Department: Applications
Editor: Mike Potel, potel@wildcrest.com

Understanding Failure Mode Effect Analysis Data Using Interactive Visual Analytics

Rahul C. Basole
Georgia Institute of Technology

Ahsan Qamar
Ford Motor Company

Biswajyoti Pal
Georgia Institute of Technology

Michael Corral
Ford Motor Company

Matthew Meinhart
Ford Motor Company

Arpit Narechania
Georgia Institute of Technology

***Abstract*—Providing actionable insights through interactive visual analytics is essential to effective decision making. Yet, many complex systems engineering (SE) domains still lack such tools. Design reviews are often still based on static snapshots of data, without any dynamic interaction, data curation, and view creation capabilities to answer salient analysis questions. In this study, we report on a tool called DataHawk that helps answer common questions associated with one prominent SE context, namely failure mode and effect analysis (FMEA). The tool provides powerful exploration capabilities that enable system engineers, designers, and managers to probe FMEA data from multiple starting points, build questions dynamically, and find triangulated answers using multiple views rapidly. Field results are illustrated through a usage scenario from the automotive industry and show that the tool demonstrates the needed versatility, scalability, and effectiveness for real-world engineering data.**

Digital Object Identifier 10.1109/MCG.2019.2944230
Date of current version 1 November 2019.

Many engineered systems, such as modern automobiles and airplanes, are highly complex systems that require consideration of an increasing number of overlapping and potentially conflicting stakeholder concerns. As modelers from multiple

Published by the IEEE Computer Society

design teams address these diverse concerns, many model interdependencies can emerge, both explicit and implicit as well as known and not yet realized.[1] Key design decisions are then ideally made based on comprehensive engineering knowledge available in multiple (yet, often generally decoupled) data repositories.

However, contemporary design review approaches utilize static snapshots of the relevant systems engineering (SE) data. While such representations provide important initial insights, they do not scale well with the depth of information, and do not allow deep exploration of the product ecosystem using diverse perspectives and a probing of different assumptions, thus restricting a comprehensive understanding of the system and making conflict resolution challenging. The sheer volume and complexity of the underlying data can often become too overwhelming, making the value of the data and the competitive advantages it can create lost in the noise.[2] Even if the architecture description of a complex system is captured formally in a model, existing tools are not sufficiently flexible and intuitive to provide actionable insights to pertinent cross-cutting questions (e.g., how does a system perform in wet weather conditions?).

To address such questions effectively, multiple SE data sources must be considered, including requirements, failure mode effect analysis, functional safety analysis, and system test data. One way to reason over heterogeneous, but interconnected, data is to transform it into a common formalism.[3] Existing research suggests that graph models are particularly well suited for such datasets.[5] In a recent study, for instance, a graph formalism was used to transform Simulink (https://www.mathworks.com/products/simulink.html) and SysML (https://www.omg.org/spec/SysML/1.5) data into a resource description framework (RDF) graph.[2] The resulting RDF graph created interconnections between data from multiple sources, which in turn when visualized provided the ability to explore questions associated with both specification and analysis of the system.

SIDEBAR: Failure Mode Effect Analysis

Product reliability and quality issues have tremendous financial and reputational impact on organizations. Failure mode and effects analysis (FMEA) (https://en.wikipedia.org/wiki/Failure_mode_and_effects_analysis) is an analytical method typically used in SE for assessing and reducing the risks associated with these issues.

The objective of a functional FMEA is to analyze the functionality of a system for failures in terms of no/degraded/unintended function. The effect of failure is severity ranked and a cause of each failure is described with an occurrence rating. The idea is to mitigate or prevent a cause, as prescribed in prevention controls. Each cause is rated in terms of detectability, and a corresponding corrective action is prescribed for it. Once the corrective actions are incorporated into the design, the severity, occurrence, and detection rating are revised for the new design that achieves the required robustness. The above process is done for each function a system performs. A function call tree has a large network of FMEA data, where failure of high-level functions is influenced by failure of lower level functions.

Figure 1 shows a class diagram of FMEA elements and their relationships. The elements (e.g., Failure Cause) represent the nodes while the relations between the elements form the edges between the nodes in the graph (e.g., Noise Factor nodes are related to a Failure Cause node through an "associatedNF" edge). In our tool, the collection of the FMEA elements and properties is stored in a graph database.

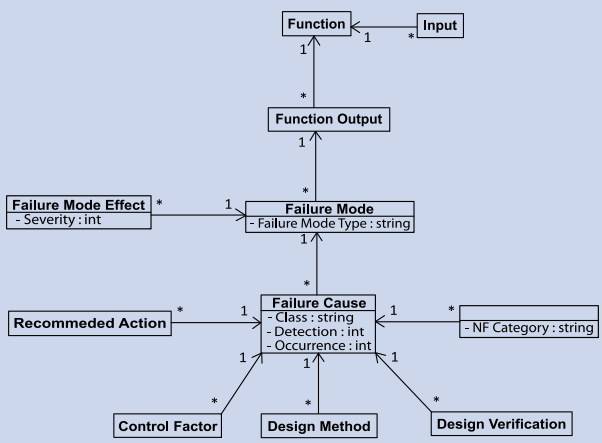


Figure 1. Data structure of FMEA shown as UML class diagram.

While access to and integration of data sources is an essential precursor, a much broader issue is how to make the data actionable to design review stakeholders. Visual analytics, an emerging field that fuses the power of data visualization with analytical models, promises to overcome these hurdles by providing interactive, user group specific capabilities to conduct deep and dynamic exploration of heterogeneous data.[7] By designing and developing a visual analytic system specifically focused on SE data, significant gains in collaborative discovery and perception of design-related information can be achieved, ultimately amplifying the user experience and accelerating time-to-insight.

COMMON FMEA TASKS

Today, most FMEA work is conducted using large, complex spreadsheets, which are cumbersome to navigate and difficult to collaborate on, significantly increasing time-to-insight. Our objective was thus to develop a tool that could accelerate the process of exploring FMEA data more rapidly, creating user-specified views on the fly and answering multi-perspective questions. Through a series of interviews with FMEA analysts, we identified several questions that an effective system should help answer. Some of the most important ones include the following.

- What elements do functions share?
- What mitigating actions were done for a similar function in another project?
- If a system part (e.g., microprocessor) is changed, what functions/features are affected and what elements are in the connection path? Which tests need to be redone?
- What is the overall health of a project in terms of robustness?

DATAHAWK

The design of DataHawk, a web-based visual FMEA data exploration and analysis tool, was driven by extensive discussions with prototypical users (system engineers, designers, and managers).[8] Following the task analysis, we identified five high-level design requirements. The system needs to: (*R1*) provide multiple exploration/analysis starting points to facilitate different types of investigations; (*R2*) capture the relational embeddedness of the large, complex, and interdependent FMEA elements; (*R3*) enable the tracing of dependencies between FMEA elements; (*R4*) provide summaries of key metrics; and (*R5*) be instantiated using an easy-to-use, familiar, and intuitive visual interface given that end-user visualization literacy skills could vary widely.

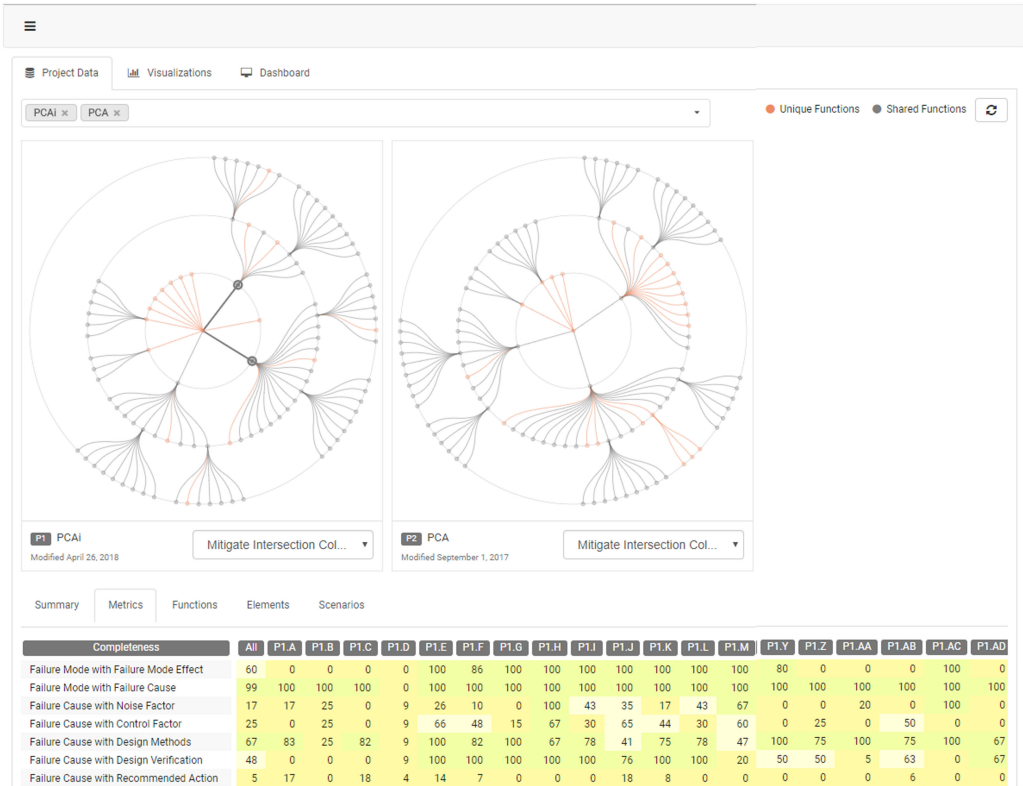
Following *R1*, we learned that an ideal workflow should follow an hourglass shape approach, beginning with a macro perspective of the data followed by a drilldown for details and then subsequent exploration. More specifically, the typical analysis would commence by selecting project dataset(s) of interest, selecting one or more initial entities for analysis, and then expanding the analyses to related entities. Selecting initial entities could be initiated through direct specification or a broader query (entity sets).

Multi-Pane, Coordinated Visual User Interface

Following *R5*, the UI is divided into three main components (a project pane, a function visualization pane, and a dashboard pane) utilizing a point-and-click interface.

Project Pane An FMEA can be done for one or multiple projects. Projects can share different types of elements, such as failure causes and noise factors. Analysts are, for instance, often interested in exploring how a recommended action in one project is reused across another. For this reason, a dedicated view showing the projects and their associated data was deemed important.

Each project consists of base functions, which can be decomposed into other functions using a function call tree. We visualize this function call tree using a radial tree visualization (see Figure 2) with base functions at the root and each subsequent hierarchy on concentric circles moving outward.[4] Since projects can contain multiple base functions, users can select focal base functions using a dropdown. Each function call tree is encapsulated in its own tile to allow users to bring in multiple projects simultaneously. The project tile contains a small information section, including the project ID, name, and last modification date to guide users of critical project information. The order of project tiles can be rearranged through drag-and-drops; each tile can



Completeness	All	P1.A	P1.B	P1.C	P1.D	P1.E	P1.F	P1.G	P1.H	P1.I	P1.J	P1.K	P1.L	P1.M	P1.Y	P1.Z	P1.AA	P1.AB	P1.AC	P1.AD
Failure Mode with Failure Mode Effect	60	0	0	0	0	100	86	100	100	100	100	100	100	100	80	0	0	0	100	0
Failure Mode with Failure Cause	99	100	100	100	0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Failure Cause with Noise Factor	17	17	25	0	9	26	10	0	100	43	35	17	43	67	0	0	20	0	100	0
Failure Cause with Control Factor	25	0	25	0	9	66	48	15	67	30	65	44	30	60	0	25	0	50	0	0
Failure Cause with Design Methods	67	83	25	82	9	100	82	100	67	78	41	75	78	47	100	75	100	75	100	67
Failure Cause with Design Verification	48	0	0	0	9	100	100	100	100	100	76	100	100	20	50	50	5	63	0	67
Failure Cause with Recommended Action	5	17	0	18	4	14	7	0	0	0	18	8	0	0	0	0	0	6	0	0

Figure 2. Project pane showing two projects loaded by the user, with unique and shared functions color-coded. Bottom panel shows the metrics on the loaded projects.

also be minimized or closed as needed. Functions are color-coded as unique (orange) or shared (gray) nodes to facilitate rapid understanding of commonalities between projects.

The bottom panel of the UI provides several different perspectives on data including the following:

- summary of function and FMEA elements;
- metrics;
- functions per project;
- elements of FMEA (to select the functions it is used in);
- scenarios (pre-defined and custom queries).

The first tab [see Figure 3(a)] contains a summary of unique functions and FMEA elements for each project. The second tab [see Figure 3(b), also Figure 2 (bottom)] contains a metrics tab providing completeness and ratio values for key metrics identified by FMEA analysts. Each function's metric is organized in a table and encoded using a two-hue color heatmap to focus analysts' attention to areas of concern (users can also define their own metrics). User can use the metrics to bring the function of interest into view. The third tab [see Figure 3(c)] allows an analyst to directly select a list of functions (a scrollable list) organized per project. Tools like search, filtering, and sorting are provided to aid the analyst. Additional info about each function is provided on a mouse click, and mouse-over operation highlights the function in the visualization. The fourth tab [see Figure 3(d)] allows the analyst to select function (s) by choosing an element from a scrollable list, which is sortable, filterable, and searchable. Clicking an element selects all functions it is present in and brings them into view. The fifth tab [see Figure 3(e)] contains a list of scenarios that can be used as a starting point by the user. Twelve predefined scenarios are populated as a drop down menu (e.g., add all functions without a recommended action). A user can also define a custom scenario through a query builder, e.g., bring in all the functions which have "Failure Causes" with "Recommended Actions" but not "Design Verification." All functionality in each tab apply to multiple projects as well.

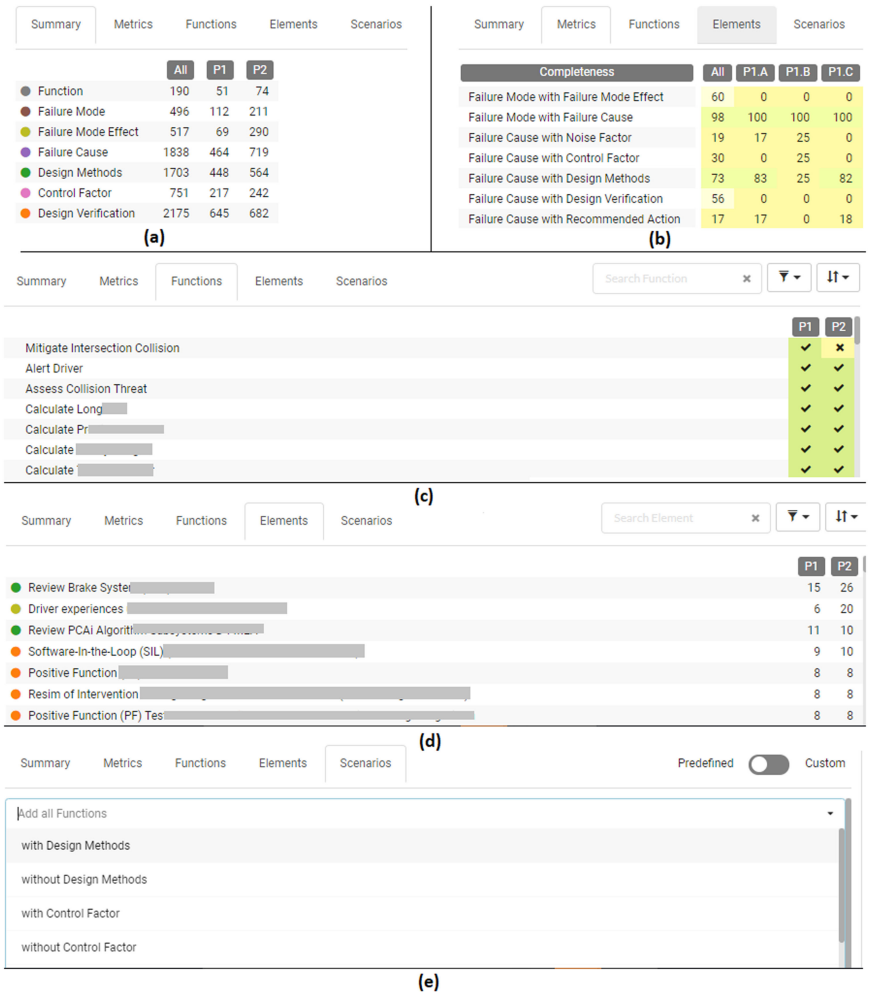


Figure 3. (a) Summary of elements; (b) Metrics; (c) Functions (per project); (d) Elements per project; and (e) Scenario (predefined and custom queries).

Any of the functions in the call tree have their own dedicated FMEA network. A function's FMEA network can influence another function's FMEA network. This happens as functions are connected together, where the output of one function feeds the input of the other. As a result of function decomposition, one function's failure mode becomes another function's cause of failure.

Visualization Pane Functions selected in the project pane are added to the visualization pane. While the two panes are separated, they are synchronized for an efficient transition from one to the other. The visualization view presents details about the selected function of interest, shown as a radial tree similar to the project data visualization (see Figure 4 top-left). Functional elements are color-encoded by type. In this view, the function forms the center of the concentric circles. The first level signifies how a function fails, the second level signifies the "effect and cause of failure". The third and the final level signifies how a failure cause is mitigated (i.e., Design Methods, Design Verification, Control Factor, Noise Factor and the Recommended Actions). A user can toggle the node/shared view button to quickly see which FMEA elements between two functions are common. At the bottom, the function identifier, function name, and the modification dates are indicated. The bottom right corner has functionalities for bookmarking a function. Once a function is bookmarked, the bookmarks can be used as a filter.

In conjunction with the radial tree view, there is an ability to create arc diagrams between two types of nodes for the final concentric circle. This visualization enables the analyst to understand

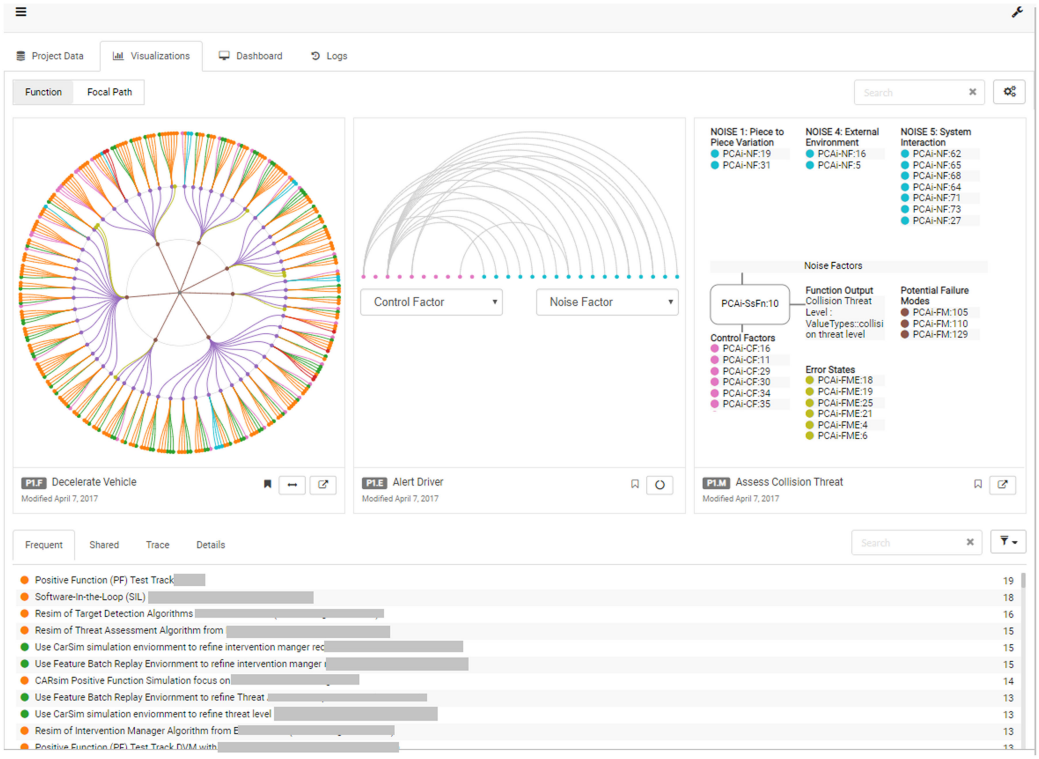


Figure 4. Visualization pane contains four different views: (top-left) FMEA network visualization via radial tree; (top-middle) arc diagram for function alert driver; (top-right) p-diagram for function Assess Collision Threat; (bottom) frequent elements.

relationships at the mitigation level specifically without cluttering the radial visualization. As an example, the analyst is trying to understand the relationship between "Control Factors" and the corresponding "Noise Factors" for "Alert Driver" function (see Figure 4 top-middle).

To anchor an analyst with familiar tools, parameter diagrams (p-diagrams) are also included as an optional view to see the underlying FMEA data (see Figure 4 top-right). We have the function in the mid-left, various types of noise factors on top, the function output and failure modes on the right and control factors and error states ("Failure Mode Effects") on the bottom.

Selecting an element in any of the visualizations creates a corresponding Sankey diagram (https://en.wikipedia.org/wiki/Sankey_diagram) with the element of interest as the focal element (see Figure 5) in the "Focal Path" tab. This diagram helps visualize all the paths leading up to the element and all paths leading away from the element (R3).

Similar to all other views, the bottom panel displays relevant information for the specific functions in a well-understood multi-tab tabular view with rich sorting, filtering, and search capabilities. The "Frequent" tab lists frequently occurring elements along with their frequencies; the "Shared" tab lists common elements across focal functions; the "Trace" tab provides information of traceability across functions; the "Details" tab shows detailed information for a selected element. All the tables and visualizations are highly interactive and coordinated. Selecting an element in any of the visualizations updates all other views to put that element in focus, including a path trace (see Figure 6) in the concentric circles as well as the Sankey diagram. This helps with consistent and anchored exploratory searches and rapid understanding of the data from different perspectives.

Implementation

DataHawk is implemented as a web application using the Python-based Flask microframework (https://www.fullstackpython.com/flask.html) and the JavaScript-based D3 visualization library. The FMEA data is stored in an OrientDB (https://orientdb.com/) graph database. FMEA data is

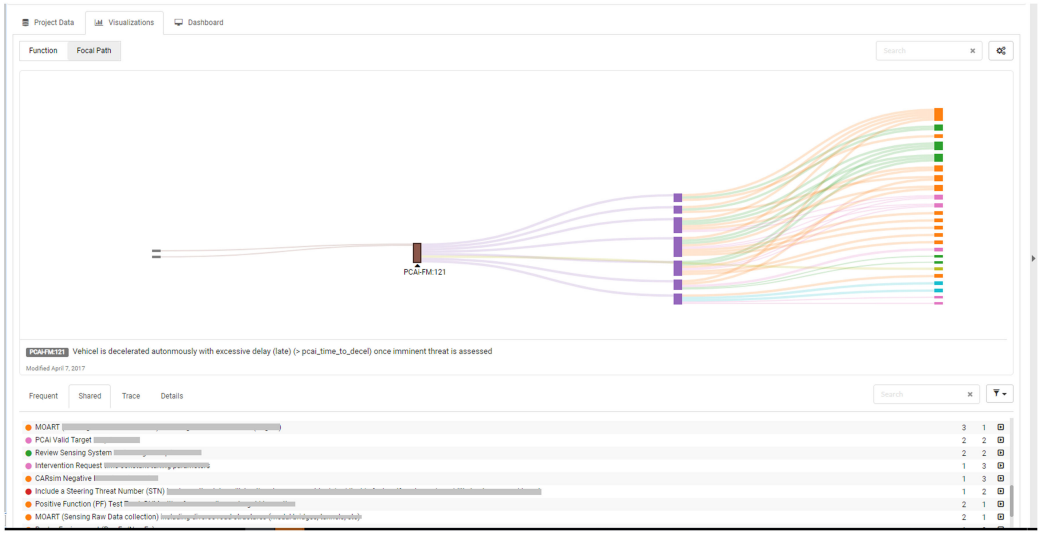


Figure 5. Sankey diagram showing traceability to elements affected by a failure mode.

usually drawn from a corporate relational database. The SysML model also has an FMEA profile that allows to author FMEA through that Domain-Specific Language. The model data is stored in a Cassandra repository (http://cassandra.apache.org/), from where it can be pushed to the corporate repository. The tool backend is a graph database that pulls information from Cassandra. A typical data set includes 5–20 high-level functions, each of which has an FMEA network. This equates to 2000–10 000 rows in a relational table, or about 30–100 k elements in a graph data structure, per project.

ILLUSTRATIVE USAGE SCENARIO

Systems engineers are often tasked to explore what failure causes have the most significant impact on the overall design and provide recommendations to the health of the system. We describe an illustrative scenario on how DataHawk can help support these tasks.

After opening the relevant project files from the data library, the engineer gains an immediate overview of all the functions and their inter-relationships by examining the radial visualizations and metrics summary heatmap (see Figure 2). The central prominence of some functions and associated poor metric levels focuses the engineer's attention to a few critical functions (i.e., P1.F "Decelerate Vehicle"). To explore a particular function more closely, the engineer selects it from the project pane and switches to the visualization pane (see Figure 4) for further investigation.

The visualization pane shows the selected function along with all of its FMEA elements. The engineer examines the "Frequent Element" tab [Figure 4 (bottom)], uses the filter function to focus on failure causes only, and double-clicks one (i.e., brake system did not deliver intervention braking deceleration due to improper arbitration) to bring in all functions affected by this element. The selected failure cause and its paths are highlighted in all function tiles. Switching to the "Focal Path" view (see Figure 5) provides insight into the end-to-end embeddedness of that failure cause.

The "Focal Path" view reveals that the "Failure Cause" (PCAi-FC:250) is related to four functions, and four different failure modes. The engineer also notices that this particular cause has no recommended actions defined. To explore if there are neighboring failure modes that also do not have recommended actions defined, the engineer switches to the "Scenario" tab [see Figure 3(e)] and selects the built-in scenario of functions without recommended actions. This now brings in all the functions without recommended actions. Together, these pieces of evidence lead her to conclude that this particular failure cause is not addressed by design yet. In fact, for the overall project, there is still a large number of failure causes, associated with different functions, without recommended actions.

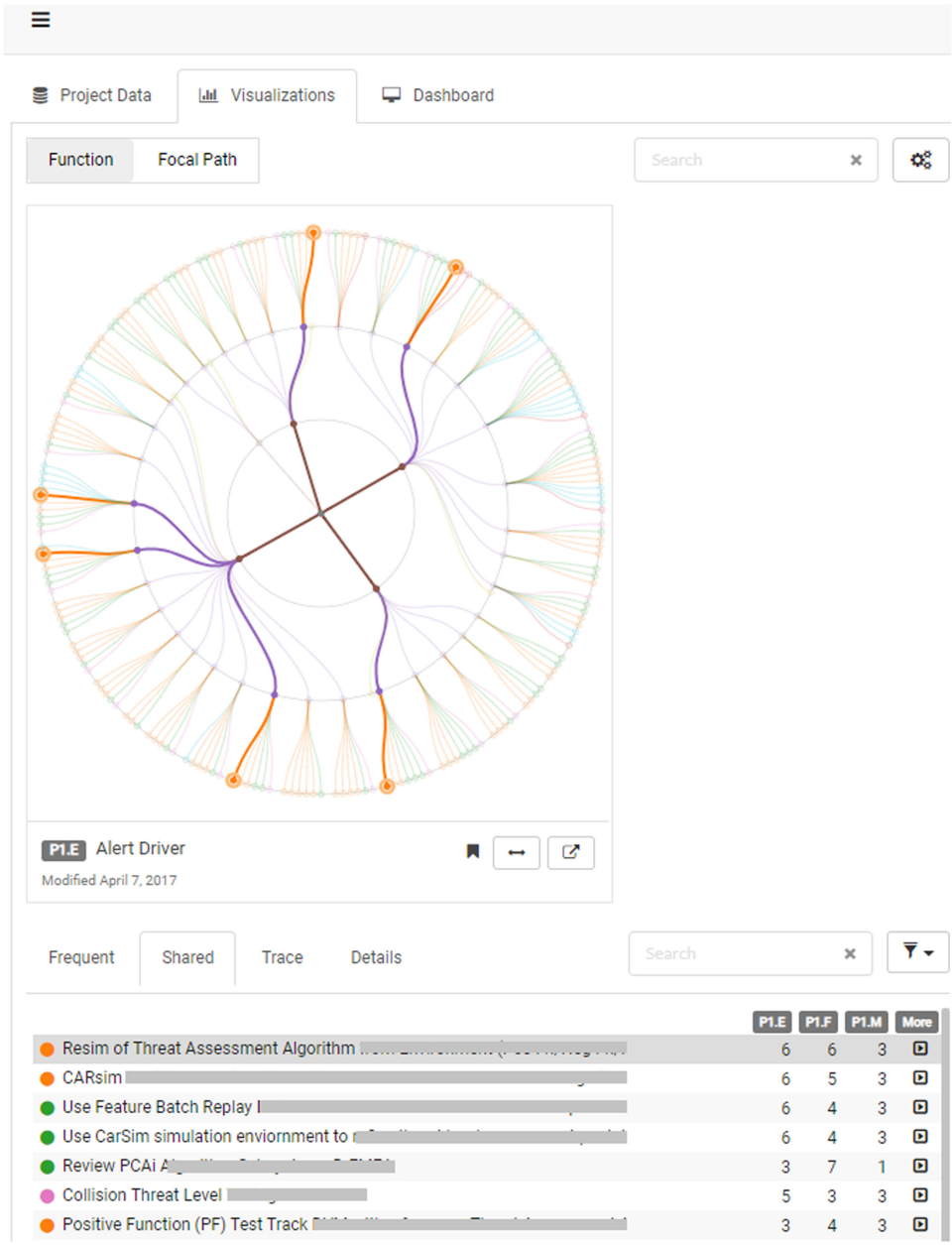


Figure 6. Example of brushing-and-linking interaction—user selection of an element brings in all functions it is present in and uses highlighting to trace the path.

The engineer can confidently conclude that the overall health of the system is potentially limited, hence requiring additional insight by the engineering team.

USER STUDIES

To assess the usability, usefulness, and overall utility of DataHawk, we conducted a series of user studies with a broad set of users, including engineers, managers, and researchers. Using best practices for visual analytic system design evaluation, users were introduced to DataHawk with a brief tutorial followed by task-driven and open-use sessions. During the task-driven session, users had to complete a series of tasks that assessed their overall understanding of the functionality and capabilities of the system using a think-out-loud protocol. Sample tasks included query tasks such as to open multiple

projects and identify common noise factors or more complex sense-making tasks, such as finding functions with a certain severity, occurrence, and detection levels that had no recommended actions. The research team observed each participant and took extensive notes to understand which UI elements were effective and which ones needed improvements. Following the completion of these two sessions, users were to provide qualitative feedback and suggestions for improvements of the tool from a task support, UI design, and system effectiveness perspective.

The qualitative and quantitative results were overwhelmingly positive across all survey categories. Common across all participants was the value of DataHawk for daily recurring FMEA tasks, reflected by comments such as "I will not mind working on this tool every day to check what updates happened on my project's FMEA, find the new areas where effort is needed and constantly monitor risk. I find the tool very powerful at supporting these scenarios.'' Participants were also impressed with the ability to explore different aspects of FMEAs but also commented on other desirable features, including the ability to generate reports of their findings dynamically. We are using this feedback to iteratively refine the design and capabilities of DataHawk.

DISCUSSION

Grounded in the observation that existing tools did not effectively support engineering design analysis tasks, we set out to build a tool that allowed dynamic exploration of heterogeneous SE data sets and answered questions spanning them. Spreadsheets are still predominantly used today, but they create unnecessary complexities when analysts have to handle multiple tables and analyze relationships across them. The cognitive load to perform tasks is significantly increased. DataHawk complements existing spreadsheets to support exploration across multiple SE projects. Our empirical user study found the system extremely valuable in finding potentially hidden patterns in the underlying data. The tool was particularly suited for tasks finding data anomalies, such as identifying element commonalities across projects and critical elements across different projects and functions.

Systems engineers often work across a variety of domains, from propulsion, vehicle dynamics, electronics, to network, infotainment, connectivity, safety, and quality. The design and development of a tool like DataHawk revealed that engineers crave more interactive visual analytic tools to help support their work.[6] However, we also learned that anchoring users in their existing routines, through familiar representations (e.g., p-diagrams) is crucial to broader acceptance. Another key lesson we learned is that existing SE processes need to be tailored, perhaps even fundamentally redesigned so that data can be curated for effective consumption in tools like DataHawk. In the short term, this means adopting templates for creating documents across the enterprise. In the long term, this means migrating from a document-based to a model-driven design environment. The formal syntax that models carry comes handy in accessing and transforming information, which is a critical step towards tools like DataHawk.

Data accessibility points to another implication for DataHawk—the interoperability between tools and the IT infrastructure of an enterprise. Earlier, we argued for using a graph as a common formalism for connecting and reasoning over heterogeneous data sets. Building such a graph is not possible without interoperability between authoring tools. We believe technologies like OSLC are vital to support the interoperability challenges but may also take time for adoption.

Overall, initial evaluations of DataHawk with FMEA analysts provide very promising results. However, we believe that this is just a starting point, and that the ultimate value is gained when multiple types of data sets are explored and analyzed. Extending toward functional safety data is a natural next step toward this goal. Functional architecture, technical architecture, strategy, program, all are important next considerations. Our aim is an engineering environment where engineers build models in their authoring tools and are able to quickly transition to tools like DataHawk, where they explore and find critical patterns across several data sources. They can bring results of their

analysis back to authoring tools or illustrate it during design reviews with the engineering team.

CONCLUSION AND FUTURE WORK

In this study, we have reflected on the design and development of an interactive visual analytics tool for FMEA. DataHawk enables users to dynamically interact with relevant data, explore it through an intuitive UI, develop salient analysis questions on the fly and answer them. User studies have provided supportive feedback on the effectiveness of DataHawk to answer questions that are time consuming to address through existing authoring tools. The multi-project exploration capability was especially recognized by users as an important value-added feature. Our plan is to continue the development of DataHawk and related tools to accommodate other data sets. To ensure scalability and broader adoption, we are also evolving the backend infrastructure of the tool along with the frontend so that more and bigger datasets can be onboarded. In doing so, we believe that we will make interactive visual analytic tools not only attractive for FMEA contexts but for other SE data analysis domains as well.

REFERENCES

1. A. Qamar, C. J. J. Paredis, J. Wikander, and C. During, "Dependency modeling and model management in mechatronic design," *Comput. Inf. Sci. Eng.*, vol. 12, no. 4, 2012, Art. no. 041009.
2. R. C. Basole, A. Qamar, H. Park, C. J. J. Paredis, and L. F. McGinnis, "Visual analytics for early-phase complex engineered system design support," *IEEE Comput. Graph. Appl.*, vol. 35, no. 2, pp. 41–51, Mar./Apr. 2015.
3. M. Broy, M. Feilkas, M. Herrmannsdoerfer, S. Merenda, and D. Ratiu, "Seamless model-based development: From isolated tools to integrated model engineering environments," *Proc. IEEE*, vol. 98, no. 4, pp. 526–545, Apr. 2010.
4. G. M. Draper, Y. Livnat, and R. F. Riesenfeld, "A survey of radial methods for information visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 5, pp. 759–776, Sep./Oct. 2009.
5. S. Feldmann *et al.*, "Towards effective management of inconsistencies in model-based engineering of automated production systems," in *Proc. 15th IFAC Symp. Inf. Control Problems Manuf.*, 2015, vol. 48, no. 3, pp. 916–923.
6. J. Heer and B. Shneiderman, "Interactive dynamics for visual analysis," *Queue*, vol. 10, no. 2, p. 30, 2012.
7. D. Keim, G. Andrienko, J.-D. Fekete, C. Gorg, J. Kohlhammer, and G. Melancon "Visual analytics: Definition, process, and challenges," in *Information Visualization*. New York, NY, USA: Springer, 2008 pp. 154–175.
8. D. A. Norman and S. W. Draper, *User Centered System Design: New Perspectives on Human-Computer Interaction*. Boca Raton, FL, USA: CRC Press, 1986.

Rahul C. Basole is a Managing Director & Global Lead for visual data science with Accenture AI, Atlanta, GA, USA, and former Professor with the College of Computing, Georgia Institute of Technology, Atlanta. Contact him at rahul.basole@accenture.com.

Ahsan Qamar is a Technical Specialist in model based systems engineering within systems engineering—product development with Ford Motor Company, Dearborn, MI, USA. Contact him at aqamar2@ford.com.

Biswajyoti Pal is currently working toward a graduate degree at the School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, USA. Contact him at bpal8@gatech.edu.

Michael Corral is a Research Engineer with the Research & Advanced Engineering Division, Ford Motor Company, Dearborn, MI, USA. Contact him at mcorral8@ford.com.

Matthew Meinhart is a Research Engineer with the Research & Advanced Engineering Division, Ford Motor Company, Dearborn, MI, USA. Contact him at mmeinha3@ford.com.

Arpit Narechania is currently working toward a graduate degree at the School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, USA. Contact him arpitnarechania@gatech.edu.

Contact department editor Mike Potel at potel@wildcrest.com.