_OVWm8R0pQ=='	MB	-
b'xaoD82cKPWO8qw=='	MB	-

Name	Type	ASIL
b'RvAmMcmcHymH0w=='	HzE	A
b'QzZI00rNSJ8y9w=='	HzE	A
b'ctwIbuScjqicgQ=='	HzE	A
b'5jMhHN1Nxj1WGg=='	HzE	A

Name	Type	ASIL
b'Z9Nqjjz0y7nzDg=='	SG	A
b'gLkz8mTHob4JwA=='	SG	A
b'PjSaMKnhYSTM4Q=='	SG	C
b'ZEXi5-86EALuFw=='	SG	B

Name	Type	ASIL
b'k6z7L6dAddTDmA=='	FSR	A
b'tpXMPjGAJ3NoKA=='	FSR	A
b'OLPWI6gD9pakOw=='	FSR	A
b'B8pEMNFyEn4-iw=='	TSR	A

Fig. 8. **Find Missing Links.** (1) Each *Malfunctioning Behavior (MB)* should have a *Hazardous Event (HzE)*. (2) Each *HzE* should have a *Safety Goal (SG)*. (3) Each *SG* should define a *Functional Safety Requirement (FSR*. (4) Each *FSR* should define a *Technical Safety Requirement (TSR)*.



Name	Type	ASIL
b'an5L8OhWjOZ2wA=='	SB	-

Name	Type	ASIL
b'an5L8OhWjOZ2wA=='	SB	-
b'UE-AKnjarIFwCA=='	MB	-

Name	Type	ASIL
b'an5L8OhWjOZ2wA=='	SB	-
b'UE-AKnjarIFwCA=='	MB	-
b'AfHsknV6H2pN4g=='	HzE	A

Name	Type	ASIL
b'an5L8OhWjOZ2wA=='	SB	-
b'UE-AKnjarIFwCA=='	MB	-
b'AfHsknV6H2pN4g=='	HzE	A
b'9MmlohHH3Xd_7Q=='	SG	A

b'an5L8OhWjOZ2wA=='	SB	-
b'UE-AKnjarIFwCA=='	MB	-
b'AfHsknV6H2pN4g=='	HzE	A
b'9MmlohHH3Xd_7Q=='	SG	A
b'MsNhleOYym1hTQ=='	FSR	D

Fig. 9. **Manually Trace paths.** (1-5): The user is manually tracing a path from a *System Behavior (SB)* up to a *Functional Safety Requirement (FSR)*. (5): The *FSR* does not have a corresponding *Technical Safety Requirement (TSR)*.

visualize paths between two nodes.

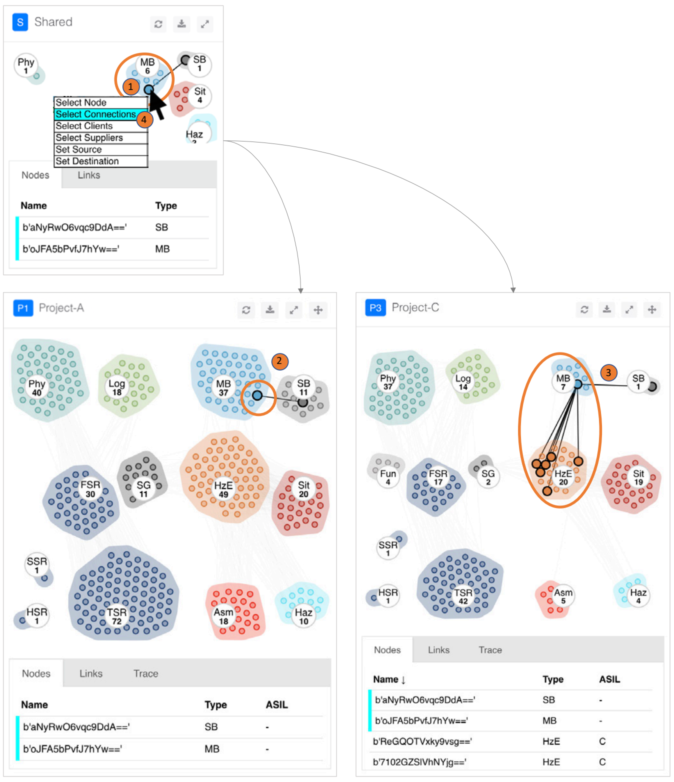
Compare ASILs along a Path. Figure 11 also illustrates an extension of the previous task to analyze the ASILs of intermediary nodes along the path. (4) The node-link (rectangles) diagram below reshapes the overlaid path in the visualization into a linear layout with the Source and Destination nodes at the ends. The rectangle's fill color is determined by the ASIL of the node. Kyle observes that the same ASIL C has propagated from the *Hazardous Event (HzE)* to the *Technical Safety Requirement (TSR)*. To verify if the ASILs are really consistent, they need to analyze their respective breakups into Severity (S), Exposure (E), and Controllability (C) values. (5) SafetyLens shows this decomposition of ASIL in the form of a heatmap. Kyle observes that while {N92, N136, N187, N209} have consistent ASILs, they do not have consistent Severity (S), Exposure (E), and Controllability (C). For example, the Controllability of N136 is C2 whereas it should have ideally been C3 (from N92). In this way, SafetyLens helps analysts trace ASILs within the functional safety network.

6.3 Find Common Elements Between Projects

Until now we have illustrated how SafetyLens helps domain experts explore and analyze one project at a time. SafetyLens also supports simultaneous comparative analysis among multiple projects. We illustrate this with another scenario.

In Section 6.1, we illustrated how SafetyLens helped Chris find and report *Malfunctioning Behaviors (MB)* that did not have a corresponding *Hazardous Event (HzE)*. With resources likely already allocated for this project, Chris showed the list to his supervisor, Parker. On seeing the list, Parker is able to recall a previous project Project-C where the same elements may have already been defined. They open SafetyLens but this time select two projects: Project-A and Project-C from the Dashboard view. An overview (distributions and counts) of these projects including shared nodes and links can be seen from the Summary View similar to that shown in Figure 5. Parker sees that there are common nodes and links among these projects and switches to the Compare View to find the specific ones.

As illustrated in Figure 10, they see a new Shared Nodes panel at the top left. SafetyLens, by applying set intersection between network nodes and links, has pre-computed the shared nodes and links between the two projects and presented in the same node-link group visualization. They hover on a *Malfunctioning Behavior (MB)* and find that it is in fact on Chris's list. (1) Hovering on this node in the shared view highlights the corresponding nodes in Project-A and Project-C along with their connections, if any. While there are rightly no links from the *MB* to any *HzE* in Project-A (2), there are several links to *HzEs* in Project-B (3). Parker right clicks the node of concern in Project-B and "Selects Connections" from the context menu (4). This selects all nodes that are connected to this node. Parker exports the list in Project-C and shares it with Chris who, instead of defining these connections from scratch, can just re-use them.



Shared — Nodes | Links

Name	Type
b'aNyRwO6vqc9DdA=='	SB
b'oJFA5bPvfJ7hYw=='	MB

Project-A — Nodes | Links | Trace

Name	Type	ASIL
b'aNyRwO6vqc9DdA=='	SB	-
b'oJFA5bPvfJ7hYw=='	MB	-

Project-C — Nodes | Links | Trace

Name ↓	Type	ASIL
b'aNyRwO6vqc9DdA=='	SB	-
b'oJFA5bPvfJ7hYw=='	MB	-
b'ReGQOTVxky9vsg=='	HzE	C
b'7102GZSIVhNYjg=='	HzE	C

Fig. 10. **Find common nodes among Projects**. (1) Hover on a node in the Shared View highlights the common nodes and links in other project views. (2) There is no link to *HzE* in Project-A whereas (3) Project-C has six links to *Hze*. (4) 'Select(ing) Connections' from the context menu selects the nodes for further analysis.

7 Discussion

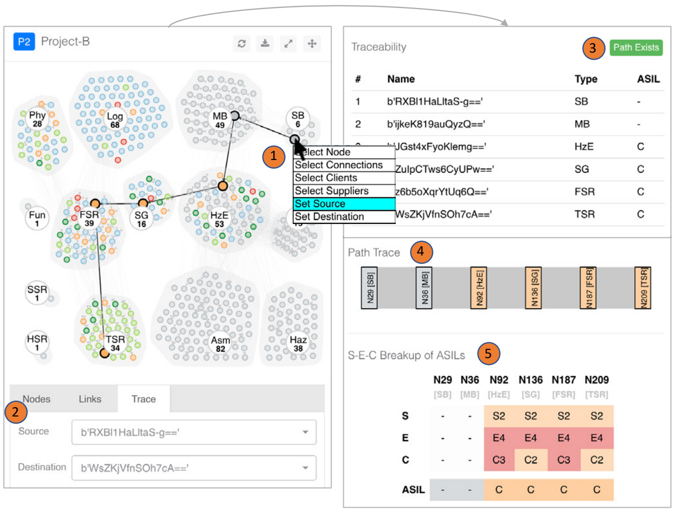
7.1 Feedback

To validate the design and functionality of SafetyLens, we gathered user feedback from the same domain experts with whom we conducted the initial feedback session and for whom the system was designed. This was performed remotely, with our collaborator at the company demonstrating the prototype to users with their data (obfuscated in this paper). The overall feedback was positive.

Users commented that creating an overview encompassing several projects is currently very time consuming, and once more data is imported in to SafetyLens, would be something that can make their tasks more effective. Users also commented that seeing their data in SafetyLens opened up new questions and tasks. For instance, one participant commented that *"this let's me see projects quickly, and spot problems. But sometimes things don't look right, can I edit right here?"* Engineers currently edit the knowledge base that contains these files, but we will consider adding editing functionality as demonstrated in [10] and [19] in the future.

Another participant asked *"how do I only see what's changed in the visualization from the last time I logged in? How do I know if this ASIL C has undergone revisions, e.g. was previously D?"* This opened up an entire aspect of visualizing temporal changes to data. This was fascinating as their database currently updates existing data with no provision to log the revision history. We will consider supporting this task of temporal evolution of ASIL in the future.

Few participants found aspects of the (node-link group) visualization challenging to follow initially and got more comfortable with use. For example, one participant asked, *"what do these (so many node) colors represent?"*. Another participant asked, *"what do the spatial positions of nodes mean?"*. This was expected considering our users' visualization literacy and will help us further improve the user experience.



Project-B — Nodes | Links | Trace

Source: b'RXBI1HaLltaS-g=='

Destination: b'WsZKjVfnSOh7cA=='

Traceability

#	Name	Type	ASIL
1	b'RXBI1HaLltaS-g=='	SB	-
2	b'ijkeK819auQyzQ=='	MB	-
	b'UJGst4xFyoKlemg=='	HzE	C
	ZuIpCTws6CyUPw=='	SG	C
	z6b5oXqrYtUq6Q=='	FSR	C
	WsZKjVfnSOh7cA=='	TSR	C

S-E-C Breakup of ASILs

	N29 [SB]	N36 [MB]	N92 [HzE]	N136 [SG]	N187 [FSR]	N209 [TSR]
S	-	-	S2	S2	S2	S2
E	-	-	E4	E4	E4	E4
C	-	-	C3	C2	C3	C2
ASIL	-	-	C	C	C	C

Fig. 11. Finding if a path exists and trace the ASIL along it. (1) Set a Source (*System Behavior (SB)*) using the context menu (or search box) (2) Set the Destination (*Technical Safety Requirement (TSR)*) by looking it up in the search box (or context menu) (3) Status of a path's existence along with its waypoints. (4) A horizontal node-link diagram which helps visualize the Trace, as well as the ASILs of nodes. (5) Heatmap to visualize the decomposition of ASIL ratings into *Severity (S) - Exposure (E) - Controllability (C)*.

7.2 Deployment

Given the positive feedback to date from pilot users, SafetyLens will be explored further. This process involves various data permissions, storage, and other infrastructure challenges. Once deployed, we plan to gather feedback from users.

7.3 Limitations

SafetyLens in its current state has a few limitations. It is read-only, in that it does not support authoring data yet. It reads in data from the graph database only and does not support other ways such as manual file uploads. While it does have an export functionality to save the application state as an image, it does not support online collaboration features such as annotations, commenting, and sharing. We have tested SafetyLens to facilitate simultaneous comparison of up to five projects comprising close to thousand nodes and links each. The performance impact of visualizing more projects is yet to be tested.

8 Conclusion

Automotive domain experts perform functional safety of the vehicles to minimize risks associated with hazards. These domain experts currently spend a significant amount of time gathering and analyzing data. We designed and developed SafetyLens, a visual data analysis tool to help them visualize and interact with Functional Safety datasets. SafetyLens is a tool that aims to assist Functional Safety analysis by showing relationships and patterns across projects. This paper explains our design process, how the user tasks and design goals guided SafetyLens's interface design and development, and provided usage scenarios to explain how the tool can be used to perform existing as well newer advanced analysis tasks.

References

[1] No Magic, 2014.

[2] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. The state-of-the-art of set visualization. In *Computer Graphics Forum*, volume 35, pages 234–260. Wiley Online Library, 2016.

[3] Ansys. Medini Analyze.

[4] Armin Ronacher. Flask.

[5] D. Auber. Tulip—a huge graph visualization framework. In *Graph drawing software*, pages 105–126. Springer, 2004.

[6] R. C. Basole, A. Qamar, B. Pal, M. Corral, M. Meinhart, and A. Narechania. Understanding failure mode effect analysis data using interactive

visual analytics. *IEEE computer graphics and applications*, 39(6):17–26, 2019.

[7] R. C. Basole, A. Qamar, H. Park, C. J. Paredis, and L. F. McGinnis. Visual analytics for early-phase complex engineered system design support. *IEEE computer graphics and applications*, 35(2):41–51, 2015.

[8] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *Third international AAAI conference on weblogs and social media*, 2009.

[9] V. Batagelj and A. Mrvar. Pajek-program for large network analysis. *Connections*, 21(2):47–57, 1998.

[10] T. Baudel. From information visualization to direct manipulation: Extending a generic visualization framework for the interactive editing of large datasets. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, page 67–76, New York, NY, USA, 2006. Association for Computing Machinery.

[11] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. In *Computer Graphics Forum*, volume 36, pages 133–159. Wiley Online Library, 2017.

[12] J. Birch, R. Rivett, I. Habli, B. Bradshaw, J. Botham, D. Higham, P. Jesty, H. Monkhouse, and R. Palin. Safety cases and their role in iso 26262 functional safety assessment. In *International Conference on Computer Safety, Reliability, and Security*, pages 154–165. Springer, 2013.

[13] S. P. Borgatti, M. G. Everett, and L. C. Freeman. Ucinet for windows: Software for social network analysis. *Harvard, MA: analytic technologies*, 2006, 2002.

[14] S. Brown. Overview of iec 61508. design of electrical/electronic/programmable electronic safety-related systems. *Computing & Control Engineering Journal*, 11(1):6–12, 2000.

[15] S. Burton, J. Likkei, P. Vembar, and M. Wolf. Automotive functional safety= safety+ security. In *Proceedings of the First International Conference on Security of Internet of Things*, pages 150–159, 2012.

[16] Callidus Software Inc. Orientdb.

[17] M. Candela, M. Di Bartolomeo, G. D. Battista, and C. Squarcella. Radian: Visual exploration of traceroutes. *IEEE Transactions on Visualization and Computer Graphics*, 24(7):2194–2208, July 2018.

[18] Y.-C. Chang, L.-R. Huang, H.-C. Liu, C.-J. Yang, and C.-T. Chiu. Assessing automotive functional safety microprocessor with iso 26262 hardware requirements. In *Technical papers of 2014 international symposium on VLSI design, automation and test*, pages 1–4. IEEE, 2014.

[19] C. Eichner, S. Gladisch, H. Schumann, and C. Tominski. Direct visual editing of node attributes in graphs. *Informatics*, 3(4):17, Oct 2016.

[20] N. Elmqvist, A. V. Moere, H.-C. Jetter, D. Cernea, H. Reiterer, and T. Jankun-Kelly. Fluid interaction for information visualization. *Information Visualization*, 10(4):327–340, 2011.

[21] F. Fischer, J. Fuchs, P.-A. Vervier, F. Mansmann, and O. Thonnard. Vistracer: a visual analytics tool to investigate routing anomalies in traceroutes. In *Proceedings of the ninth international symposium on visualization for cyber security*, pages 80–87, 2012.

[22] G. H. Fisher. Model-based systems engineering of automotive systems. In *17th DASC. AIAA/IEEE/SAE. Digital Avionics Systems Conference. Proceedings (Cat. No.98CH36267)*, volume 1, pages B15/1–B15/7 vol.1, Oct 1998.

[23] T. E. Foundation. Papyrus.

[24] S. Friedenthal, A. Moore, and R. Steiner. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[25] Google Inc. AngularJS.

[26] R. Gove. V3spa: A visual analysis, exploration, and diffing tool for selinux and seandroid security policies. In *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8, 2016.

[27] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[28] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, 6(1):24–43, 2000.

[29] Q. V. E. Hommes. Review and assessment of the iso 26262 draft road vehicle-functional safety. Technical report, SAE Technical Paper, 2012.

[30] A. Ismail and W. Jung. Research trends in automotive functional safety. In *2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)*, pages 1–4. IEEE, 2013.

[31] ISO. Road vehicles – Functional safety, 2011.

[32] P. Jordan. Standard iec 62304-medical device software-software lifecycle processes. 2006.

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

[34] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, page 1–5, New York, NY, USA, 2006. Association for Computing Machinery.

[35] W.-S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie. Fault tree analysis, methods, and applications: A review. *IEEE transactions on reliability*, 34(3):194–203, 1985.

[36] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister. Upset: visualization of intersecting sets. *IEEE transactions on visualization and computer graphics*, 20(12):1983–1992, 2014.

[37] Microsoft Corporation. Microsoft excel.

[38] Mike Bostock. D3: Data-Driven Documents.

[39] J. O'Madadhain, D. Fisher, S. White, and Y. Boey. The jung (java universal network/graph) framework. *University of California, Irvine, California*, 2003.

[40] J. Pretorius, H. C. Purchase, and J. T. Stasko. Tasks for multivariate network analysis. In *Multivariate Network Visualization*, pages 77–95. Springer, 2014.

[41] A. L. Ramos, J. V. Ferreira, and J. Barceló. Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):101–111, Jan 2012.

[42] R. Sadana, T. Major, A. Dove, and J. Stasko. Onset: A visualization technique for large-scale binary set data. *IEEE transactions on visualization and computer graphics*, 20(12):1993–2002, 2014.

[43] B. Saket, C. Scheidegger, and S. Kobourov. Comparing node-link and node-link-group visualizations from an enjoyment perspective. In *Computer Graphics Forum*, volume 35, pages 41–50. Wiley Online Library, 2016.

[44] M. Sedlmair, C. Bernhold, D. Herrscher, S. Boring, and A. Butz. Mostvis: An interactive visualization supporting automotive engineers in most catalog exploration. In *2009 13th International Conference Information Visualisation*, pages 173–182, July 2009.

[45] M. Sedlmair, A. Frank, T. Munzner, and A. Butz. Relex: Visualization for actively changing overlay network specifications. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2729–2738, Dec 2012.

[46] M. Sedlmair, W. Hintermaier, K. Stocker, T. Büring, and A. Butz. A dual-view visualization of in-car communication processes. In *2008 12th International Conference Information Visualisation*, pages 157–162, July 2008.

[47] M. Sedlmair, P. Isenberg, D. Baur, M. Mauerer, C. Pigorsch, and A. Butz. Cardiogram: visual analytics for automotive engineers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1727–1736, 2011.

[48] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE transactions on visualization and computer graphics*, 12(5):733–740, 2006.

[49] R. Singh. International standard iso/iec 12207 software life cycle processes. *Software Process: Improvement and Practice*, 2(1):35–50, 1996.

[50] M. E. Smoot, K. Ono, J. Ruscheinski, P.-L. Wang, and T. Ideker. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–432, 2011.

[51] O. Source. Sysml architect.

[52] D. H. Stamatis. *Failure mode and effect analysis: FMEA from theory to execution*. Quality Press, 2003.

[53] J. A. Stevens. Visualization of complex automotive data: A tutorial. *IEEE computer graphics and applications*, 27(6):80–86, 2007.

[54] F. Van Ham and A. Perer. "search, show context, expand on demand": supporting large graph exploration with degree-of-interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):953–960, 2009.

[55] 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: state-of-the-art and future research challenges. In *Computer graphics forum*, volume 30, pages 1719–1749. Wiley Online Library, 2011.

[56] D. Ward, I. Ibarra, and A. Ruddle. Threat analysis and risk assessment in automotive cyber security. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, 6(2013-01-1415):507–513, 2013.

[57] J. Zhao, M. Sun, F. Chen, and P. Chiu. Missbin: Visual analysis of missing links in bipartite networks. In *2019 IEEE Visualization Conference (VIS)*, pages 71–75, 2019.