Journal of Innovation
February 2025 | 26th Edition

Object Management Group

Navigating Contextual Complexity with Graph Visualization

Authors:

Liana Kiff
Tom Sawyer Software
lkiff@tomsawyer.com

Janet Six, Ph.D.
Tom Sawyer Software
jsix@tomsawyer.com

CONTENTS

FIGURES

TABLES

Graph-based data structures provide the flexibility required to encode and analyze the increasingly vast, complex, and connected data spaces we are faced with in modern systems. The use cases for graph visualization appear in many disciplines. This paper will focus on use cases relevant to human understanding and analysis of systems-of-systems models and knowledge graphs.

Context is the source of meaning in a data-centric world. For each data content element there may be many legitimate contexts described by unique meta-data. Managing the coherence of those connected contexts within a system of systems is a critical factor impacting the integrity of every potential use case for each piece of managed data.

Graphs provide another tool for communication, but graphs run into measurable human perception and cognition limitations. This paper discusses the practical applications of advanced graph-based navigation and visualization techniques for aiding human comprehension and interaction with complex, multi-layered systems. We cite specific use cases and examples, with reference to deeper research.

1 THE HUMAN FACTOR

"The more things change, the more they stay the same."[1]

There has been a nearly continuous tug-of-war between the performance and convenience of storing information as opposed to the needs of people to consume and manipulate it. The use of graphs for storage doesn't eliminate this battle. Graph-based databases and visualizations only move the limits of what is possible in certain directions and deliver new ways to express the challenges. The same tensions still apply, the same principles are still at work, and many of the same technical solutions are still employed.

If graphs allow for greater facility to link related information, it is also the case that you can build a graph that makes it hard to deliver the answer to the question your user wants to ask. Furthermore, if the data and the human brain now bear more resemblance to each other in terms of networked knowledge, humans are still functionally limited by human faculties of perception.

In their 2018 paper, Vahan Yoghourdjian and colleagues [1] review and summarize the findings of the best available studies to date regarding the human cognitive limits encountered when working with graphs of different size (number of nodes) and density (number of edges). One of their conclusions is that not enough work has been done to adequately understand the best solutions and guide designers to the most appropriate knowledge graph visualization techniques for their purpose. Many studies so far have included limited numbers of users, and the major source of those users (college students) are not necessarily representative of the real-world user base. Burch et.al. [2] concluded in their survey of empirical studies that 20 years of research has

[1] Jean-Baptiste Alphonse Karr - 1849

only touched “the tip of the iceberg” in understanding the cognitive factors of working with graphs of any size.

Some studies have been conducted that help direct partitioners in the right direction for visualization using generic approaches, based on the size and density of information and a known analytic challenge. [3] Domain-agnostic visualization tools, and well understood layout algorithms provide a variety of approaches for graph display that, while effective for some, can fall short of meeting domain-specific expectations for end users. [4]

It is our job as the architects of information spaces to provide humans with the tools and affordances that let them explore data effectively, with fewer barriers, and less noise. While the human factors research is still catching up, the best software development tools and platforms make it possible to apply a full range of interactions so that interface designers can work with users to arrive at the best combination for each problem space.

Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space. -- **Edward Tufte**

1.1 Orientation and Navigation

Since it was published in the 1970’s, Air Force Colonel John Boyd’s Observe-Orient-Decide-Act cycle, referred to commonly as the OODA-loop, has guided many thoughtful UI design approaches for critical monitoring systems.

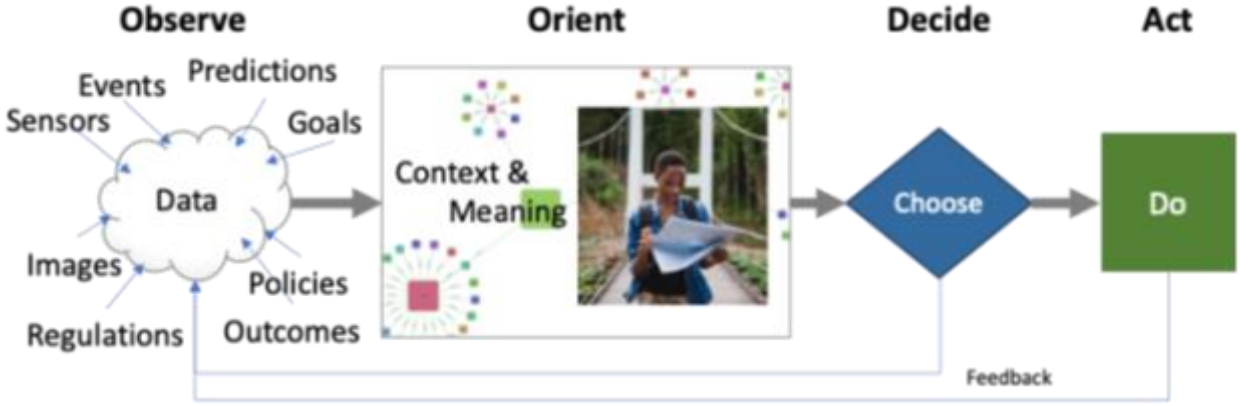


Figure 1-1: Example illustration of Observe, Orient, Decide, Act (OODA) model.

This conceptualization was developed largely to explain cognitive processes required by humans to react in situations where there is high information density (complexity), and high cost of errors in judgement. It is useful to guide system design considerations when life-changing decisions need to be made quickly, but it also works when considering situations where a human is being asked to make an informed decision where there are many facts to consider and process. The outcome of a good user-oriented solution that follows these principles should be good situation awareness, and the right tools to make the correct responses easier to achieve, during both normal and abnormal operating situations. [5]

In recent years, the observability of information has exploded, with new sensors, algorithms, and machine learning summarizing and expanding the available data in nearly every domain. Orientation in these increasingly complex data spaces can be difficult. Maintaining orientation when navigating within those spaces is a further complication.

Intuitively, sighted persons know this as users of maps. If you don't have both a clear idea of where your starting point is, as well as where your target destination is, you may have trouble orienting your mental model with the map view. As users of information systems, we also know that "you can't get there from here" is not only a real-world wayfinding problem, but also a common user interface nightmare that can interfere with both navigation and orientation.

It is also true that we don't always know where we are going when we start a journey, but we do have a starting point "You are here," and even researchers generally begin with a hypothesis which sets a direction. It is important to be certain when "you can't get there from here" is the correct answer to a specific question, and not a flaw in the affordances that have been provided to the user [6], either in the underlying graph, or in the presentation of that graph.

Graph-based data management and graph-based visualization offer new tools that afford users new options for orientation and navigation across large data systems. They also offer system designers new ways to confound users with unnavigable information spaces. If we fail to provide tools to manage complexity effectively, the user may be left navigating in a dense, dark forest. If we are too heavy-handed when managing complexity, we might hide valuable information the user needed.

1.2 Complexity Factors

Filipov, et.al., summarized a set of network visualization disciplines in their meta-survey and roadmap-based discussion of the state of the art of network visualization research. [6] Their work identifies the specific special considerations for graphs that are very large or multi-faceted, with special techniques for multi-facet needs that include time, geospatial aspects, or layering of concerns. These are all common requirements of graphs that represent systems.

Statistical	Structural	Layered	Spatial	Temporal
Emphasizes clusters of connected or similar nodes, based on characteristics of the population, which may be calculated.	Emphasizes the shape or cyclic nature of connection patterns in the data.	Isolates abstraction layers through filtering, expanding /collapsing, swim lanes, or other	Superimposes a graph on geographical or other domain specific spatial maps (e.g. buildings, processing plants).	Synchronizes with timelines or otherwise provides a temporal map of change over time.

Table 1-1: Data abstractions that may require specialized visualization to aid human comprehension.

The sheer size of the knowledge graphs we interact with is one form of complexity. However, aside from the problem of data volume, each of the significant variables or facets may need to be addressed as a unique abstraction of the data, or may need to be combined, as with change over time combined with geospatial awareness. As we understand geospatial data in the context of maps or renderings that are spatially proportionate, it is also the case that many other domains provide either physical or conceptual maps or metaphors as well, and representing these conceptual maps is likely necessary to effectively present data to users in those domains.

2 NAVIGATION AND GRAPH STRUCTURES

Many of us have been conditioned as users of technology to navigate with trees, which afford movement up and down through hierarchical structures, but to move sideways, to another branch, you may be taking a big move that fundamentally changes your context and point of reference. Other types of graphs, on the other hand, facilitate movement in any direction afforded by the graph, which can make maintaining that point of reference even more challenging if the right contextual clues are not present. von Landesberger et. al. [7] present a very thorough overview of graph-based display strategies and interaction methods. They further describe navigation strategies as falling into one of three categories: top-down, bottom-up, or middle-out.

Top-down navigation starts with the whole graph and affords navigation to drill into the data. Identification of a node of interest may involve some form of visual analysis. Top-down may be the preferred starting point for data science tasks, or system monitoring tasks where the whole, or some abstraction of the whole system informs the point of interest.

Figure 2–1 is an example of top-down navigation in a network display that uses edge color to indicate bandwidth stress. Tabular display of connection details supports selection and isolation of one more nodes within the network for deeper analysis. The overview display supports orientation from the zoomed-in portion of the graph to the wider network. Legends and filters are displayed on the left-hand side.

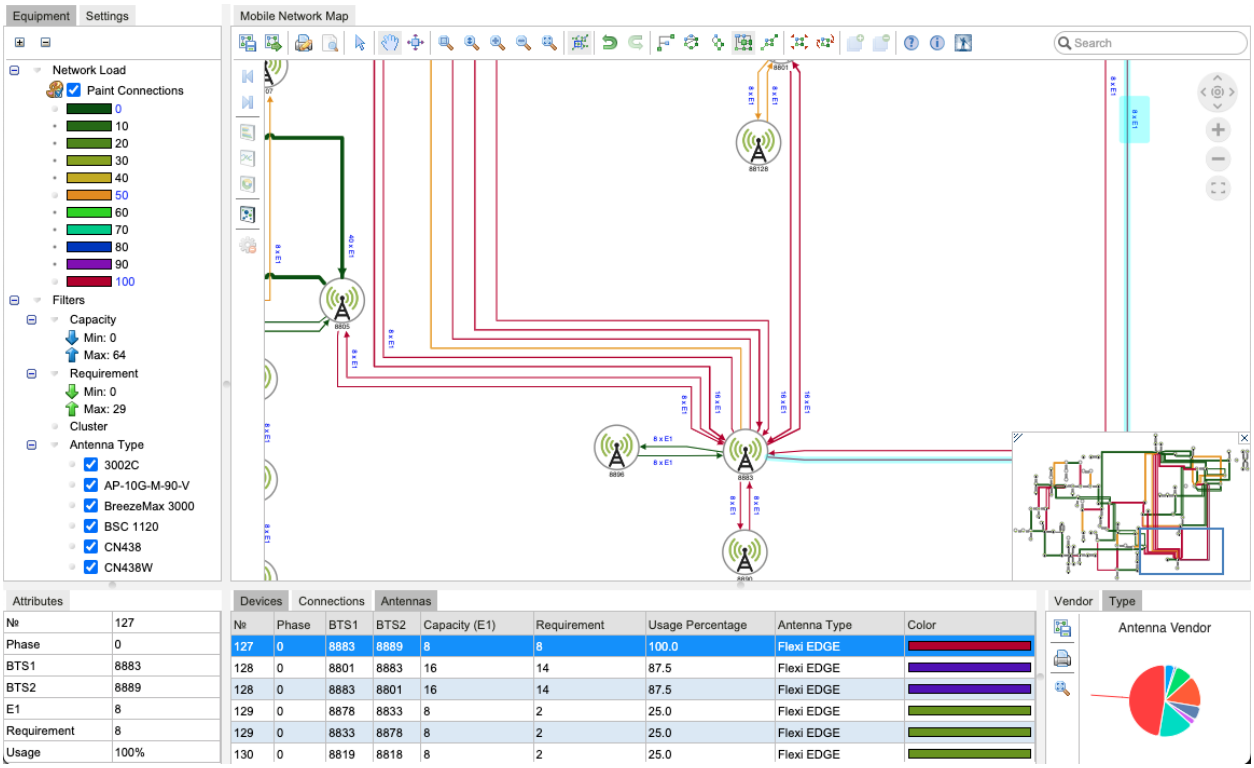
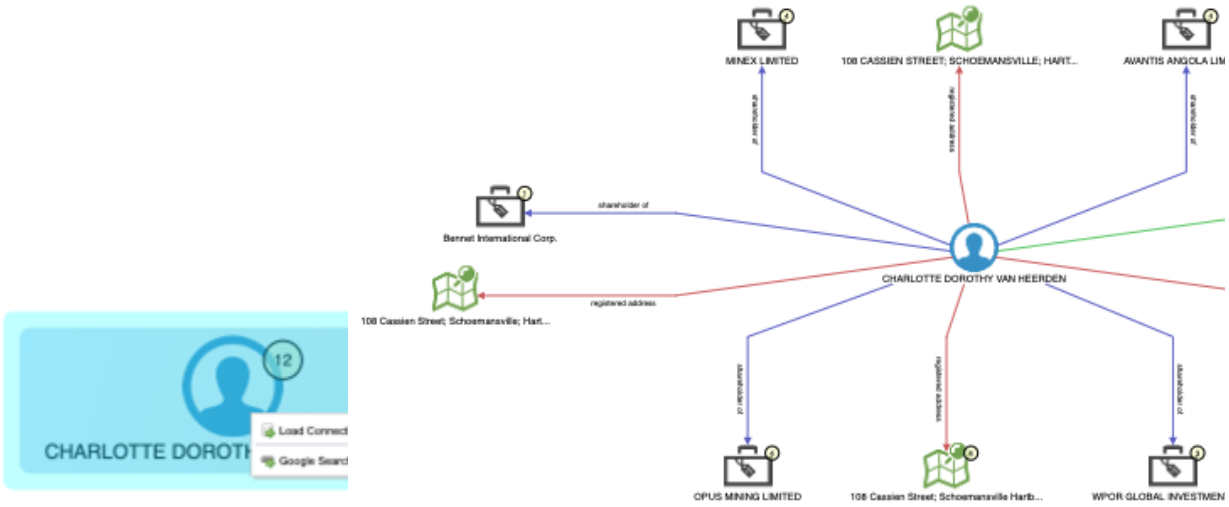


Figure 2-1: Top-down Visualization of a network.

Bottom-up navigation begins from a point of interest and supports the user to explore outward. The starting node may be identified by targeted search. Bottom-up navigation is suited to procedural and task-specific cases, such as investigations, where the starting point is known. In the case of a monitoring system, the starting point may be a known problem.



(a) (b)

Figure 2-2: Bottom-up approach starting with a specific node. A context menu supports searching outward from the selected node.(a) Loading connections expand the graph. (b)

Middle-out navigation begins with some abstraction or simplified graph and supports moving to another representation layer. The beginning abstraction may be determined by an algorithm, or by domain-specific considerations. Middle-out navigation may be suited to browsing or indexing a large system, such as is often the case with a wiki or website navigation scheme.

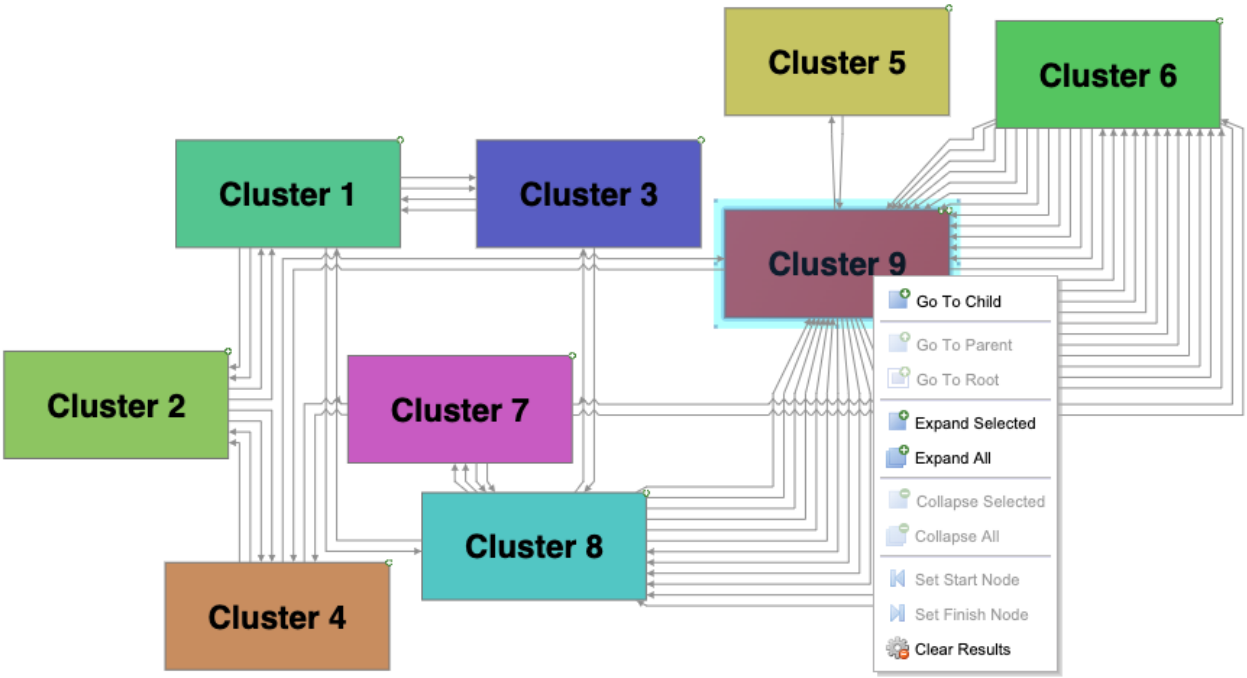


Figure 2-3: Middle-out navigation in a network where a clustering algorithm has been applied. A context menu supports drilling into the selected cluster.

Within any of these navigation strategies, it is also possible to provide selective navigation based on the type of edge(s) to be navigated, or to specify the distance to navigate, in terms of either the number of connections (hops), or by somehow specifying the target objective of the navigation (the node(s) to be reached).

2.1 Tools for Navigation and Interaction

It is possible to categorize methods of user interaction as either acting on the data to be displayed (selection, filtering, calculation), or on the way that data is presented (layout style, or other visual parameters). [7] These tools work together, and how they work forms the backbone for how navigation is afforded within the user experience.

Control Method	Aggregation	Abstraction	Context	Spatial	Navigation
Data-level examples	Statistical clustering, or domain-based aggregation.	Abstraction by query or through Graph AI summarization techniques.	Query-based context selection	Location-based query, either geographic, or topological.	Path query or path traversal algorithm.
Display-level examples	Grouping, nesting, combining.	Applying attribute-based, time-based, or statistical abstraction.	Filter-based context visibility	Projection to a map, or layout constraints.	Follow links, find by search or selection.

Table 2-1: Sample of tools and approaches for context and complexity management.

3 Special Considerations for System Engineering

Model-based System Engineering (MBSE) is discipline that relies heavily on diagrams as both a canonical input mechanism and communication tool for collaboration. SysML v1 is a modeling system that utilizes diagram syntax as the declarative form of the model [8]. This mode of visual graphical editing can be natural, but it is also prohibitive in terms of the number of different diagrams that are required to communicate all aspects of the design or behavior of the system for all the stakeholders that demand customized diagrams to illustrate specific concerns.

Furthermore, it is up to the engineers modeling these systems to ensure that changes in the model are manually propagated to all the relevant diagrams. This can lead to diagrams that are missing elements or connections or contain other inconsistencies that are difficult to identify through manual review and may introduce errors. Undetected, errors like these can be propagated across multiple teams and product components. [9] The longer such errors remain undetected, the more likely they are to create defects in implementation that can become costly.

Diagram-focused standards face the same challenges as older, document-based solutions. Both drawings and documents can introduce inconsistencies that are hard to detect because of the difficulty of thorough analysis and the limitations of humans to correctly integrate engineering context that is spread across many diagrams. [10] Manually maintained diagrams may fail to reflect important elements because an engineer didn't include them, or the standard didn't allow for them. [11] Producing, maintaining and reviewing these necessary diagrams consume considerable engineering time. Under this pressure, only the most critical diagrams are produced, and many useful diagrams won't be maintained.

SysML v2 uses a graph-based approach to modeling a system. [12] Graph-based data structures and data-driven visualizations are an important tool in the evolution of model-based system engineering, since they can readily support the constructs demanded of the system modeling language. Graphs readily support direct navigation between model representations based on the underlying system data. Data-driven visualizations are not predicated upon the existence of the correct manually created diagram, or a diagram specification that includes the items of interest. [13]

Graph analysis and graph visualization are tools that may be employed to detect model inconsistencies. Graph structures support seamless navigation between abstraction layers and other defined contexts, enabling collaboration between end users of different disciplines.

Automatically generated graphs of engineered systems have the capacity to illustrate model inconsistencies because a query of the model data will return what is actually present in the model, not what the engineer manually placed in the diagram. Automated, data-driven visualization of system models have the potential to reveal direct connections, which might otherwise remain implicit because they were not included in the prescribed or manually maintained diagram. Conversely, they may reveal missing connections, by showing unconnected ports on elements that should be related. [9]

Dynamic and interactive graphs and automated graph-based layouts support human comprehension of increasingly abstract and complicated data landscapes. [14] Manually creating visualizations for each of these cases is prohibitively expensive. Data-driven graph-based diagram generation is likely to deliver meaningful productivity improvements throughout a system lifecycle.

4 USERS AND USE CASES

The definitions below align well with the persona's identified by Li et.al. [4] Individual users will move between these modalities if they are both a designer and user of graph technology, or both an operator and analyst of systems in operation. We present examples of how these reference personas are reflected across participants in the practice of system engineering.

- **Graph designer or ontologist:** A well-described system requires a well-described, and possibly domain-specific meta model or schema. A graph schema designer needs to see and manipulate all the details of the meta-structure of the graph. The meta-modeler needs to understand the finer points of logic and the effect that rules and constraints have on the behavior of the resulting instance graph(s). When ontologies or other schema get very large, tools that apply to the data scientist/analyst may be needed.
- **Data scientist/analyst:** The analyst tends to act on vast amounts of data to generate insights or to perform visual analytics, as well as use graph queries, graph analytics and advanced features of graph visualizations to arrive at insights. In many big data analysis cases, graph analytics are used to generate graph layouts that resemble and are perceived as scatterplots. While the data source may be a knowledge graph, individuals are less important than population effects and statistics. Data scientists may translate the results of these explorations into new knowledge to be acted on by other users.
- **Knowledge Graph User:** A knowledge graph user may not know or care that data comes from a graph-based source. They are tasked with getting something done quickly and correctly, whether that is to describe a system that complies to the meta-model or using the knowledge graph to draw a conclusion about a system. These users tend to work on smaller amounts of task-specific information, presented in a fashion designed for clarity and efficiency, and possibly in support of a specific process or domain.

Each of these users has a need for intuitive graph navigation through the system. How complex this navigation becomes is dependent upon the complexity of the information space they need to navigate, and the task they are engaged in. The following sections will describe examples that apply to these users in system engineering or similar disciplines.

- Graph Designer: Navigating and understanding large meta-models
- Data Scientist: Navigating and interpreting large graphs
- Knowledge Graph User: Navigating systems knowledge graphs and digital threads

4.1 Navigating and Understanding System Meta-Models

Meta-data design tasks are distinctly different than modeling instance data. This is as true for graphs as for relational data, and as true for large systems as for any other type of graph data.

Triple stores and graph databases offer new ways to store and query meta-model information, but the process of visualizing the meta-level concepts, links, and rules is distinctly different than for the users on the other end of the data pipeline, as described in section 4.3.

Graph schema editors require the basic data engineering tools for creating and modifying the structure of the meta-model: supplying the details of a schema, adding and modifying nodes, edges and attributes, or describing the rules and constraints on the meta model. The meta-model

prescribes all the things, and patterns of connected things, that it is possible to represent with that meta-model. The simple model depicted in Figure 4-1 includes one type of node or vertex in the graph, called a Device, which has several attributes. It includes one edge type, Connection, which also has attributes. Each attribute has a defined data type. Rules or constraints could be added to restrict the string values allowed for specific attributes, or prevent instances described by the model from allowing a Device to have a connection to itself, for example.

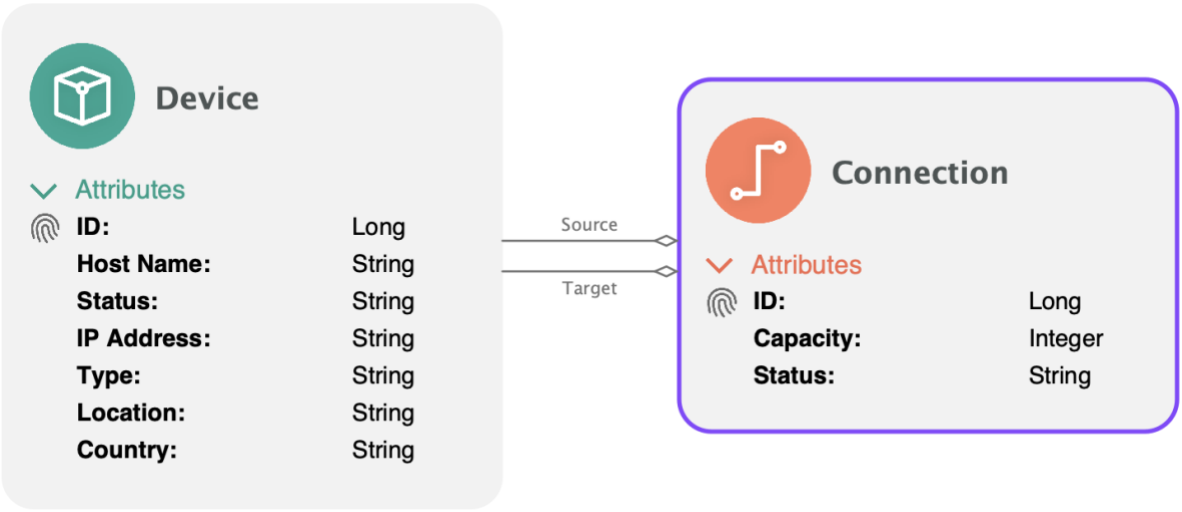


Figure 4-1: Simple meta model with one node type and one edge type.

Tools for interacting with a graph schema, whether an ontology or property graph, should also afford users with the utility to analyze the schema at various levels of detail, to support discovery of unique properties or patterns within the schema, and to identify inconsistencies or gaps.

Industry ontologies are an example of large and complex graphs that can benefit from multiple modes of visualization that are not always readily offered in schema design interfaces. Graph-based systems can help to both illustrate and to enforce common patterns in collaborative or crowd-sourced information models, with less overhead in review and analysis of new submissions to the model.

The following examples are derived from the last freely available draft of the Industry Foundation Classes Model (IFC4x3), the foundation of the Building Information Modeling standard ISO 19650-1:2018, maintained by buildingSMART International, and accessible from buildingSMART.org. The RDF ontology is generated from the native form of the IFC model, which is STEP (ISO 10303-21). All the following visualizations were produced with Tom Sawyer Perspectives.

Figure 4-2: Hierarchical visualization of a sample RDF model of IFC4x3.

The IFC ontology is extensive, with more than 1500 defined classes and more than 7000 edges. The hierarchical visualization of this ontology tends to be quite flat, which makes it challenging to identify patterns, or to navigate effectively, as seen in Figure 4-2.

By contrast, the circular layout in Figure 4-3 highlights the distinct modeling patterns at work in this ontology and more readily affords an investigation of sections of the model that indicate unusual patterns, or lack of patterns. Approaches for automatic graph layout, such as the results we see in this circular layout, address several NP-Complete and NP-Hard problems such as minimizing the number of edge crossings, minimizing total edge length, and minimizing the number of inter-cluster edges. Due to the nature of the graph layout problem, automatic graph layout technologies use heuristic approaches to produce quality visualizations. See [16] for further discussion.

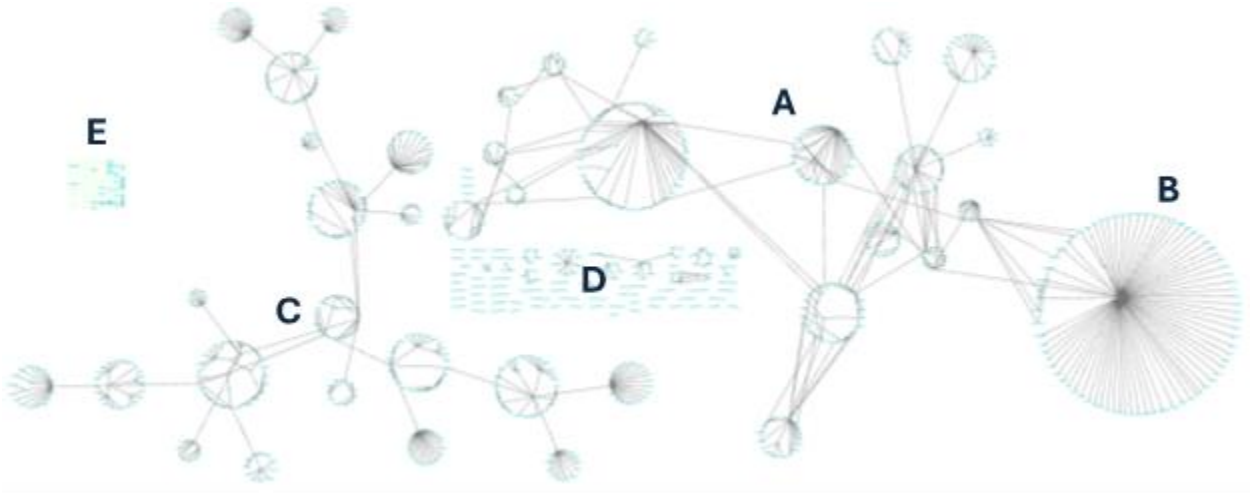


Figure 4-3: Circular layout of the IFC4x3 sample ontology showing distinct modeling patterns.

The labels in Figure 4–3 show patterns related to these portions of the model:

- Elements near label **A** represent drawing primitives and spatial elements
- The large group near **B** is the set of primitive measures
- Groups near **C** represent domain-specific portions of the model.
- Disconnected nodes near **D** are resource level elements.
- The group of disconnected nodes at **E** are Enumerations.

This illustrates the distinct separation of the drawing layer (A) from the domain concept layer (C). While views such as these improve the chance that interesting patterns, or deviations from

standard patterns will stand out, it is not well-suited to understanding details of the ontology or navigating the ontology.

To facilitate easier navigation, it is helpful to utilize additional domain knowledge. The IFC model has well-defined structure, as illustrated in Figure 4-4. In the case of this RDF source, that additional domain meta-data was either lost in translation between the model serializations or was only present as external documentation. These meta-level concepts that guide the design of the model are helpful for navigation and orientation.

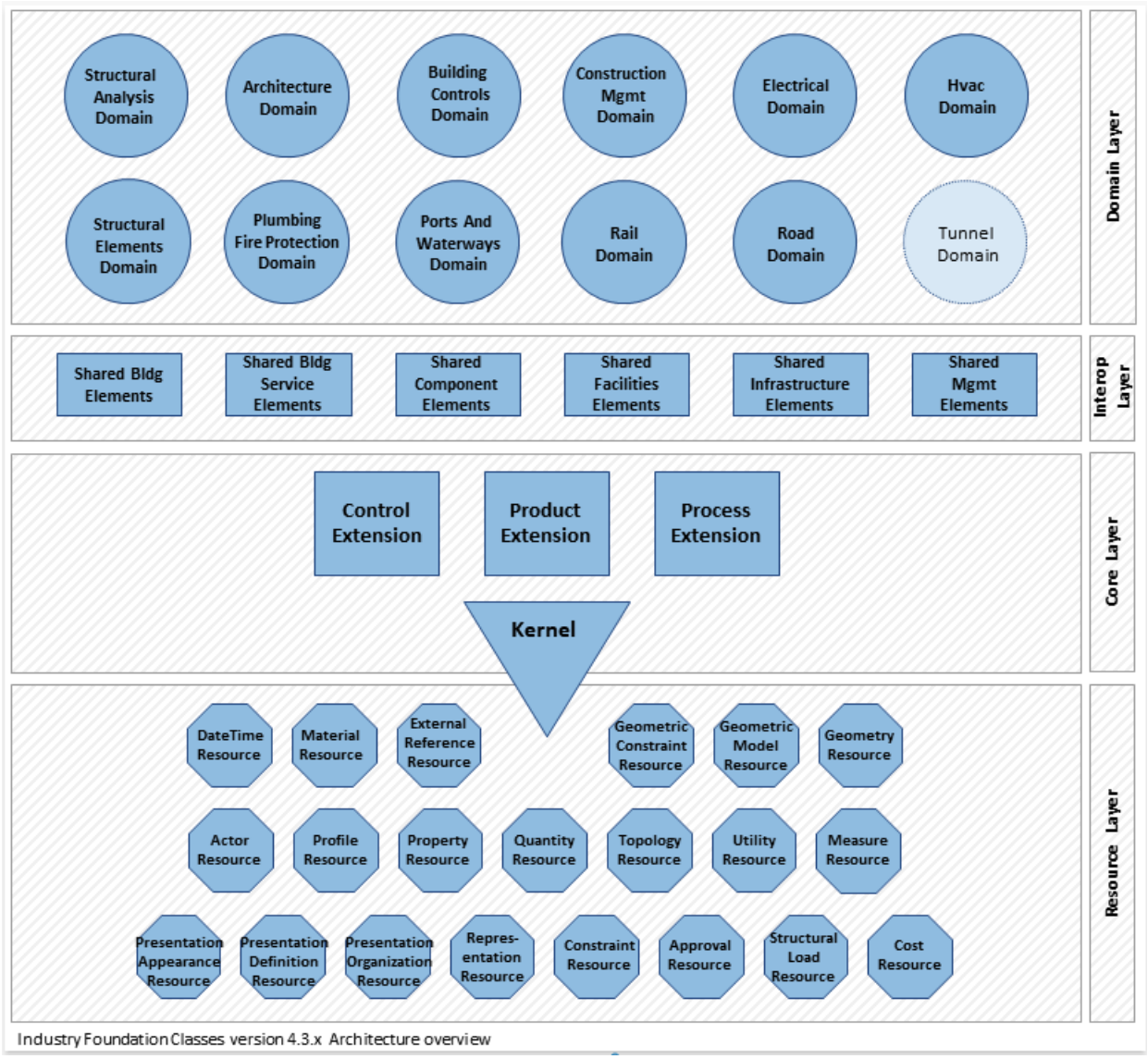


Figure 4-4: Architecture diagram showing the underlying structure of the IFC4x3 model. [17]

By utilizing this additional information in the generation of graph layouts, it is possible to examine the patterns at work to distinguish elements in the model and how they work together in the context of each domain.

Figure 4–5 shows detail of the inheritance hierarchy for the IFC Flow Controller class, which is the superclass for devices such as dampers, valves, and electrical switching devices, which all belong to their own specific mechanical sub-domains. This layout shows that kernel level concepts are further refined to support domain-specific concepts for building control and electrical systems by aligning domain membership within vertical columns. The horizontal swim lanes help to clearly illustrate a pattern in this model for defining products and enumerating their types.

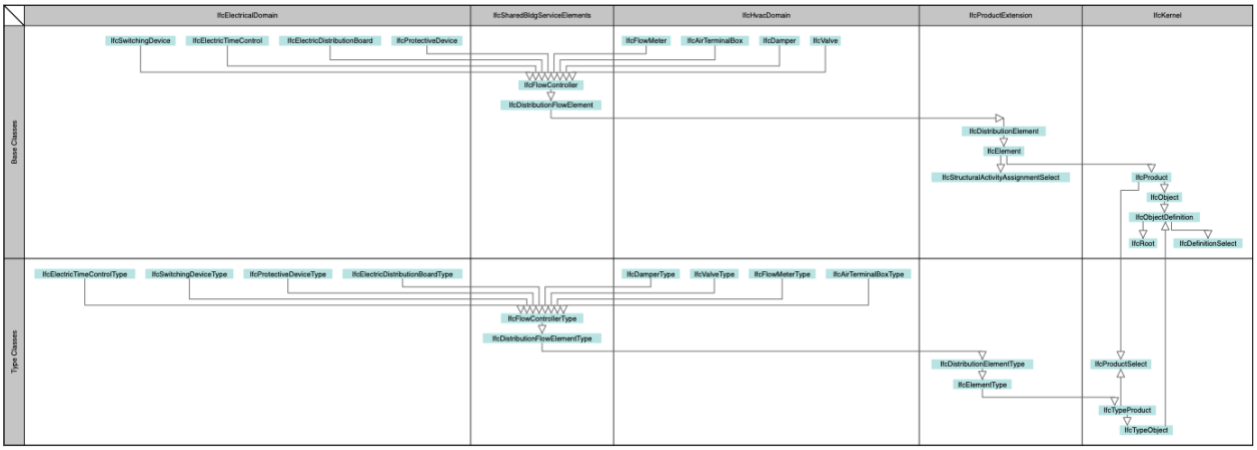


Figure 4–5: Two-dimensional swim lane graph layout applied to a portion of the IFC4x3 model.

Ontologies are not all the same, and most of the time, an inheritance hierarchy is not enough to help people to understand them and apply them correctly. The use of swim lanes, in this case, enables navigation within the context of other available abstractions, providing both predictability in where information will be displayed, and contextual clues that enhance understanding.

The ontology model we examined is one effort to describe a system of systems in an industrial field that involves many disciplines. System complexity is compounded by the array of tools, and standards that must be navigated and harmonized to ensure that shared data resources retain integrity through every transformation and across a product lifecycle. Furthermore, the available data needs to be correctly understood, within the appropriate context(s) of use, and applied by end users of varying disciplines. These are the same tensions we find in system engineering more generally.

Model-based System Engineering practices are still in the process of moving from manual maintenance of diagrams (diagram-first engineering) to more data-driven techniques (diagram generation). Data-driven solutions for diagram generation, as those supported by Tom Sawyer Software's Model-Based Engineering plug-in for Catia/NoMagic, can meet with some resistance if the automatically generated drawings aren't consistent with user expectations, or interfere with their ability to orient quickly to the model data. The development of SysMLv2 is working to enable a model-first approach to system engineering. Figure 4–6 is one possible visualization projected from a SysMLv2 model, by querying the model store and applying a visualization algorithm.

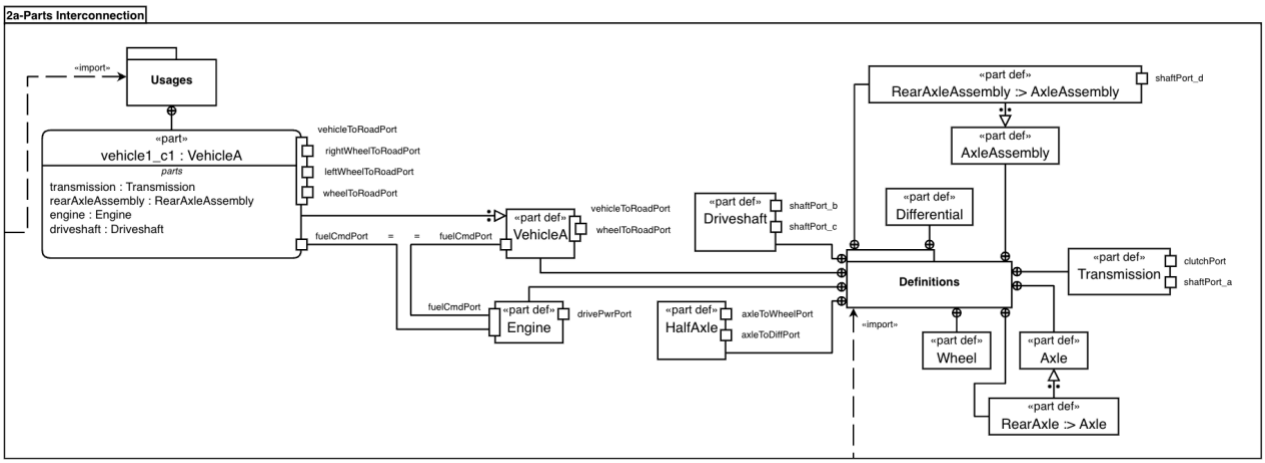


Figure 4–6: SysMLv2 Part Interconnection visualization from Tom Sawyer SysMLv2 Viewer.

End users following any prescribed modeling practice, especially in domain-specific modeling contexts, benefit from some standardization at the visualization layer – both in the expectations of specific visualizations, but also in the consistency of automated layouts as models change over time. Constraints on automated layouts can be useful to assert consistency, however, the use of constraints can restrict the effectiveness of automated layout algorithms by forcing them to create less-than-ideal node placement or edge routing.

The examples in Figure 4–5 and Figure 4-14 suggest that the answer may lie in providing and utilizing sufficient meta-data to arrive at more consistent automated representations, or a representation that is more true to domain-specific conventions to achieve a higher degree of user acceptance.

4.2 Navigating and Interpreting Large Graphs

Working with larger graphs presents several obstacles including: where to start analysis, how to focus in on relevant data, and how to communicate key findings. In this example, we explore how to apply visualization and analysis to a social network to understand how agents are working together in a system. In Figure 4-7, an example social network shows human agents linked if they have worked together.

It is possible to focus into especially relevant parts of the network applying filters which indicate, for example, types of activity, or geospatial locations. It is also possible to apply data-driven analysis which indicates higher importance for certain agents; or, as in the following figures, combine the findings of human and data-driven analysis.

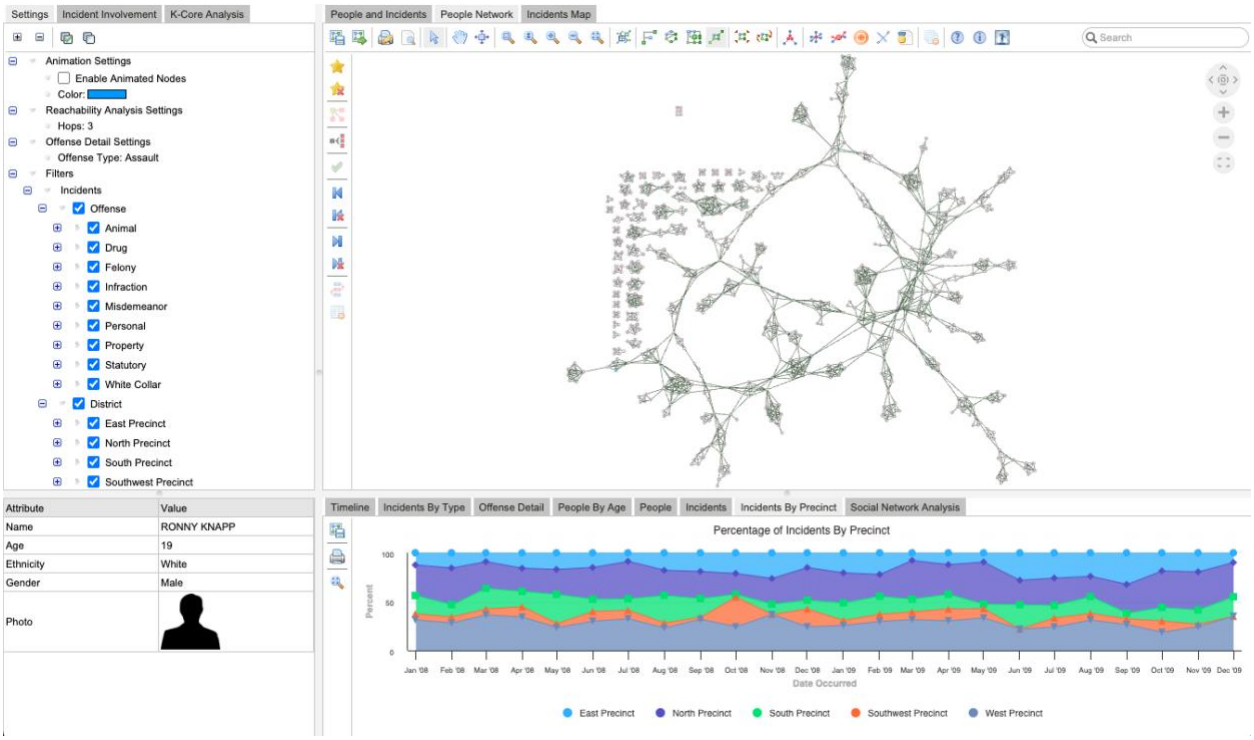


Figure 4-7: Social network example showing how agents and the activities are linked.

Analysts may determine key agents in the network; these are noted in the graph in Figure 4-8.

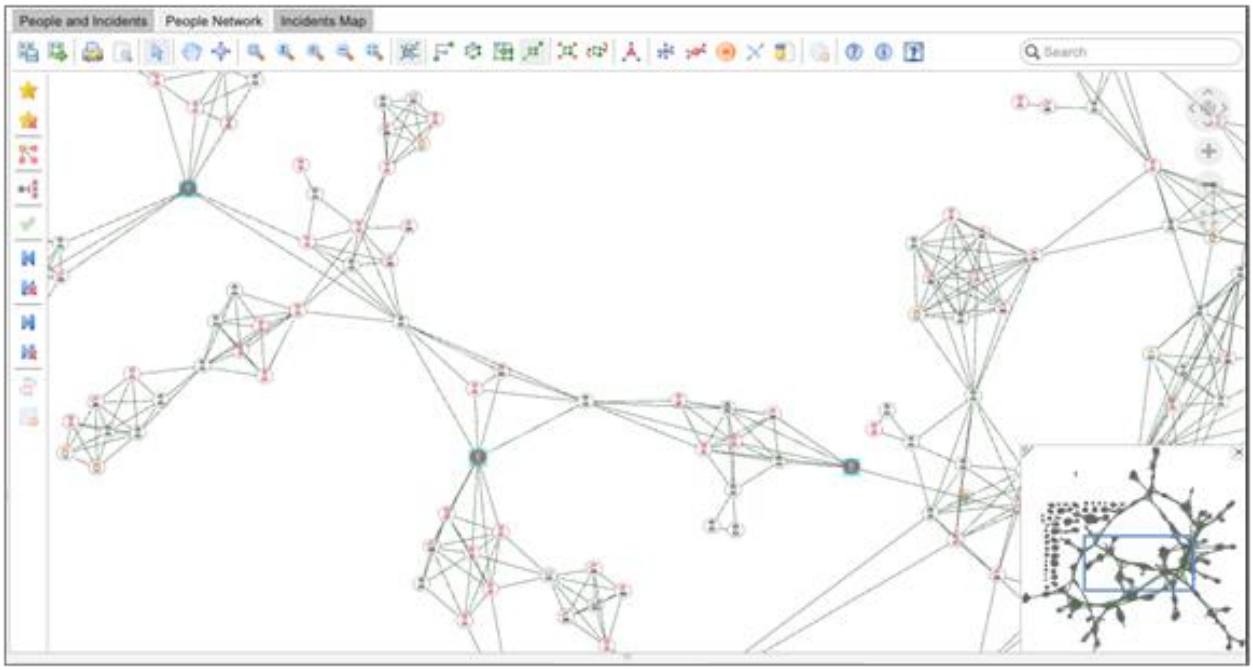


Figure 4-8: Social network marked with specific agents of interest.

In Figure 4-9, a reachability analysis is applied to focus in on the network near these key agents.

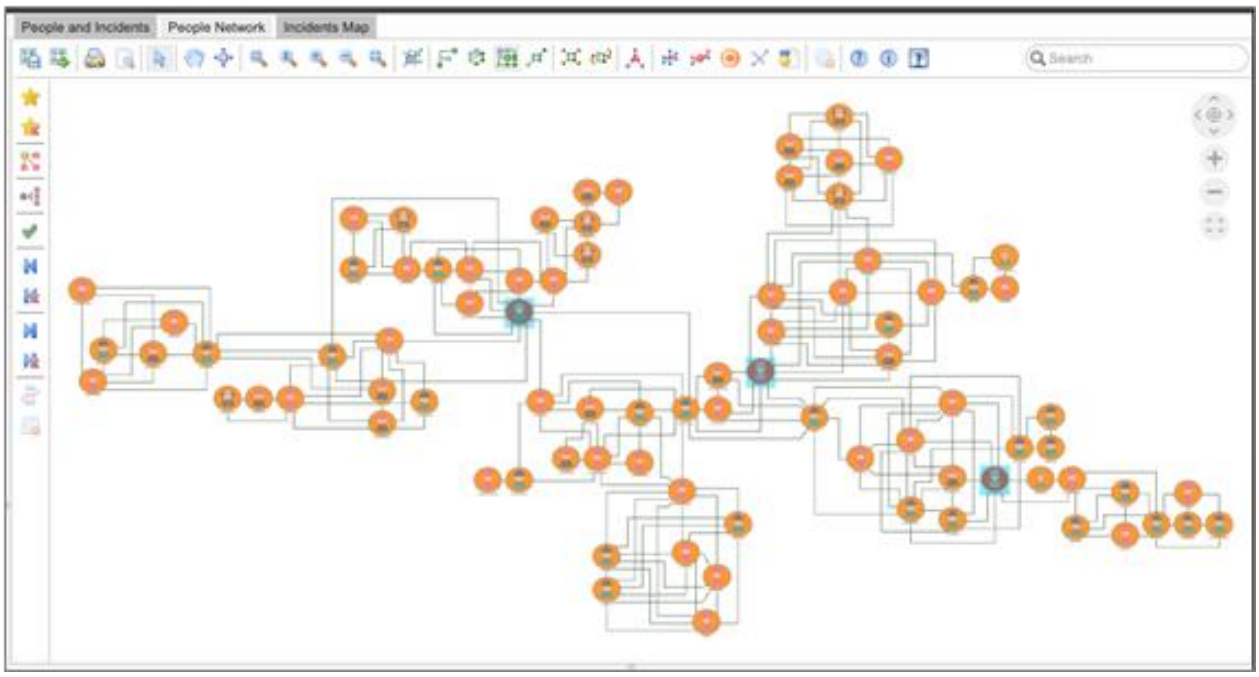


Figure 4-9: Reachability analysis on key agents.

Then, betweenness centrality analysis is applied to discover key agents in this sub-network, as illustrated in Figure 4-10.

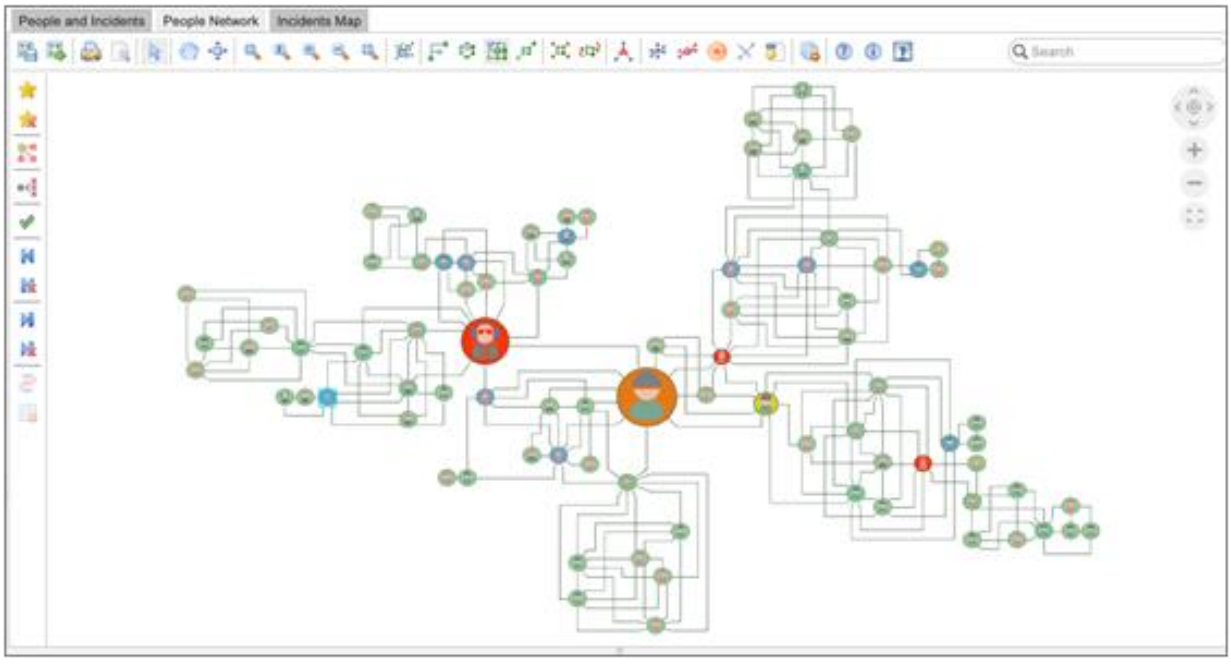


Figure 4-10: Betweenness centrality analysis on a connected social network of agents.

There are different types of centrality analyses, and betweenness centrality shows agents that are on especially active pathways of communication within the sub-network. The result is that the key agents found by the analysts were used as input to data-driven analyses which led to

finding the leader of the sub-network. With this approach, it is possible to discover key characteristics of the network that would be difficult to find through automated analytics alone.

Figure 4-11 shows this sub-network in relation to the surrounding network.

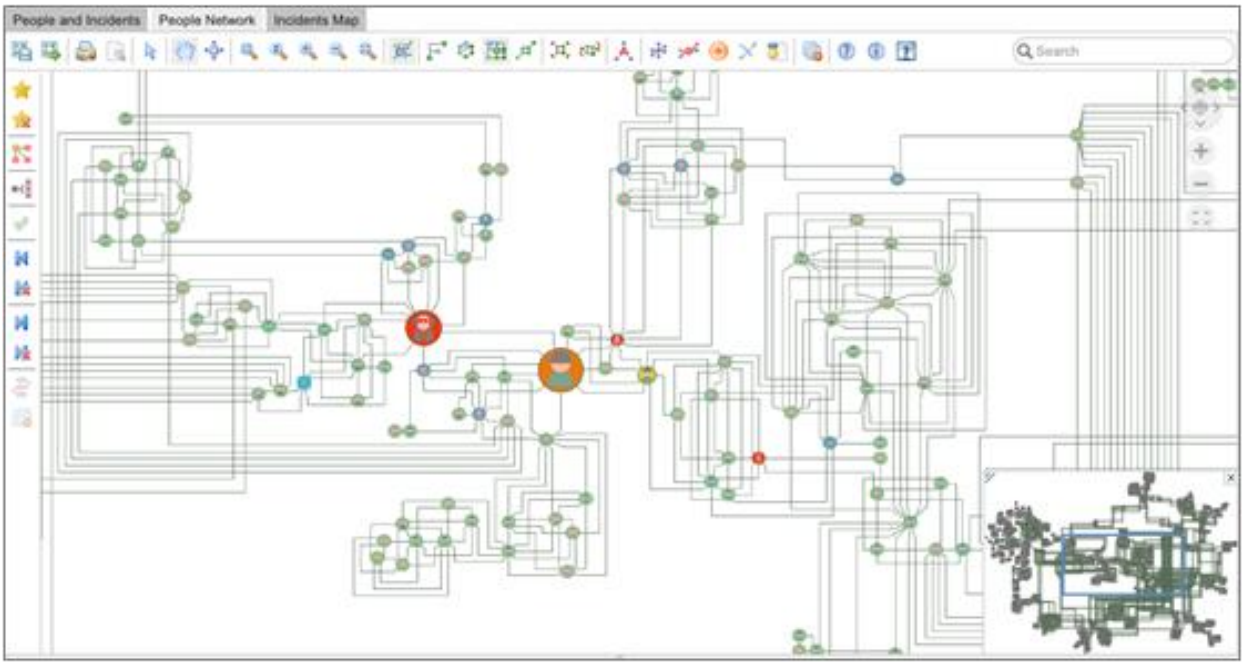


Figure 4-11: The network of interest with expanded network context.

4.3 Navigating in a System-of-Systems Knowledge Graph

Whether you are a system engineer or not, the environment you work in is itself a system of systems. Engineered systems are all around us. ISO/IEC/IEEE 21839 (ISO, 2019) provides a definition of Systems of Systems and constituent systems:

- **System of Systems (SoS)** — *Set of systems or system elements that interact to provide a unique capability that none of the constituent systems can accomplish on its own.*
- **Constituent Systems** — *Constituent systems can be part of one or more SoS. Note: Each constituent is a useful system by itself, having its own development, management goals and resources, but interacts within the SoS to provide the unique capability of the SoS.*

Many people are tasked with monitoring a system or understanding the design or behavior of a specific system. Interacting with the instance of a system, be it a procedure, or a physical system, such as a hot water distribution system, or an electrical distribution grid, typically requires new types of interaction than those that were required to model it. Figure 4-12 shows an example of a typical roof-top air handling system as found in many commercial buildings in North America and elsewhere, rendered in the style typical for this discipline. [19] The heating and ventilation control system manages the flow of water through the respective valves, while the respective water systems control the availability of hot or chilled water to this and other HVAC units. These

systems overlap where the heating and cooling coils are present within the supply air flow. Each distribution system (air, hot water, chilled water) operates independently.

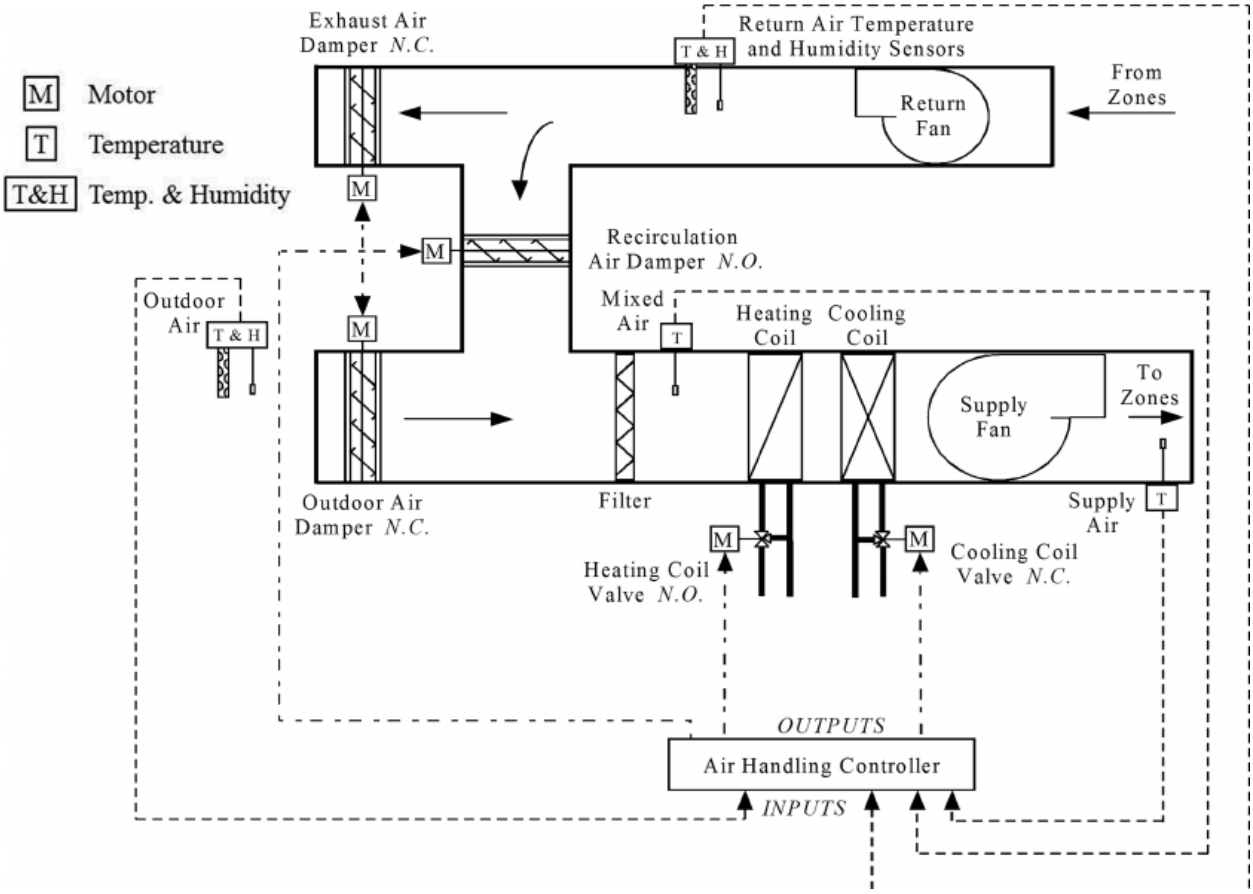


Figure 4-12: Exemplar schematic of an industrial air handling unit showing the air flow chambers, the position of heating and cooling coils, and associated control valves and sensors. [19]

Facilities staff will typically interact with these systems using a commercial control systems interface. The interface shown in Figure 4-13 [20] is an example of such a control display. These displays approximate the configuration of air systems but typically are not true digital twins with respect to geometry. They follow similar layout rules as the mechanical diagrams, but use a rich presentation style, and support interaction with the elements being controlled.

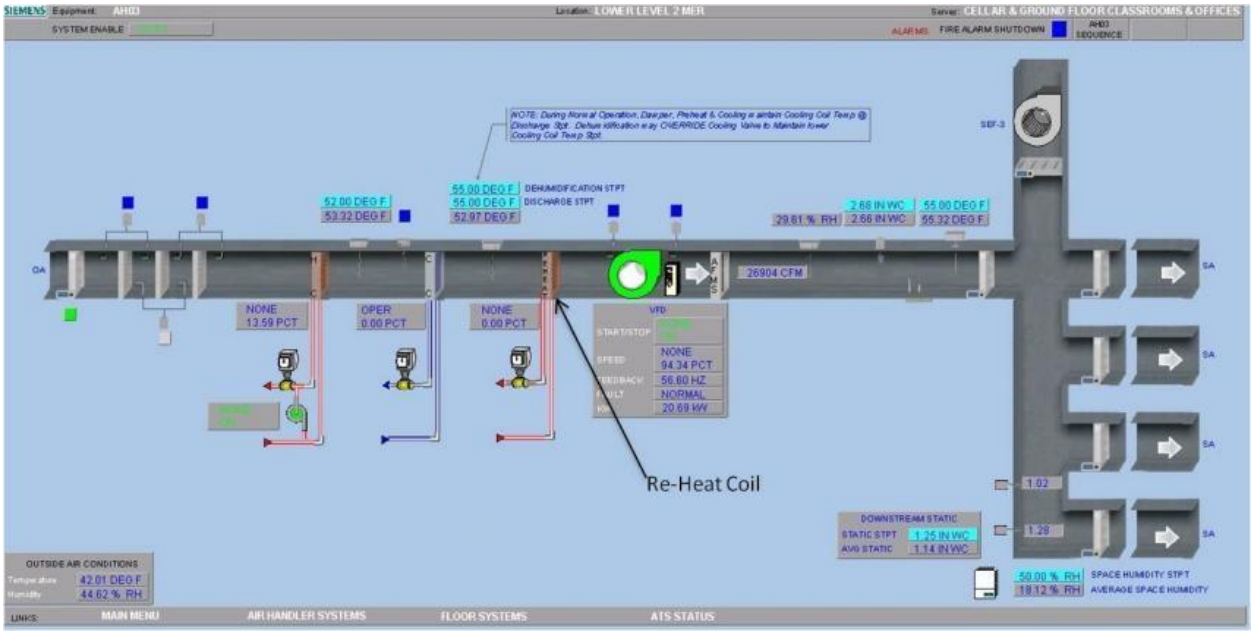


Figure 4-13: Siemens control screen for an Air Handling Unit showing the assembly of heating, cooling, and flow control devices within the air flow chamber. [20]

The automatically generated system graph in Figure 4-14 represents an example instance model of a heating, ventilation, and air conditioning (HVAC) unit, based on elements defined by the IFC meta-model described in section 4.1. This generated graph layout conforms to domain expectations for arranging the connections between the related air distribution loop, chilled water distribution loop, and the hot water system similar to the example in Figure 4-13, using a simpler style. The portions of the air flow system are aligned, as they would be in an air chamber, and the cooling and heating coils are positioned relative to other components in the system.

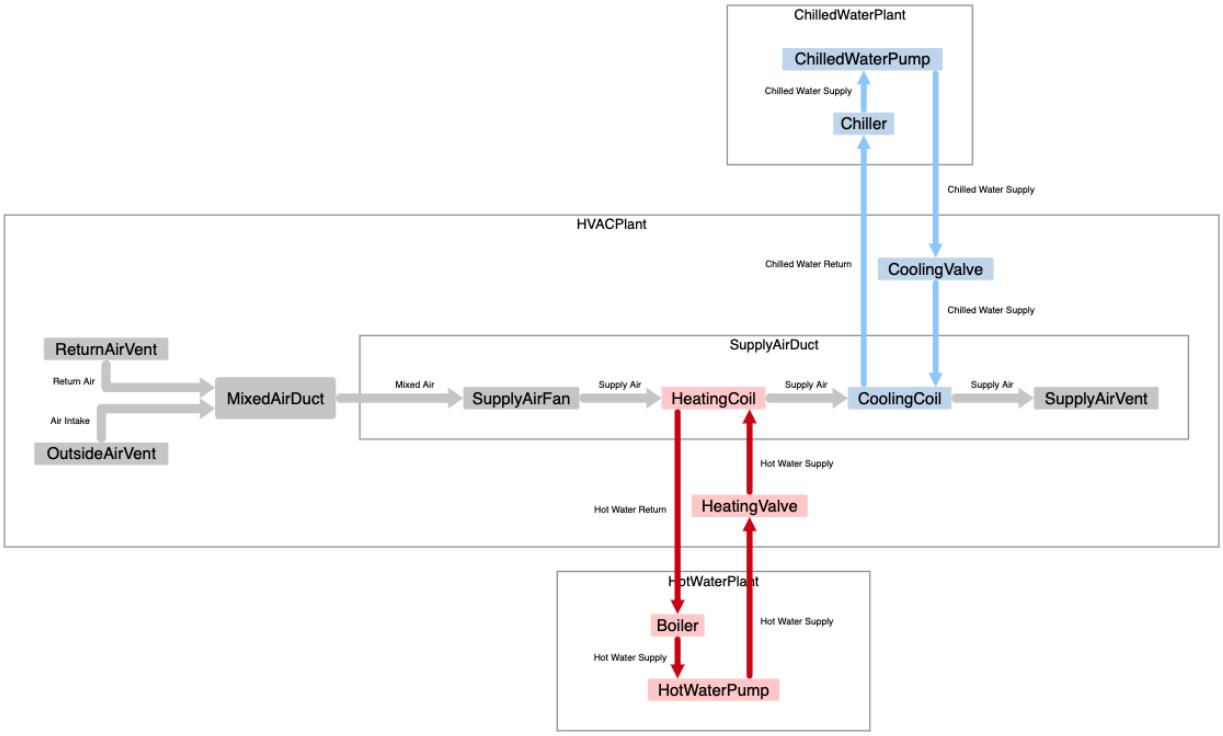


Figure 4-14: Automatically generated graph layout of an example HVAC system.

As stated in section 4.1, the type of visualization and navigation required to work with instance graphs that represent real systems is typically very different from what is required to model them. End users in these systems require only as much context as is needed to explain or explore the instance in a manner suited to the user's task. Often, the task is to understand the run-time conditions of the systems in a specific facility. In the case of our example, the instance graph of the system can be described by the simple schema in Figure 4-15. The details, such as the type and connection information supplied by the IFC meta-model have become attributes on the nodes and edges in the instance graph.

The schema that underlies the visualization in Figure 4-14 is shown in Figure 4-15.

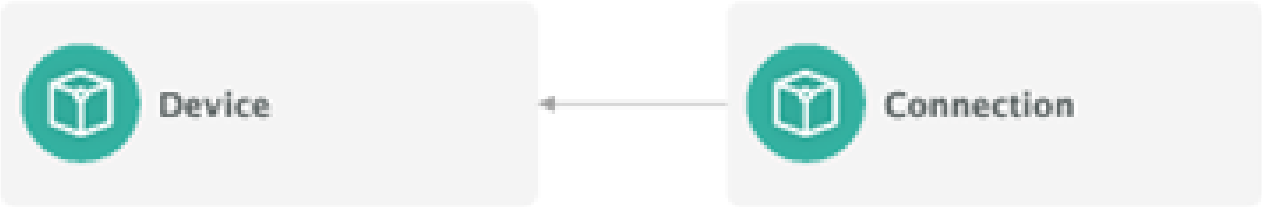


Figure 4-15: Simple schema for a directed system graph.

Domain-aware visualization of the schema relies on the domain details provided to populate information about each device. This simple model of devices connected by directed edges is a

specific projection of the ontology that informed it. The parameters associated with each device include information about the location of devices in a system of assemblies, where each assembly is a device comprised of other devices. The connections include both material flows (e.g., Hot Water) as well as containment (part of) relationships. It is possible to automatically generate arbitrarily detailed diagrams of real-world systems, so long as the required meta-data, and tools for visualization (e.g., suitable icons) are defined. Devices in the chain can be represented by icons or pictures rather than simple boxes.

Figure 4–16 illustrates the simple model with simulated run-time data. The connections can now be further enhanced using the data about material flow, the status of valves and devices, as well as other instruments such as temperature and position controls. In this example, the heating valve is closed (0%) and no hot water is flowing to the heating coil. The cooling valve is open, and the pump is operating, so flow is enabled. Similarly, the Supply Air Fan is operating, and air is flowing over the coils. At runtime, active flows can be animated for faster comprehension.

The graph in Figure 4-16 is interactive. Portions can be collapsed or expanded as needed for comprehension. In a typical control system, information about the status of a pump might not be displayed on the same screen as the schematic of an air handler, because the pump is controlled by another system. Using generated graph layouts, we can provide faster access to information about the pump, enabling the user to expand or collapse the Chilled Water System and Hot Water System to get more detail as needed without changing focus away from the Air Handling System.

Most control systems still consist of manually crafted screens that tie the control system (sensors, status indicators and actuators) to visual components. Typically, this level of system information is only available to the building operators. Definitive data measuring the impact that more flexible system displays deliver for facility operators has not been collected.

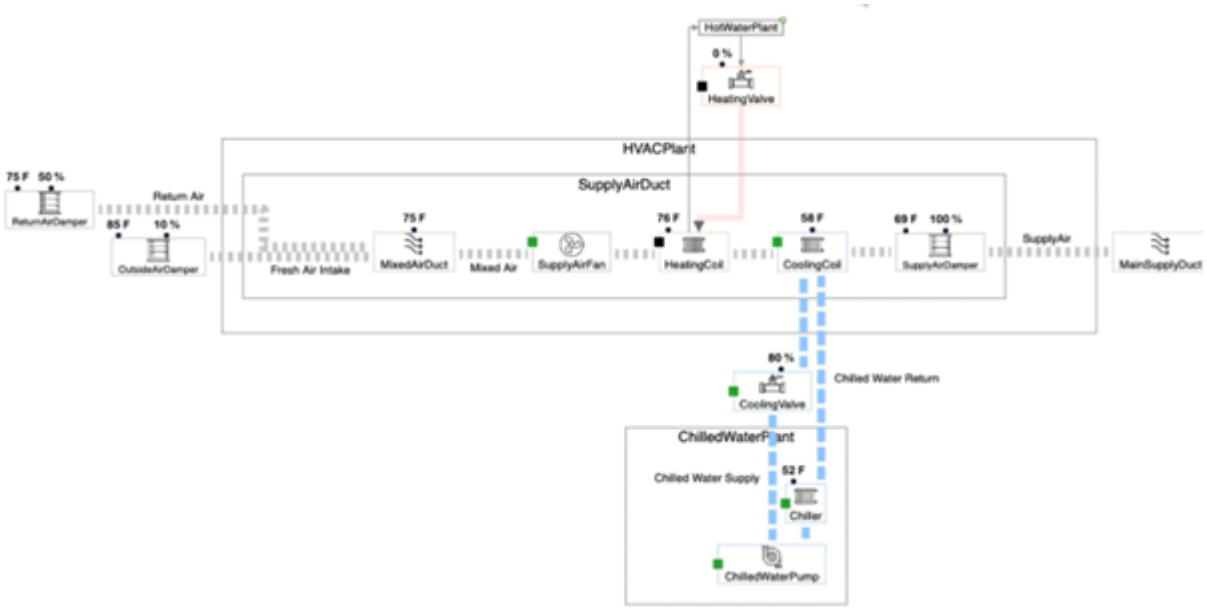


Figure 4-16: Model showing simulated real-time monitoring data.

Control system manufacturers, installers, and facility managers might not adopt automated layouts anytime soon. However, the ability to generate system graphs is useful for analysts to understand system configuration, and for the latest approaches at data-driven optimization to work with improved contextual information about the physical systems. Navigating these graphs involves understanding the flow of material, and graphs naturally support navigation along these connections.

Distribution systems can be arbitrarily complex. Graph technology is fit for the challenges of representing these systems effectively for machine-based analyses of systems, as well as human interaction and comprehension. Effective system visualizations facilitate understanding across different disciplines, explicitly illustrate shared context at appropriate layers of abstraction, and support informed decision making in multi-disciplinary problem spaces. Graph-based illustrations inform people about critical processes that may have direct impact within another context with which they are less familiar.

Today, most commercial distribution systems are designed by one supplier, installed by another, and instrumented by one or more other service providers. System monitoring diagrams are manually developed, for the most part. The generation of the knowledge graph for digital twins like those represented here is usually a non-trivial, and unrepeatable exercise when working with the typical engineering and implementation artifacts, since these consistently lack standard contextual meta-data. Industry standards like IFC/BIM and related design tools and processes are moving us toward more standardized, model-driven solutions; however, industry practices are slow to change. [21]

Data systems are very much like physical distribution systems, with clear elements to monitor the flow of information from one context to another, sensors to understand the nature or volume

of that flow, and limits placed on where those flows can be delivered. Data systems may or may not have the required contextual consistency that is lacking in the design of many physical systems. Digital thread engineering tends to be fraught with challenges similar to those that are found in physical engineering systems. [18]

Digital threads are designed to describe and orchestrate data management and machine interaction across the many layers of concern and through multiple translations to new contexts and throughout a product lifecycle. [15] Every system or subsystem, and every digital thread connecting them, are subgraphs in a System of Systems (SoS) that describe the network of information and how it is changing over time. Using graph-based visualization to make the paths between raw data sources, where and how they are transferred and translated to other systems, and audit or trace the paths that data takes, may make it easier to validate digital processes, build trust, and uncover discrepancies.

5 CONCLUSION

Filipov et. al., asked the question, “Are We There Yet?” with respect to the current state of network visualization research and concluded, unequivocally, “For the last time, no… and stop asking.” [6] The white space for further research in this area is vast.

The tools with which to manipulate graphs for analytics, display, and interaction are growing in power, capability, and complexity. However, it is clear from reviewing the available research that empirical studies of cognitive and other human aspects of working with ever-more complicated graph-based data environments hasn’t kept up with the state-of-the-art of the commercially available technology. It is possible to produce ever more complex visualizations, but there are few empirical measures of their effectiveness with real-world examples for the people who are tasked to use them.

As solution providers, this means that we will continue to expand what we make available based on what application developers, designers, and end users demand from us. As new human factors research is published, we will know more about when and where to use specific techniques. Until then, we will continue to develop new graph-based visual interaction applications, for and with our partners.

6 REFERENCES

[1] V. Yoghourdjian, D. Archambault, S. Diehl, T. Dwyer, K. Klein, H. Purchase and H.-Y. Wu, "Exploring the Limits of Complexity: A survey of empirical studies on graph visualization," *Visual Informatics,* pp. 264-282, 2018.

[2] M. Burch, W. Huang, M. Wakefield, H. C. Purchase, D. Weiskopf and J. Hua, "The State of the Art in Empirical User Evaluation of Graph Visualizations," vol. 9, 2021.

[3] H.-J. Schultz and H. Schumann, "Visualizing Graphs - A Generalized View," 2006.

[4] H. Li, G. Appleby, C. D. Brumar, R. Chang and A. Suh, "Knowledge Graphs in Practice: Characterizing their Users, Challenges, and Visualization Opportunities," vol. 30, no. 1, pp. 584-594, Jan 2024.

[5] P. Bullemer, D. V. Reising, C. Burns, J. Hajdukiewicz and J. Andrzejewski, Effective Operator Display Design, Abnormal Situation Management Joint R&D Consortium, 2008.

[6] V. Filipov, A. Arleo and S. Miksch, "Are We There Yet? A Roadmap of Network Visualization from Surveys to Task Taxonomies," vol. 42, no. 6, 2023.

[7] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. van Wijk, J. Fekete and D. Fellner, "Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges," *Computer Graphics Forum Volume 30, Issue 6,* pp. 1719-1749, 2011.

[8] M. Bajaj, S. Friedenthal and E. Seidewitz, "Systems Modeling Language (SysML v2) Support for Digital Engineering," *Insight,* vol. 25, no. 1, pp. 19-24, 2022.

[9] M. Alenazi, N. Niu and J. Savolainen, "SysML Modeling Mistakes and Their Impacts on Requirements," in *IEEE 27th International Requirements Engineering Conference Workshops (REW)*, Jeju, Korea (South), 2019.

[10] J. Grudin, J. Hahn and H. Hahn, "How Do We Understand a System with (So) Many Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning," *Information Systems Research,* vol. 11, no. 3, pp. 284-303, 2000.

[11] J. Holt and S. Perry, SysML for Systems Engineering, Vols. ISBN 978-1-78561-555-9 , IET Professional Applications of Computing Series 20, 2018.

[12] S. Friedenthal and E. Seidewitz, "A Preview of the Next Generation System Modeling Language (SysML v2)," *PPI SyEn,* no. 95, pp. 6-22, 27 November 2020.

[13] M. Bajaj, J. Backhaus, T. Walden, M. Waikar, D. Zwemer and C. Schreiber, "Graph-Based Digital Blueprint for Model Based Engineering of Complex Systems," in *Annual INCOSE International Symposium (IS 2017)*, Adelaide, Australia, 2017.

[14] M. Lima, Visual Complexity - Mapping Patterns of Information, New York, NY: Princeton Architectural Press, 2011.

[15] AIAA Digital Engineering Integration Committee, "Digital Thread: Definition, Value and Reference Model," 2023.

[16] J. M. Six and I. G. Tollis, "A Framework and Algorithms for Circular Drawings of Graphs," *Journal of Discrete Algorithms,* vol. 4, no. 1, pp. 25-50, 2006.

[17] buildingSMART International, *https://www.buildingsmart.org/*

[18] B. Bailey, "Data Coherence Across Silos and Hierarchy," 27 June 2024. [Online]

https://semiengineering.com/data-coherence-across-silos-and-hierarchy/

[Accessed 20 October 2024].

[19] N. M. Ferretti, M. Galler, S. T. Bushby and D. Choinière, "Evaluating the Performance of DABO and HVAC-Cx Commissioning Tools using the Virtual Cybernetic Building Testbed," *Science and Technology for the Built Environment,* vol. 21, no. 8, 2015.

[20] M. Baglione, "Building Sustainability into Control Systems - Air Handling Units," The Cooper Union, 2012. [Online]. *https://engfac.cooper.edu/melody/417*
[Accessed 12 12 2024]

[21] L. Kiff, K. Johnson, M. Raymond, T. Plocher, T. Napier, B. Brucker, J. Bush, D. Schwenk and M. Carrasquillo, "Model Driven Energy Intelligence," Arlington, VA, 2012.

ACKNOWLEDGEMENTS

The views expressed in the *OMG Journal of Innovation* are the author's views and do not necessarily represent the views of their respective employers nor those of the Object Management Group® (OMG®).

- Return to the beginning of this article
- Return to the Table of Contents