MBSE-driven Visualization of Requirements Allocation and Traceability

Maddalena Jackson
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-0319
mjackson@jpl.nasa.gov

Marcus Wilkerson
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-3487
Marcus.Wilkerson@jpl.nasa.gov

Abstract—**In a Model Based Systems Engineering (MBSE) infusion effort, there is a usually a concerted effort to define the information architecture, ontologies, and patterns that drive the construction and architecture of MBSE models, but less attention is given to the logical follow-on of that effort: how to practically leverage the resulting semantic richness of a well-formed populated model to enable systems engineers to work more effectively, as MBSE promises.**

While ontologies and patterns are absolutely necessary, an MBSE effort must also design and provide practical demonstration of value (through human-understandable representations of model data that address stakeholder concerns) or it will not succeed. This paper will discuss opportunities that exist for visualization in making the richness of a well-formed model accessible to stakeholders, specifically stakeholders who rely on the model for their day-to-day work. This paper will discuss the value added by MBSE-driven visualizations in the context of a small case study of interactive visualizations created and used on NASA's proposed Europa Mission. The case study visualizations were created for the purpose of understanding and exploring targeted aspects of requirements flow, allocation, and comparing the structure of that flow-down to a conceptual project decomposition. The work presented in this paper is an example of a product that leverages the richness and formalisms of our knowledge representation while also responding to the quality attributes SEs care about.

TABLE OF CONTENTS

1. INTRODUCTION 1
2. VISUALIZATION AND SYSTEMS ENGINEERING 2
3. CASE STUDIES: INTRODUCTION 3
4. CASE STUDY 1: CONCEPT HIERARCHY 4
5. CASE STUDY 2: CONSTRAINT ALLOCATION 7
6. CASE STUDY 3: HIERARCHY COMPARISON 9
7. CASE STUDY 4: REQUIREMENTS TRACEABILITY 10
8. TECHNOLOGIES 11
9. CONCLUSIONS 12
10. FINAL RECOMMENDATIONS 13
APPENDIX 14
ACKNOWLEDGMENTS 17
REFERENCES 17
BIOGRAPHY 17

978-1-4673-7676-1/16/$31.00 ©2016 IEEE

1. INTRODUCTION

This paper discusses interactive visualizations of Systems Engineering (SE) data developed and use in the context of MBSE practiced on NASA's proposed Europa Mission. We argue that development of visualizations specifically, and prioritization of facilitating SE interaction with model data more generally, deserves more attention in MBSE infusion efforts. We will use the example of visualizations developed for the Europa project to motivate and ground our assertion. In particular, we will discuss interactive visualizations created to provide insight and visibility into the requirements development process for the proposed Europa mission.

With the selection of the Europa Mission for concept and technology development in June of 2015, the mission became the first large-scale flagship project at the Jet Propulsion Laboratory (JPL) to fully adopt an MBSE approach starting at formulation, endorsed by the top levels of the project, and intended to support all of our engineering activities or at least drive them throughout the entire mission [1]. The Europa project faces the challenge of all early adopters - being on the cutting edge, building infrastructure as we use it, developing strategies to do the work we have traditionally done using new model-based techniques and methodologies. We are not participating in a shadow effort, tech demo, or pilot; we have crossed a tipping point and now face a rapid, ongoing adaptation process.

Of interest here, out of the full scope of MBSE activities for the Europa project [2], is that our requirements derivation process is occurring in the model. Our requirements, constraints, rationale, verification plans, traceability, and links to conceptual design are in the model and must now be implemented. Our application of visualization is in the domain of supporting the SE requirements derivation process and reasoning about it through interactive visualizations.

SEs on the Europa project are provided multiple ways to interact with the model depending on their level of comfort with tools or personal preference: through direct use of our MBSE tool, through a web interface presenting a predefined set of editable views into the model data, or both. Consequently, SEs at the project, flight system, and ground system level are actually working with the system model as the Single Source of Truth (SSoT) for the mission. This has already provided many benefits, such as eliminating the need to manage and coordinate the distribution and relationships between siloed latest data. However, it has also revealed some gaps that are not explicitly discussed in our MBSE visions. Specifically, the underlying knowledge that there are good reasons why SEs have traditionally chosen to work with spreadsheets and viewgraphs, reasons that can cause problems for a large-scale

MBSE effort when those use cases are not accounted for in our infrastructure.

The challenge

Our central argument is that successful MBSE infusion means adoption by projects, and our efforts must respond equally to the needs of practicing systems engineers and to the abstract correctness of knowledge representation or they may be abandoned.

Adopting MBSE requires the development of rules for capturing necessary project and design data in a SSoT. Our traditional document-based approach to information management allows people to capture truth as they need it and is most practical, which is why spreadsheets are so appealing. This is effective for individuals, but presents problems if we wish to leverage any advanced, machine-based reasoning or algorithms to assist checks of completeness, correctness, integrity, metrics, simulations, etc. In adopting MBSE, there is a role/job that is usually performed by an ontologist or architect, which is to make sure that the SSoT rules are constructed so that the schema for knowledge representation meets everyone's needs. A test for adequacy of the result would entail the capability to extract each person's data model from the system model.

The challenge is that for MBSE to be effective for SEs, it must support the kinds of analysis, application of personal experience, engineering judgment, improving and checking the system by assessing by different slices, conversation, negotiation, that was traditionally available through ad-hoc spreadsheets and tools.

2. Visualization and Systems Engineering

When we talk about visualizations in this paper, we mean representations of data that viewers digest and analyze with their eyes. Visualizations (as opposed to viewgraphs) speak intuitively, but are driven by rich semantics. Visualizations, at the most obvious and basic, are visual representations of data to reinforce, assist, and expand human understanding and perception. They allow us to see, explore, and understand large amounts of information of various types at once. Interactive visualizations capitalize on this rapid, intuitive, pattern-based communication, and the insight obtained from interacting with the data is now limited only by our curiosity and creativity.

SE is an inherently creative discipline. It is also questioning, cross-cutting, innovative, skeptical, broad, detailed, and exploratory. SEs have the job of managing, defining, and designing a system, and problem solving happens through trades, negotiation and compromise, all of which require the correct balance of depth and breadth from the SE. A core competency of Systems Engineering is the ability to assess domain areas from an integrated perspective: from the perspectives of many stakeholders with different interests and concerns. SEs must be adept at making connections between facts, systems, domains, behaviors, concerns, events in time, cause and effect, processes, concepts, and realizations, because this is where system gaps are found and resolved. To do this, systems engineers develop good mental models, which help us understand, connect, remember, explain, and iterate in solving the engineering problems we are faced with.

Visualizations for Systems Engineering

Visualization is effective for Systems Engineering because it can both augment our mental models and can assist in communicating the model and model-based insights to others. Much of Systems Engineering is communication and negotiation, so having clear and accessible ways for SEs to exchange information and insight makes the discipline more efficient. Interactive visualizations or interactive tools in general thus lend themselves well to the inherent need of SEs to reorganize, group, filter, and creatively manipulate the information. Like a spreadsheet, interactive visualizations provide the ability to manipulate the data to their liking, but unlike a spreadsheet, that data conforms to ontological rules, so the SE is inherently asking questions within the framework.

To best target visualization for SE, it is useful to look at how SEs interact with data in the current (document-based) paradigm. This is a valuable exercise because our record of successful and increasingly complex missions indicates that what we are doing now works and is effective. Our goal, however, is to further *improve* the effectiveness of SE, so transition to MBSE must least support existing capabilities if it aims to then improve upon them.

In traditional Systems Engineering, SEs build their own ways to manage and manipulate data to answer questions, support meetings, reviews, exploration, and learning, often through use of spreadsheets, slide decks, or diagrams. In spreadsheets, we use categorization and tagging so that we can do filtering and organizing. SEs also write emails, ask questions, draw on whiteboards, etc. This is a very organic and individualized tool ecosystem. Visualizations, when carefully crafted, provide a very useful balance between allowing the SE to have creative and exploratory access to the data while maintaining the robust and rigorous structure that encodes meaning.

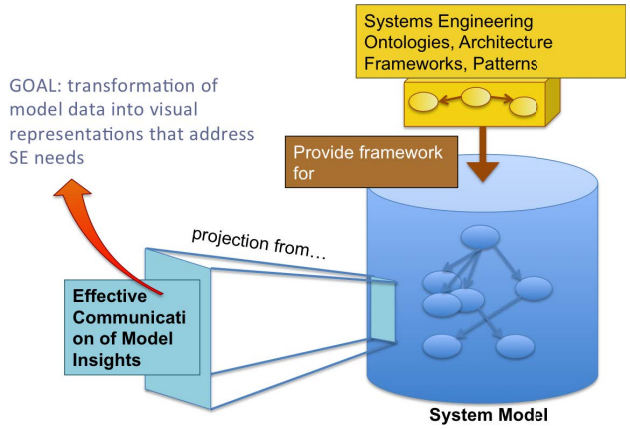

Figure 1. Frameworks provide the rules for populating and analyzing system models.

Model-driven Visualizations

The system model is a repository containing objects, properties, values, and relationships to other objects. Our ontologies set out definitions of different objects and relationships and the rules for what is allowed to connect to what. We populate our models through an act of translation and transformation: mental SE models (and traditional artifacts) become objects and relationships that obey the rules of the frameworks. This process requires discussion, explanation, and negotiation be-

tween experienced SEs, modelers, architects, and ontologists until the transformation becomes familiar to all. From a populated model we can extract subsets of the data, projecting views from the model to generate key products and analyses.

At JPL, we have created a very strong core of infrastructure for configuring and generating these view projections and with them have made significant progress in breaking down the barriers between MBSE and traditional documents: with DocGen, we can produce static model-based artifacts, and with View Editor, we have a web-based read/write capability for model data [3][4][5]. Neither of these capabilities, however, yet addresses the SE need to dynamically explore and manipulate model data. This is where our visualizations for the Europa project come in.

3. Case Studies: Introduction

We will describe four interactive visualizations that we created to address specific concerns in the design and requirements derivation process for the Europa project:

Table 1. Case Study Description

Case Study	Description
CS1: Concept Hierarchy	Functional/logical decomposition of technical areas on Europa
CS2: Constraint Allocation	Allocation of constraints between technical areas
CS3: Hierarchy Comparison	Comparison of an asserted decomposition to an inferred decomposition
CS4: Traceability	Full requirements traceability

To explain the value our visualizations added to the mission, we need to provide a brief overview of the requirements derivation process for the proposed Europa mission. Design activities for the project, including requirements derivation, are done through formal methodology that, unfortunately, cannot be given full justice here [6]. The salient feature of that methodology for the purpose of this paper is that we maintain a strict separation between the needs of our mission and the actual design. The conceptual needs drive the design, but stay separate, with formal binding requirements originating from the conceptual needs and acting as an interface to the real design. Our requirements derivation process is the development and elaboration of those conceptual needs and the mapping of those conceptual needs to formal requirements, and is notionally depicted in Figure 2.

Early in the mission, a group of architects and experienced engineers identified functional domain areas that, taken together, form the complete set of perspectives and technical areas of concern from which requirements, architectural decisions, and design can logically originate. These functional domain areas are called "Concepts" and a responsible engineer and modeler are assigned to develop and model the content of each one.

The concepts relate to each other through their place in a hierarchy (an example of which is shown in Figure 3) that was informed by functional and logical decomposition and engineering judgment based on knowledge and tradition of previous missions. The purpose of the hierarchy is to structure and provide scope for the conceptual elaboration

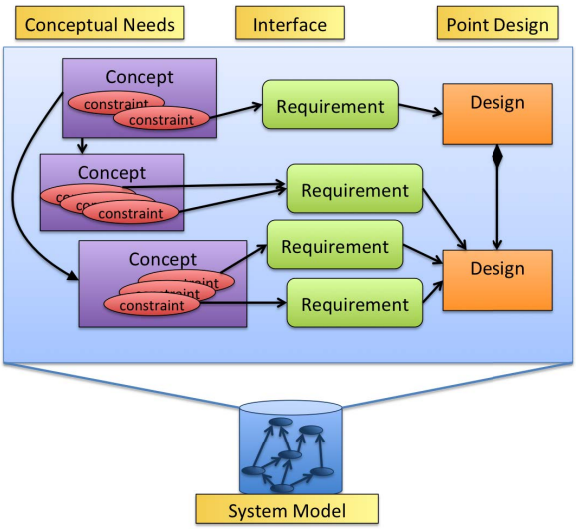

Figure 2. The role of requirements in separating concept and realized design.

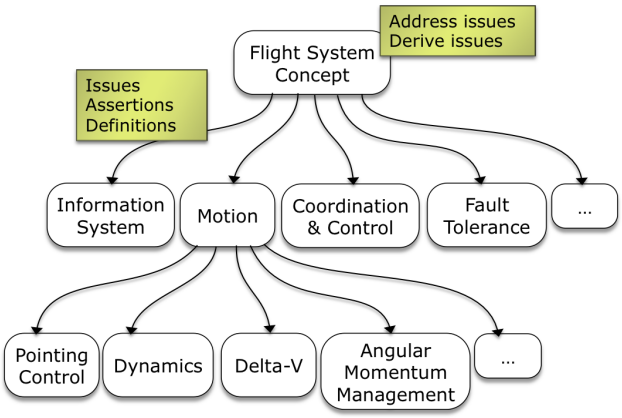

Figure 3. Notional concept hierarchy.

and refinement of high-level concerns and constraints, such as mission success criteria, the strategy for conducting observations of Europa, and operability, into lower level concerns such as temperature control of the flight system, mass allocation, and ground system infrastructure.

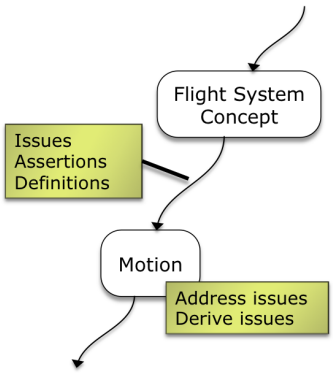

Figure 4. Parent and child concepts.

The job of the responsible engineer ("concept lead") is to explain, within the scope of their concept, the Europa

Table 2. Stakeholder concerns

Stakeholder	Questions	Solution
Concept Lead	• Where is my concept situated in the hierarchy? • Who are my parent and child views? • What functional areas influence me? • What functional areas do I influence?	• Concept ping • Concept trace, highlight, and filter
Systems Engineer	• What concepts influence each other? • What concepts have biggest impacts? • What concepts require a lot/a little work? • Does this hierarchy look complete?	• Highlight and filter • Bar sizes and line aggregation • Exploration and diagram relayout
Architect, Manager	• What is the state of concept development? • What areas need attention? • How is the status trending?	• Colorization and table

Project's approach to addressing the concerns, needs, and constraints on the mission specifically related to the technical domain of their concept. This is done through structured narrative that explains our response to incoming constraints and provides justification and rationale for the derivation of new constraints to be allocated to lower level concepts. Constraints are assertions made with respect to the conceptual needs that must be true of the ultimate design. This process of elaboration, derivation, and allocation is recursive throughout the hierarchy of concepts so that the traceability from detailed and low level decisions back to high level success criteria and mission goals is clear and navigable.

The concept lead works with their assigned modeler to capture the conceptual design, constraints, elements, and relationships in the system model. This rich engineering activity provides the context and justification for our requirements: the constraints identified in the concepts are assessed by the responsible engineers and requirements team, and projected into Requirements when there is a need to contractually bind a supplier to deliver an item that satisfies these requirements.

Flowdown and Traceability

Of specific interest to us in this paper is the aforementioned ability conferred upon a "parent" concept to levy constraints upon concepts directly below it in the hierarchy ("child" concepts), with the expectation that the child concept will describe our approach to satisfying them. Because this recursive elaboration drives the requirements derivation, we are concerned with understanding first the hierarchy, then the particular constraints that are allocated between concepts, and then the full traceability story of those constraints. It is this traceability, representing the result of our system functional decomposition that we are interested in visualizing, because we are now presented with a new, MBSE-based requirements derivation process, with new tools, and a rich dataset that is difficult to mentally track.

Visualization Overview

The visualizations discussed here follow the common format of an interactive web-based tool that takes data in the form of a graph of nodes and edges and renders the graph in a visually coherent way for manipulation and exploration by users. Our datasets and visualizations are large and intended to be explored in a working environment; consequently, we have included full-page versions of the smaller figures in the Appendix.

All of the visualizations discussed here are Sankey diagrams. Sankey diagrams are a form of flow diagram showing nodes and edges where the width of edges is proportional to amount of items flowing between the connected nodes. Our Sankey diagrams render directed acyclic graphs of nodes and edges. This is appropriate for the kind of data we are dealing with in concept hierarchies and requirements flow and traceability, where the data is acyclic, hierarchical, and directed, but not a tree like most common plotting tools expect.

The first case study, the interactive concept hierarchy explorer, was the first visualization developed and was the basis for the other visualizations discussed in this paper. The framework developed for the concept hierarchy visualization is generic and was easily customized and extended to our other examples.

4. Case Study 1: Concept Hierarchy

The first visualization we will discuss is a straightforward tool that lets us investigate and comprehend the full concept hierarchy in the Europa project. The usefulness of such a visualization is apparent when we become aware of the size of the dataset: there are approximately 150 concepts in the hierarchy. This immediately gives rise to some relevant questions, a few of which are categorized by stakeholder and presented in Table 2.

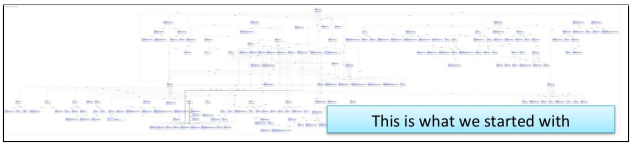

Figure 5. An original concept hierarchy visualization: "The Scroll."

Our Starting Point

The first visualization of the concept hierarchy was a simple static diagram of the concept hierarchy drawn in our MBSE tool and shown in Figure 5.

Problems with this are immediately obvious in that the reader of this paper cannot actually see anything other than a graph

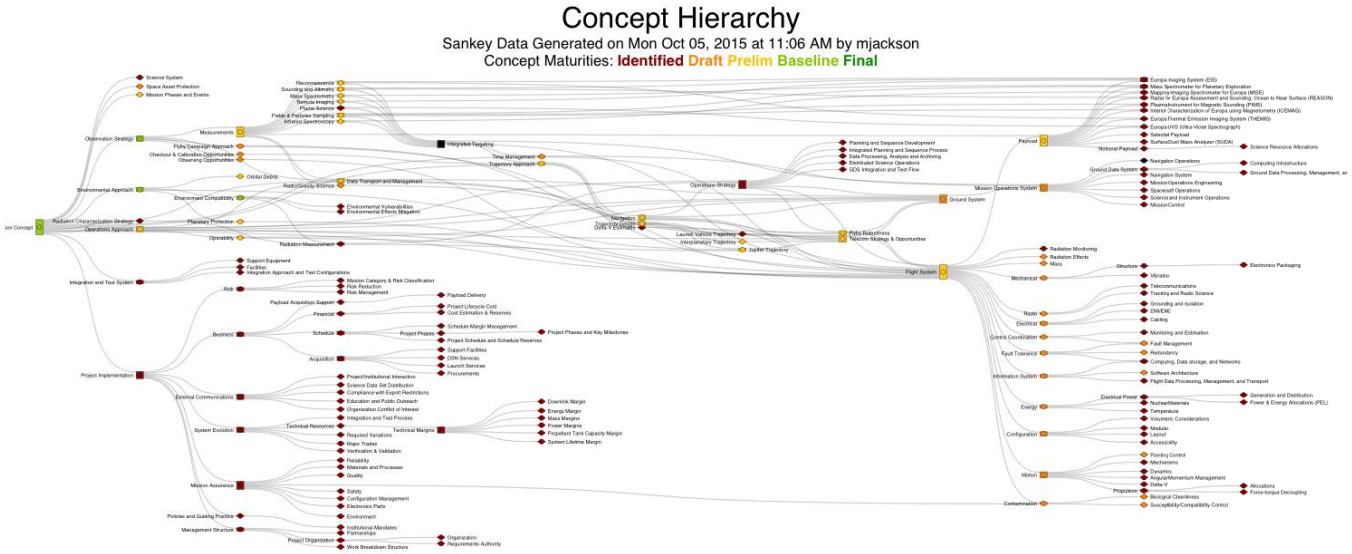

Figure 6. **Our solution: interactive Europa Project concept hierarchy.**

structure; the dataset is too large and spread out.

In practical use, this diagram was printed on a roughly 4ft long sheet of paper and unrolled on a conference table like a scroll. To use this diagram, someone had to a) know who had the print copy and borrow it then b) manually locate the concept of interest and finally c) pan and zoom around either with their finger to learn anything about related concepts. People who knew where to find the diagram in our MBSE tool faced the same issues, except on their monitor. In addition to these user interface issues, there were back-end problems: a) someone had to manually manage the layout of this diagram, b) someone had to ensure that the content was complete, and c) someone had to make sure that the printed-out "scroll" was the latest.

This was indeed a "visualization" of model data but not a particularly convenient one. It was frustrating and tedious to learn from, it could not be filtered or easily rearranged, and very little could be observed at a glance.

Our Approach: Interactive Graph Visualizations

We addressed the problems presented by static, difficult to navigate scrolls by creating a web-based interactive visualization of the concept hierarchy graph that helps address the concerns of different stakeholders.

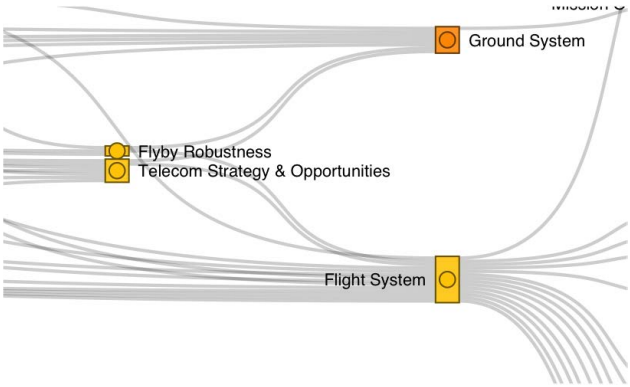

Figure 7. **Zoomed view of the concept hierarchy.**

In our visualization, shown in Figure 7, each concept is represented as a node connected to other nodes through grey lines. The diagram is read left-to-right (parent concepts are at left, children at right). The number of connected parents and children, whichever is greater, determines the height of each concept's node. Figure 7 is a small selection of the entire concept hierarchy (the entire hierarchy is extremely large, as seen in Figure 6). The lines on the left side of the "Flight System" node represent links incoming links from parent concepts, and the outgoing lines on the right side link to concepts for which FS is a parent.

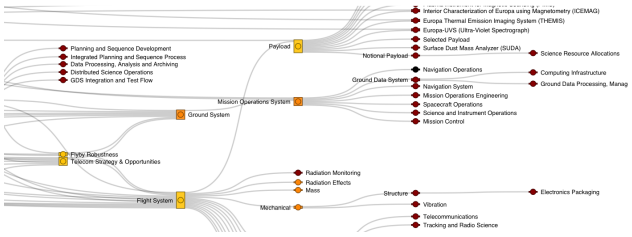

Figure 8. **Medium-zoom view of the concept hierarchy.**

The zoomed-out Figure 8 shows that the Flight System concept has many children, including Payload, Mechanical, and Radiation Monitoring. We can easily see at a glance what domain and concern areas are expected to influence our conceptual approach to designing a Flight System, and we can see what lower level concepts will be driven and bound by the decisions made in the Flight System concept. The entire concept hierarchy is shown in Figure 6.

From these figures the reader can see that the hierarchical graph is more legible than the inital visualization, but still quite complex. To address this, we have added some usability features to assist engineers in exploring and answering their questions. The layout of the visualization tool is shown in Figure 10. In addition to interacting with the visualization directly (clicking, dragging, mouse-overs, etc.), users can ping, filter, and modify some layout parameters through a control panel located directly beneath the visualization. Additional element information is displayed in a dynamic table below the display controls.

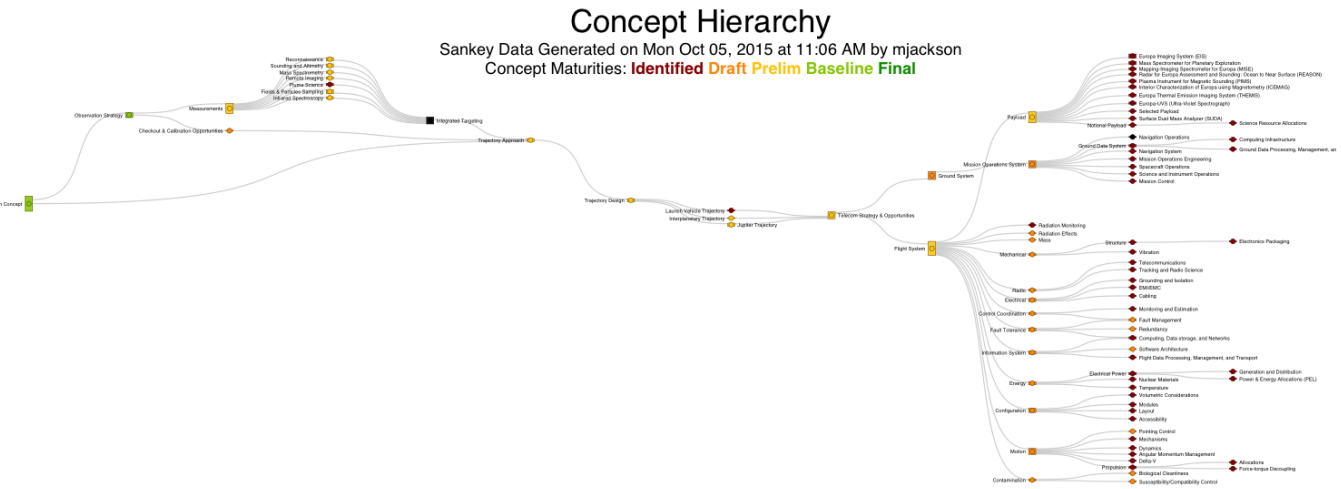

Figure 9. The concept hierarchy graph has been filtered to show only parents and children related (directly and transitively) to the Trajectory Approach concept.

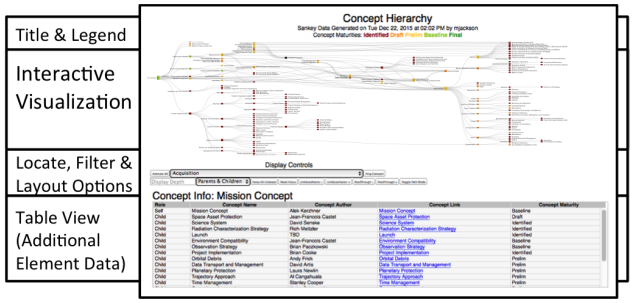

Figure 10. The visualization and interaction interface as presented to the user.

Highlighting

When the cursor hovers over a node, the incoming and outgoing links are highlighted in red so that the user can clearly see which nodes are directly connected to the one of interest. This is depicted in Figure 11.

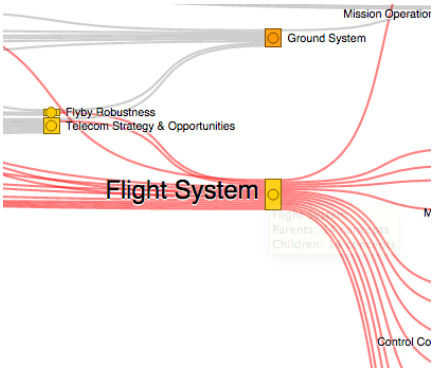

Figure 11. Highlighting: when the cursor is moved over a concept, the incoming and outgoing connections turn red.

Ping

To find specific concepts among the large number present in the system, we created a "ping" feature. The user selects the concept of interest from a dropdown list of all concepts (in the locate, filter, and layout options immediately below the diagram) and when the ping button is clicked, a location ping (red pulsating circle) appears at the location of the concept so that it can be quickly identified.

Node rearrangement and dragging

The visualization also supports the vertical rearrangement of nodes so that the user can lay out the diagram to their liking beyond the initial constraints of our layout algorithm. This aides comprehension by allowing users to reorganize the data according to their concerns. Dragging causes the links to other concepts to dynamically move (they are stretchy), so by grabbing a concept and moving it around, they can get an intuitive sense of the magnitude of its relationship to other concepts.

Filtering

The visualization also provides a mechanism to display only certain portions of the hierarchy. Currently this is limited to recursive display of parents and children based on selecting a concept of interest and choosing the depth of related nodes to show. In Figure 9, we selected the concept Trajectory Design and showed all of its parents and children.

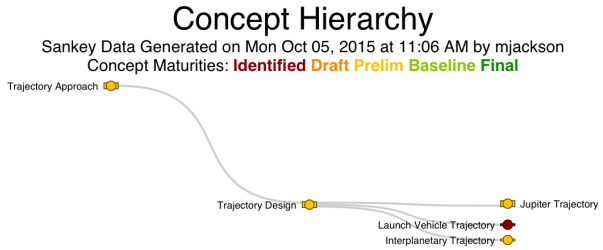

Figure 12. A filtered selection of hierarchy data showing only direct parents and children of the Trajectory Approach concept.

In Figure 12 we show the same concept's parents and children to a depth of one (i.e., direct parents and direct children).

Concept Info: Trajectory Design

Role	Concept Name	Concept Author	Concept Link	Concept Maturity
Self	Trajectory Design	Al Cangahuala	Trajectory Design	Prelim
Child	Jupiter Trajectory	Al Cangahuala	Jupiter Trajectory	Prelim
Child	Launch Vehicle Trajectory	Al Cangahuala	Launch Vehicle Trajectory	Identified
Child	Interplanetary Trajectory	Al Cangahuala	Interplanetary Trajectory	Prelim
Parent	Trajectory Approach	Al Cangahuala	Trajectory Approach	Prelim

Figure 13. **Concept-related data provided to users after clicking on a concept in the interactive diagram.**

Table

The last feature we will discuss is the table we provide below the locate, filter, and layout area. This table populates and updates its displayed data based on which nodes or edges are clicked in the diagram. The purpose of the table is to give users more information about the concept of interest and its related parents/children and is shown in Figure 13.

In the example above, we clicked on the Trajectory Design concept. The "role" column shows the concept in relation to the concept clicked that was selected. The concept name, the concept author (responsible engineer), a link to the model-generated narrative of the concept, and the maturity of the concept are also displayed. The maturity is also shown on the diagram via the color of the node [7].

About the Visualization

While the Concept Hierarchy visualization does not make use of the width-proportional edge capability found in most Sankey diagrams (each edge here has a weight of one), it does convey the magnitude of a concept's connections with its neighbors via the height of the concept node. We will see use of the proportional-width edges in the next case study.

Addressing SE Concerns

This visualization allows us to intuitively comprehend the nuances in how the concepts relate to each other. We can, by our visual inference, understand that a concept with many incoming connections and only a few outgoing connections is a concept that responds to the needs of a domain from many sources, and will probably have significant SE work to do in reconciling the different needs (which may overlap) and then crafting a reconciled approach.

When a concept has many children, but few parents, we can infer the opposite: that it is motivated by a small number of key functional areas that require the collaboration and union of many disciplines to ensure that the need is met.

When a concept has a large number of incoming and outgoing concepts, we can infer that this concept will be a central player, probably requiring extra systems engineering attention to make sure that the diverse source needs and large number of responding disciplines are coordinated effectively.

By looking at not just the direct parents and children but their relatives as well, we can begin to understand, analyze, and track the paths of influence that drive our system.

SE use on the proposed Europa Mission

This visualization supports our ongoing work as Concept Leads, Systems Engineers, and Managers. It is a living view of the latest data: it is fed by a projection of model data and ultimately can/will pull the data on page load. The definition of the hierarchy is authoritatively contained within our system model, and that data is automatically extracted and fed to this particular hierarchy explorer tool.

The images shown in this paper are exactly what SEs see in their day-to-day work on the Europa project and Europa SEs currently use it in a variety of contexts. On our project website (which is model-driven and where most SEs capture their work in this phase of the mission), this visualization may be accessed quickly from the top-level page, making it easy for SEs to find and refer to it.

This visualization added value by providing concept leads, systems engineers, managers with system context and state information that had been missing. The visualization is often shown in meetings to ground discussion; it is used by managers to check that the concept development work, the status of which is characterized at a high level by the maturity, is continuing as expected; the filtered views are commonly saved as snapshots and included in project and concept reviews to quickly orient the audience as to where the concept is in the larger conceptual and design effort.

It was difficult and tiring to internalize and draw conclusions from such a large network of interrelated information by inspecting the original static diagram, but with the interactive view, the information is more compact. The interactivity, highlighting, filtering, and table helps SEs modify the initial data presentation quickly to whatever communicates most effectively to that individual. This has resulted in increased data accessibility for a larger set of our SEs, managers, and other stakeholders.

5. Case Study 2: Constraint Allocation

The concept hierarchy shows us what is intended in terms of the influence and allocation of issues and constraints between concepts. The next visualization in our case study goes further, showing us not what we intended to do but what we *did* do: what constraints are actually allocated from parent to child. The full allocation graph is shown in Figure 14.

We now leverage the width-proportional edges of the Sankey diagram to indicate the number of constraints being allocated from parent to child (again left to right). Besides the edge width, this visualization has all of the locating, filtering, and layout options available in the last visualization (so they will not be introduced again).

Ideally, the flow of constraints should obey the channels asserted by the concept hierarchy. However, our concept and requirements development process allows SEs to levy constraints from concept to concept as needed in order to con-

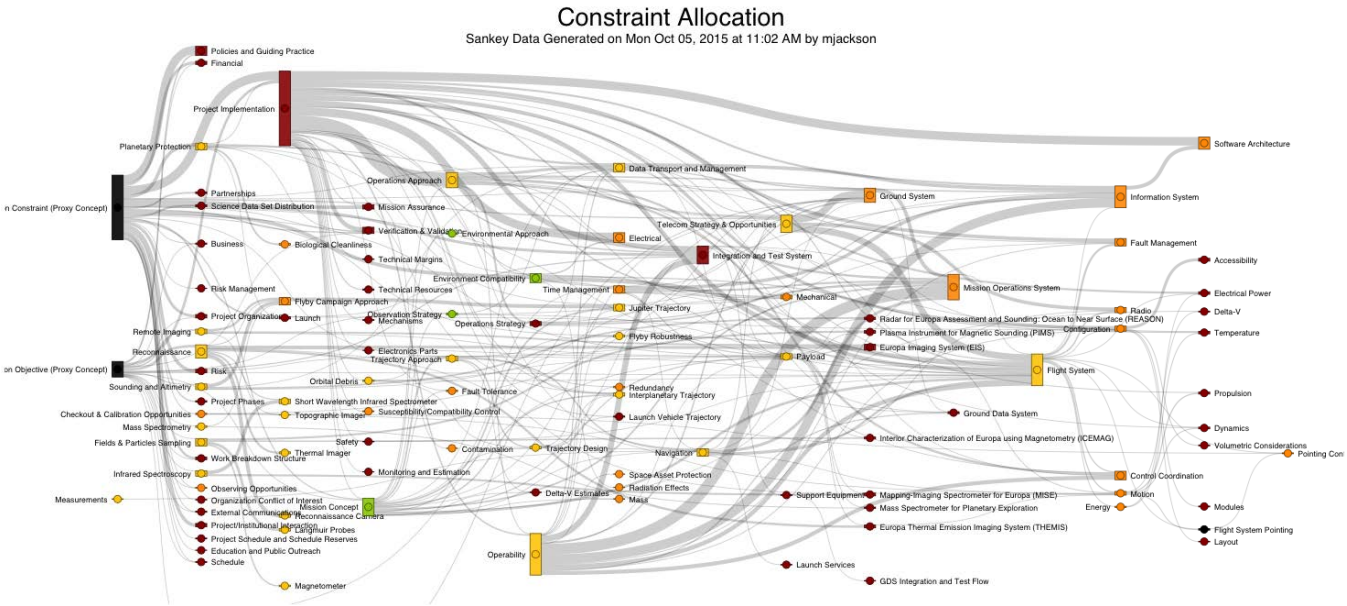

Figure 14. **Case Study 2: Constraint allocation (left to right) between concepts. The number of constraints allocated from parent to child is indicated by the width of the line between the concepts.**

tinue efficient forward progress. A process of reconciliation and resolution occurs later (and is supported by the third of our visualizations). When one queries the actual constraint allocations from concept to concept out of the model and visualize them in our second tool, we see the allocation network shown in Figure 14.

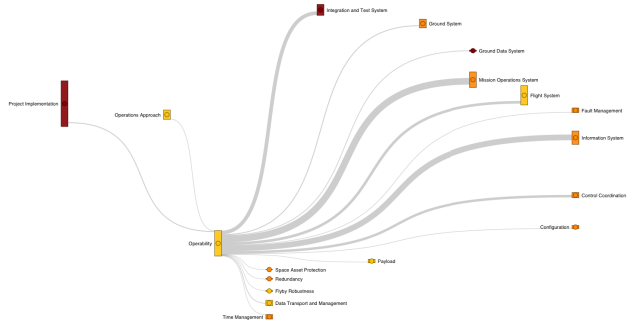

Figure 15. **Filtered graph showing allocations to and from the Operability concept.**

Figure 15 shows a closer look at a one concept (Operability) and in this visualization we can see the width-proportional lines. This diagram, at the time of writing, is not at the desired end state, but it is an expected step in conceptual and requirements development. Over time, and as the maturity of each concept, constraint, and allocation improves, we expect the structure of the allocation visualization to evolve towards that of the concept hierarchy.

Analysis

Using this visualization, we can explore and comprehend what is really happening in our engineering development. Concept leads can use this tool to examine the set of constraints that they will have to address in their concept and who levied them (in the previous visualization, they could only see who they should *expect* to receive constraints from). They can also see the maturity (development state) of each constraint, in addition to the state of the allocation (accepted, rejected, awaiting review, etc.). This is presented through the table, which now displays constraint-specific information (rather than concept information):

Constraint Info: Operability

Direction	AFID	Constraint ID	Constraint Name	Constraint Text	Allocation Status	Constraint Status
Outbound (Integration and Test System)	CT103.812	OPR.347	State checkpoint comparison report	The testbed shall provide the capability to compare two state checkpoints and produce a difference report.	Allocation Awaiting Review (by Parent AND Subview)	Identified
Outbound (Integration and Test System)	CT103.817	OPR.344	Testbed fault injection	The testbed shall be capable of tripping all fault protection monitors.	Allocation Awaiting Review (by Parent AND Subview)	Identified
Outbound (Integration and Test System)	CT103.803	OPR.337	Mid-test testbed state checkpoints	The testbed shall have the ability to record state checkpoints periodically during a test to enable the testbed to be re-initialized by the operator to a state captured mid-test without needing to perform a full testbed initialization.	Allocation Awaiting Review (by Parent AND Subview)	Identified
Outbound (Integration and Test System)	CT103.825	OPR.345	Testbed uplink product build	The system testbed shall provide the capability to quickly build uplink products for test use only (single commands, strings of commands, and relative timed sequences), outside of the regular flight build process.	Allocation Awaiting Review (by Parent AND Subview)	Identified
Outbound (Integration and Test System)	CT103.819	OPR.320	Testbed initialization time	The testbed (e.g., system testbed, WSTS) shall be able to be initialized to a user-defined state in under 1 (TBD) hour.	Allocation Awaiting Review (by Parent AND Subview)	Identified

Figure 16. **Allocation information associated with the Operability concept. This helps users understand the particular allocations beyond the network and magnitude presented in the interactive diagram.**

With this view, Systems Engineers can find the concepts that they oversee or those that affect their domain and understand the allocation of engineering needs to be addressed. They can understand quickly and intuitively the flow of issues to be addressed; what areas need work; what areas are getting traction. This gets at the cross-cutting and exploratory nature of the SE job. For SEs and managers who want details of maturity, these are available, and the architect who wants to know how well the actual constraint flow conforms to the asserted hierarchy can begin to assess that.

By looking the number of constraints a concept must address, constraints it derives, and the proportion of derived constraints allocated to each of its child concepts, SEs can understand at a glance the magnitude of work required in each concept and assess the sensitivity of the entire system to that concept. This allows concept leads to quickly understand how their own state of work fits into the Project's current state of development.

These two examples begin to demonstrate the power of using model-driven visualizations to assist our SE process. We can begin to realistically envision tools that answer even more questions. We will, for example, want to understand the full traceability of a single requirement. We can imagine clearly

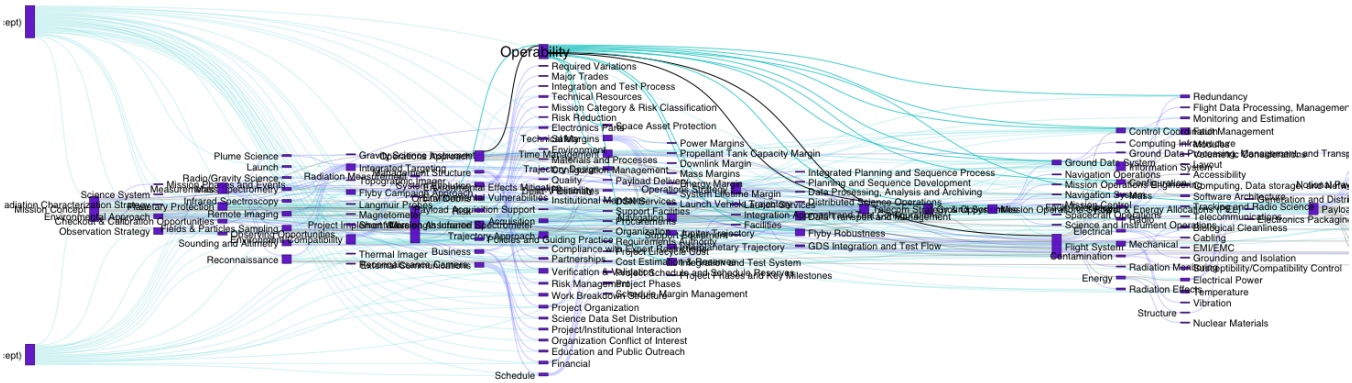

Figure 17. **The hierarchy comparison visualization.**

understanding more impacts of changes to constraints and requirements from the perspective of SEs: in the simplest case, who else is affected? Has anything I care about changed? Am I fully addressing my incoming requirements? How have these constraints and requirements changed over time? How sensitive are we to our driving requirements? The next two case studies will demonstrate our ability to answer these kinds of questions.

6. Case Study 3: Hierarchy Comparison

We noted in the previous example that the actual flow of constraints between concepts does not particularly match the asserted conceptual hierarchy we had expected it to obey. But how do we understand these differences? Our answer is a visualization that overlays the expected and actual hierarchies. The full comparison visualization is shown in Figure 17.

To visualize the comparison, we query from our system model a graph structure that contains both the expected and actual relationships. We start with the expected ("asserted") hierarchy, and add to that graph edges representing actual constraint allocations. A concept that allocates many constraints to a child concept is assigned a single edge of width 1 (because we are interested in the general flow, not specific constraints). Where we find actual ("inferred") constraint allocation for which there is an edge in the asserted hierarchy, we mark that edge as "asserted and inferred." If we find no actual allocations for an edge in the concept hierarchy, it is "asserted only." If we find allocations but no matching edge in the concept hierarchy, it is "inferred only." These distinctions are represented as different colored links between concepts, as described in Figure 18.

Concept Hierarchy: Asserted and Inferred
Sankey Data Generated on Mon Oct 05, 2015 at 11:01 AM by mjackson
Concept Maturities:**Identified** **Draft** **Prelim** **Baseline** **Final**
Line Color Key:**Asserted Only** **Inferred Only** **Asserted and Inferred**

Figure 18. **Comparison visualizations use color to differentiate between the kinds of parent-child relationships in the merged hierarchy.**

Inferred only (light teal) means that we have flowed constraints outside the rules. Asserted only (royal blue) means we have an asserted relationship but are not flowing any constraints. Ideally, we would only see asserted and inferred allocations (black lines).

This diagram gives us a way to directly assess correctness and completeness: we presume that our concept hierarchy is correct, so every case where we find a violation or no uses of edges that we expected to need is a case for further analysis.

To use this diagram more practically, we can look either at the entire dataset (unwieldy) or filter it for specific areas:

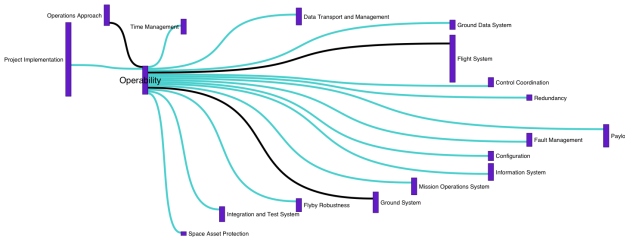

Figure 19. **From this filtered view we can assess how well the constraint allocations to Operability (Figure 15) match up to the concept hierarchy of CS1.**

As seen above, the Operability concept was only supposed to receive constraints from and allocate constraints to three other concepts. From a quick inspection of this diagram, we can see that Operability believes that it requires the ability to levy constraints upon a large number of child concepts in order to fully address the issues it was given, and these child concepts were not anticipated in the construction of the concept hierarchy.

This discrepancy is actually extremely useful. It means that we have work to do in either making a case to add the children (thus legitimizing the flows we needed in our actual development process), or we must scrutinize the allocations we have made from Operability to these other concepts to see if they are in fact incorrect.

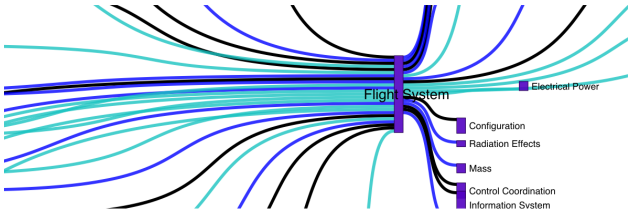

Figure 20. **The Flight System concept shows every permutation of the comparison.**

The Flight System concept's links to its parents and children

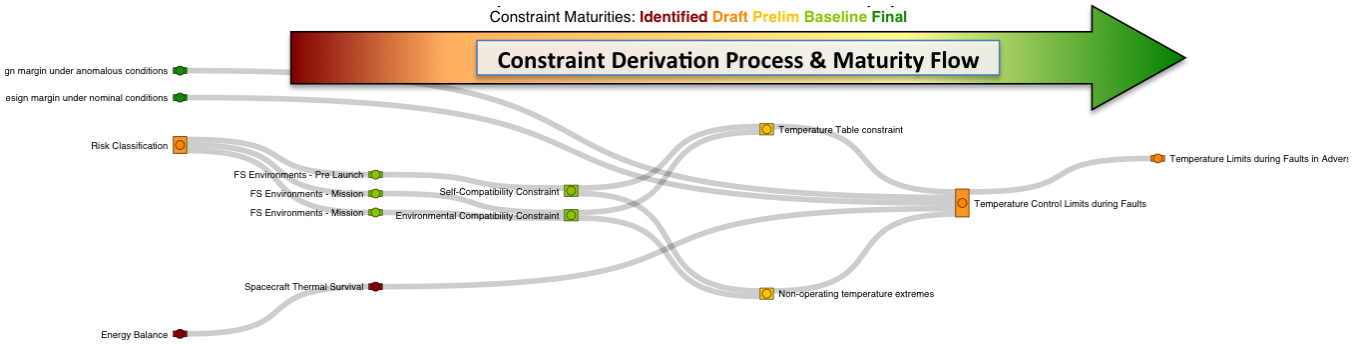

Figure 21. **We construct a traceability graph showing the derivation story for a constraint.**

(seen in Figure 20) show all three cases: there are links where we expected to get constraints but have not; cases where we get constraints from sources we did not expect; and cases where we both expect and have made allocations.

The analysis of completeness and correctness that this view supports is an analysis we have not been able to do easily in the past. Now we can do them simply by spending a few seconds looking at a diagram. Further, given the filtering and scoping capabilities of the visualization, we can perform this analysis using the entire concept hierarchy and all allocations, or we can dynamically render scoped views of concepts, parents, and children in the work area of each concept.

7. Case Study 4: Requirements Traceability

The last case study presented in this paper is visualizing the traceability of individual constraints through the concept hierarchy. The piece that none of our visualizations have previously showed is the mapping of a concept's derived constraints to its inbound constraints. This mapping is required in order to follow an individual constraint to its parent constraints, and their parents, or to children and their children going the opposite direction.

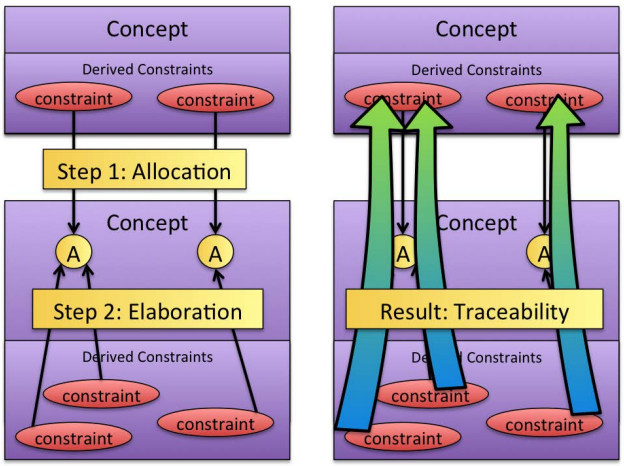

Figure 22. **We use the model to construct the traceability story. The constraint is allocated to the concept in step 1, elaboration and derivation occurs in step 2, and from there we can infer traceability links between constraints.**

Our concept hierarchy specifies expected connectivity, our allocation flow shows what concepts are responsible for addressing constraints, and our comparison visualization shows us how well our allocations match up to what we expect. None of these, however, show us the state of our derivation process at an individual constraint level and allow us to assess whether the constraint derivation process, and thus the *requirement* derivation process, have worked.

Recall that a major result of the concept and requirement development process is inbound constraints richly elaborated into a set of derived constraints that address them. This elaboration, shown in Figure 22, is done both in a rich narrative and in an underlying set of model relationships that form a graph between the incoming and outgoing constraints for each concept.

Traceability from child constraint to parent constraint is inferred by looking at the model links between allocation (which is to an intermediate object in the concept) and elaboration (which is from a constraint to that intermediate object). This results in complex network of traceability that is extremely difficult to understand manually. Our visualization translates the modeled traceability network into a graph structure and renders it for inspection by the user.

The constraint traceability network is quite large, so the reader will only get a sense of the scale of the graph from seeing the full network shown in Figure 23. While this complete view is much more comprehensible on a large monitor, visualizing the entire data set is really only useful to get a sense of the work completed and work remaining to be done. As this diagram indicates, there is significant work to go in mapping derived constraints to alocated constraints: we can see that there are many fragments of traceability. A traceability fragment is a case where one concept has completed the mapping of its derived constraints to its inbound constraints, but the parent and child concepts have not completed their mapping work. From this large diagram we can also get a sense of whether parent constraints are of higher maturity than child, and if not, identify individuals or groups to work the problem. This is extremely useful for the requirement SEs.

The other intended use of the visualization is to examine the traceability graph of one or a small number of constraints at a time. We have selected one constraint and show its traceability in Figure 21.

The individual constraint is furthest right with the parent constraints it responds to (and then their parents, etc.) propagating leftwards. The traceability view presented here masks the

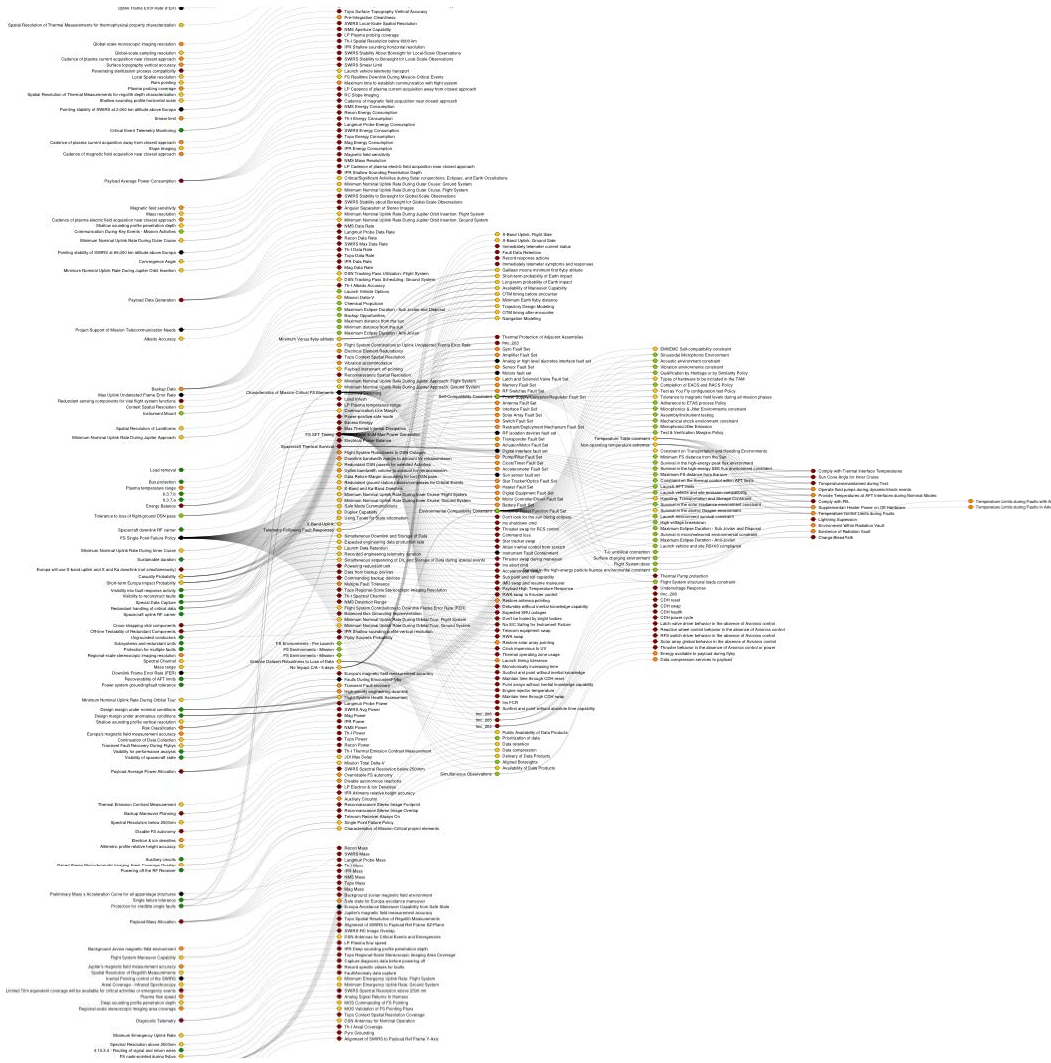

Figure 23. **Showing the traceability of all constraints at once gives a high level impression of the traceability flow, maturity, and state of work.**

separate allocation and elaboration steps into an abstracted, point-to-point edge between constraints. The "why" column of the table shows textual explanation and justification for the traceability obtained from the model. Figure 24 shows the table that explains the constraint attributes and relations to its neighbors.

Requirement Info: Temperature Control Limits during Faults

Role	Constraint ID	AFID	Constraint Name	Constraint Text	Why?	Concept Link	Constraint Maturity
Self	820	CT100.893	Temperature Control Limits during Faults	The thermal control design shall maintain allowable flight temperature limits, operating or non-operating whichever is appropriate, during system-level fault protection responses except for the following unlikely failure modes: a) S/C faults creating abnormal power dissipations and b) S/C faults at environmentally adverse altitude and/or solar distance having very small windows of vulnerability	---	Temperature	Draft
Child	821	CT100.894	Temperature Limits during Faults in Adverse Environments	The thermal control design shall maintain Prototyfight/Qualification temperature limits during S/C faults at environmentally adverse altitudes and/or solar distances having very small windows of vulnerability.	**Explanation:** Separating out faults in adverse environments **Explanation:** *no explanation text* **Rationalize:** *no rationalize text*	Temperature	Draft

Figure 24. **Table data provided to the user when an individual requirement is selected from the traceability visualization.**

Because our requirements are derived and projected from our constraints and concepts, the traceability network shown in these visualizations constitutes relationships between our requirements, allowing us to easily construct the traceability graph of our formal requirements based on the result of our derivation process.

Traditionally, traceability links are extremely superficial and lack semantic relevance. We do not require explicit rationale for why or how a requirement derives from another (a requirement usually has a rationale, but it is not a rationale for the traceability). We also do not traditionally capture the particulars of how a requirement actually addresses (in whole or part) the parent. Our modeling approach for the Europa project provides that richness in the pattern we use for linking our inbound and outbound constraints.

8. Technologies

The visualizations we have presented in this paper rely on a combination of commercial, homegrown, and open-source technologies. We use a commercial MBSE tool and our JPL-developed web front-end, View Editor. The visualizations are currently designed as a standalone web application that ingests datasets generated from the model and displays the visualizations discussed previously. An abstracted representation of our software stack is shown in Figure 25.

The visualization app relies on the open-source JavaScript library d3.js, and all visualizations use a significantly mod-

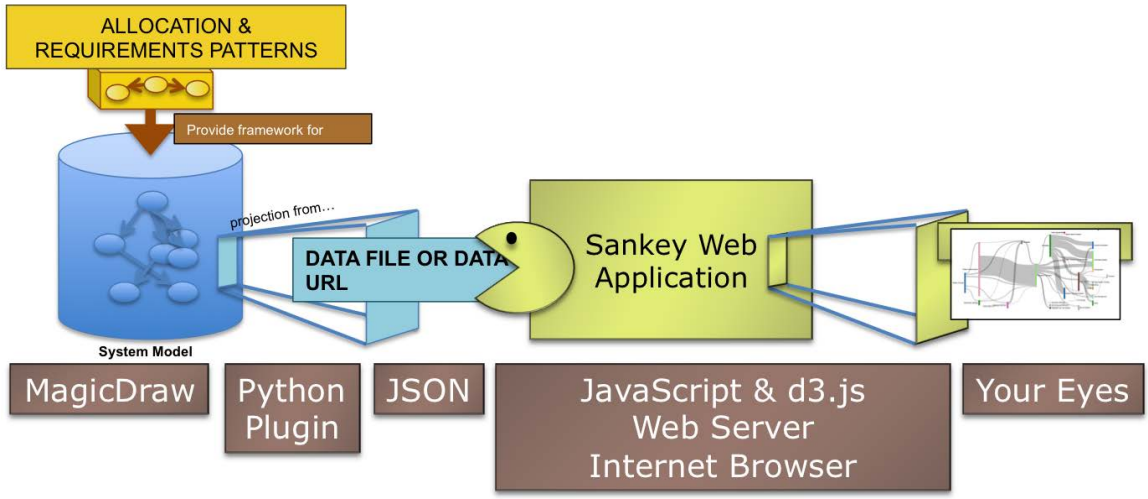

Figure 25. The end-to-end model to viewer toolchain.

ified version of the Sankey plugin for d3 (sankey.js) [8][9]. The customizations modify the sankey.js layout algorithm and data structure to add support for the kind of hierarchical layouts we want to show, add additional layout input parameters, and modify the data structures.

We currently interchange data between the system model and the visualization app via data generators in our MBSE tool that generate JSON data files containing all or a subset of the relevant data, somewhat preprocessed into a graph structure. The data is provided either as a link to a served file, a manual local upload, or a JSON string on the URL.

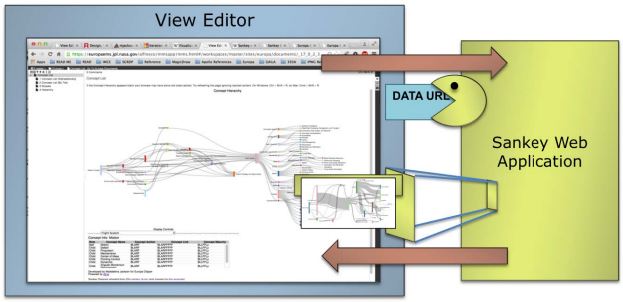

Figure 26. The visualization dashboard is provided by the standalone visualization application and rendered inside our web front end for easy access by viewers.

Other URL parameters are defined to provide initial filtering and sizing of the visualization for convenient inclusion into other technologies, such as View Editor or other web pages. This simple inclusion into other web applications is how we primarily use this capability on the Europa project.

9. Conclusions

The visualizations presented here, through their application to the MBSE effort of the proposed Europa Mission, eased some of the frustration and confusion that is an expected side effect of adapting to a new paradigm. These visualizations were found to make the SE-relevant content of the system model much more accessible to SEs, managers, and other stakeholders. The application of visualization was a demonstration of a new capability that is enabled by MBSE and is difficult to do traditionally because our traditional paradigm lacks the single source of truth of MBSE and the formal data structures contained therein against which we can write.

The case studies discussed here demonstrate the efficacy of visualizations in facilitating access to and comprehension, analysis, and exploration of large data sets by Systems Engineers on the Europa project. While these visualizations always responded to driving use cases from SEs, we found that they always addressed many more communication and system visibility issues than we initially expected.

Skills sets and tensions in SE and MBSE

Our ultimate goal in doing MBSE-driven visualization is to facilitate interaction with model data for SEs in an organized effective needs-based way so that they can make better-informed decisions more quickly. While visualizations are certainly interesting, this work, in the context of large-scale MBSE efforts and increasing MBSE adoption, suggests some areas we should revisit and focus on in future MBSE efforts. Specifically, the skill set of SEs on MBSE projects and the focus we devote to planning our MBSE efforts, and the role of different technologies.

This work was done at a very low level of effort by one developer over nearly a year, resulting in an estimate of less than three weeks in total. By reusing code, standing up new visualizations of hierarchical acyclic directed graphs now requires only a few hours of developer time, and new features are captured and made available in libraries or patterns. This means that the ratio of project value to time invested is extremely high.

A set of favorable circumstances created an environment where these visualizations could easily happen, and a brief discussion of the skills and circumstances is useful in making recommendations for facilitating similar work.

These visualizations were created when a Systems Engineer on the Flight System Requirements Team happened to have both the need for such visualizations firsthand and the skill set required to prototype the visualizations to satisfy that need. The was technically possible due to the colocation in one person of MBSE tool knowledge required write data generators from the MBSE tool, background in writing d3 web

applications, and also being customer for the tool. The work was *practically* possible due to a project structure and culture that allows and encouraged development of potentially useful applications.

This lead to a rapid and centralized development process that resulted in prototype visualizations, followed by demonstrations to other SEs, and ultimately in visualizations adopted by the mission that now are heavily used and respond to feature requests from the rest of the team. The volume of feature requests coming in indicates desire for more work of this kind.

This, however, is not a sustainable way to apply visualization to MBSE, because not every SE, developer, or modeler should be expected to have that particular skill set, nor do we explicitly staff our SE teams with those skills. The point we wish to emphasize here is that it was the intersection of software development experience, MBSE experience, *and* SE on the ground knowledge that was needed for this work to happen, which suggests it is a worthwhile skill set to consider in building future MBSE teams.

This consideration of skill sets brings us to an area of tension that should be addressed directly: the skill sets that we find useful in MBSE projects:

- Should SEs be modelers?
- Should SEs be software developers?
- If not, how can we do compose teams that do have those skill sets? Can we make use of cross-training? Will that lower costs?

We do not propose to answer these questions here, or imply that we have answers; instead, we suggest that projects and organizations wishing to employ MBSE confront this openly and discuss it directly. Looking at the "ingredients" that were present for this work, we synthesize three roles that were at play:

- Systems Engineer (with SE domain knowledge and need)
- Product designer/developer (with design and software experience)
- Facilitator/requirements engineer (bridge between SE and developer, analyzes needs vs capabilities, etc.)

We have many people in at least one of these roles on our projects; we do not need them to manifest in one person. When the SE does not have the time, interest, or specific skill set to make visualizations, thinking about roles required to do this development effectively helps us select the people we should put together to make effective visualization happen.

A second area into which this work provides insight is that of planning for MBSE efforts. As previously mentioned, this work initially came about as a result of an SE with a certain skill set trying to solve their own problems in this new paradigm. It was not a planned effort as part of an MBSE infusion strategy or vision.

It is the impression of the authors that much of the planning effort goes to knowledge representation and capture when attempting to apply MBSE. While unarguably valuable, we assert that more effort must be devoted to studying the SE process employed by practicing SEs, developing use cases for how SEs wish to interact with the model data, collecting SE questions and desired capabilities and mapping them to views and visualizations that can be constructed to help SEs work in an MBSE project - all before the project even starts. The positive responses of SEs and MBSE practitioners to the work presented here and the degree to which these simple tools have been adopted and infused into the rest of the mission and continual requests for new features and additional visualizations supports our assertion.

If we want to approach MBSE-driven visualizations in an institutional top-down way, we must study our current SE practices to get use cases and requirements. At the same time, MBSE-driven visualizations are not mature and the area has not been explored sufficiently to begin any kind of top-down standardization or technology selection that would limit research and development in this area.

10. Final Recommendations

The core challenge facing MBSE infusion efforts is this: if we as Systems Engineers cannot explore, interact with, categorize, filter, annotate, and generally be creative with the data in the model, we cannot do our jobs. Our vision for pushing the boundaries of the SE practice and advancing the state of the art with MBSE is irrelevant if engineers cannot do the SE work we are tasked with, and in that situation the future may not include MBSE. Inability to communicate model data intuitively, effectively, creatively to stakeholders in a way that gives them confidence in and ownership of the system can make MBSE infusion difficult or even unsuccessful.

From this exercise in developing MBSE-driven visualizations for requirements development for the Europa project, we conclude that MBSE-driven interactive visualizations are one way to facilitate interaction with relevant model data by SEs, and should continue to be formally explored as part of MBSE infusion efforts. We have formed some recommendations for parties wishing to pursue similar efforts.

Catalog SE questions

Begin soliciting and cataloging the nuanced questions and concerns that Systems Engineers ask now, but do not always verbalize.

Study visualization strategically

Invest in a strategic study of MBSE-driven visualization. Identify explicit goals for the study of SE needs, processes, and views, and identify explicit goals and requirements for visualization infrastructure development, including best practices and standards for software development.

Find the right skill sets

Consider the three roles discussed earlier and make sure that they are represented in teams assigned to creating visualizations.

Collaborate with software and visualization domains

Foster collaboration between line organizations that provide skilled people in areas of visualization, SE, software development, and architecture.

Encourage communication of results

Support regular visualization showcases and discussion groups to motivate practitioners to generate new ideas, share tools and skills, and start creating a community of practice.

APPENDIX

This appendix contains full-width renderings of the smaller figures presented earlier in this paper.

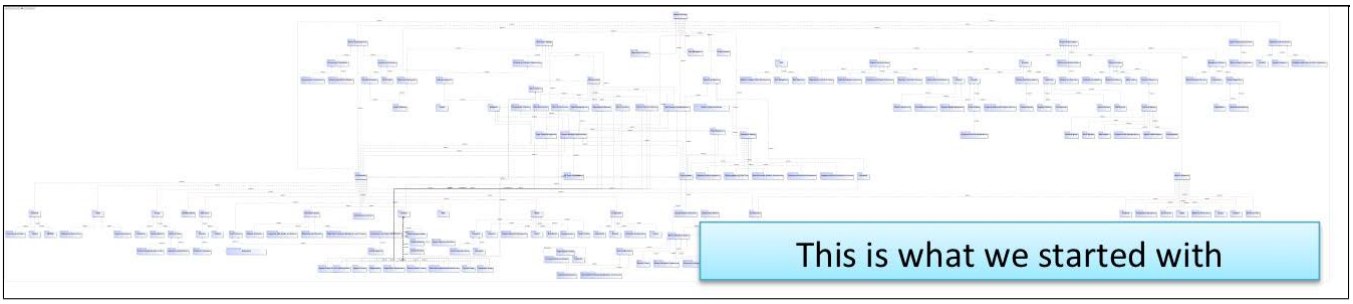

Figure 27. (Full-sized Figure 5) An original concept hierarchy visualization: "The Scroll."

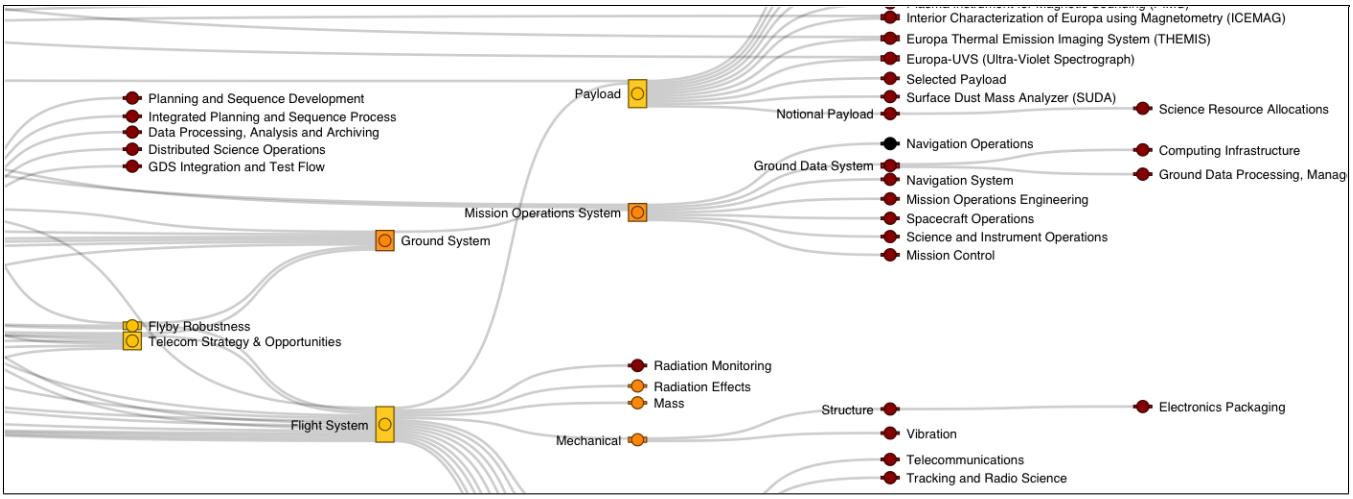

Figure 28. (Full-sized Figure 8) Medium-zoom view of the concept hierarchy.

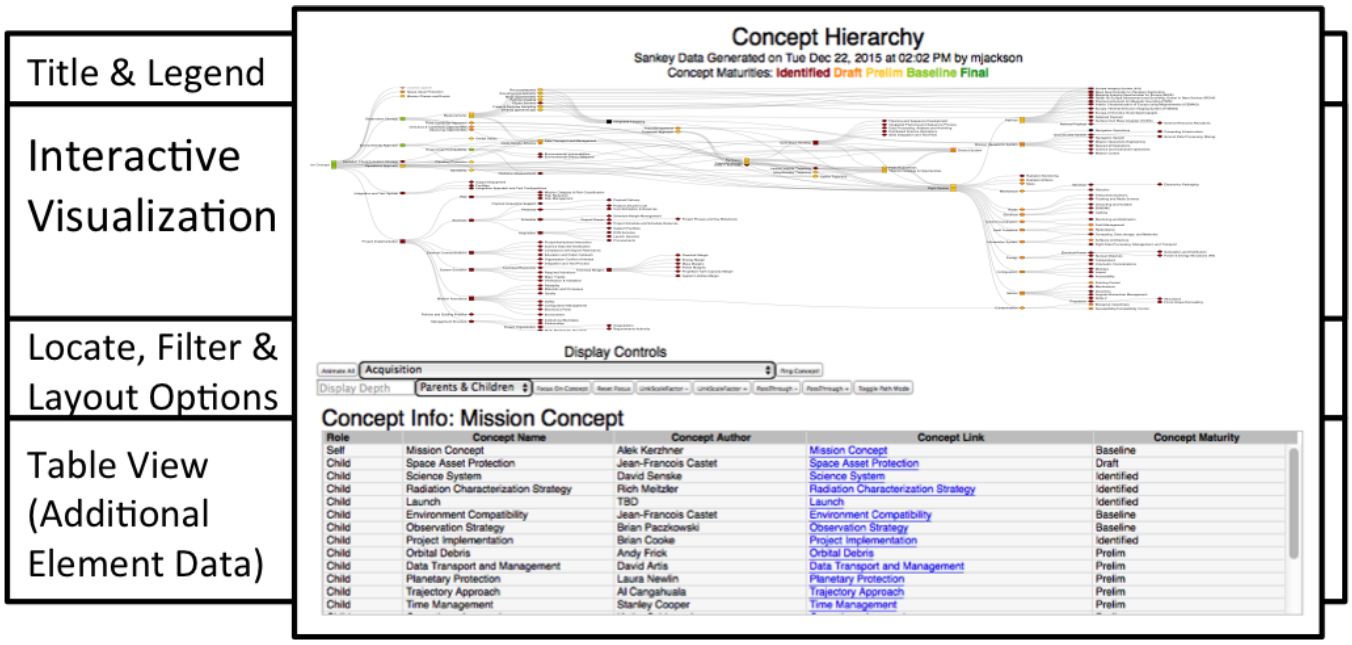

Figure 29. (Full-sized Figure 10) The visualization and interaction interface as presented to the user.

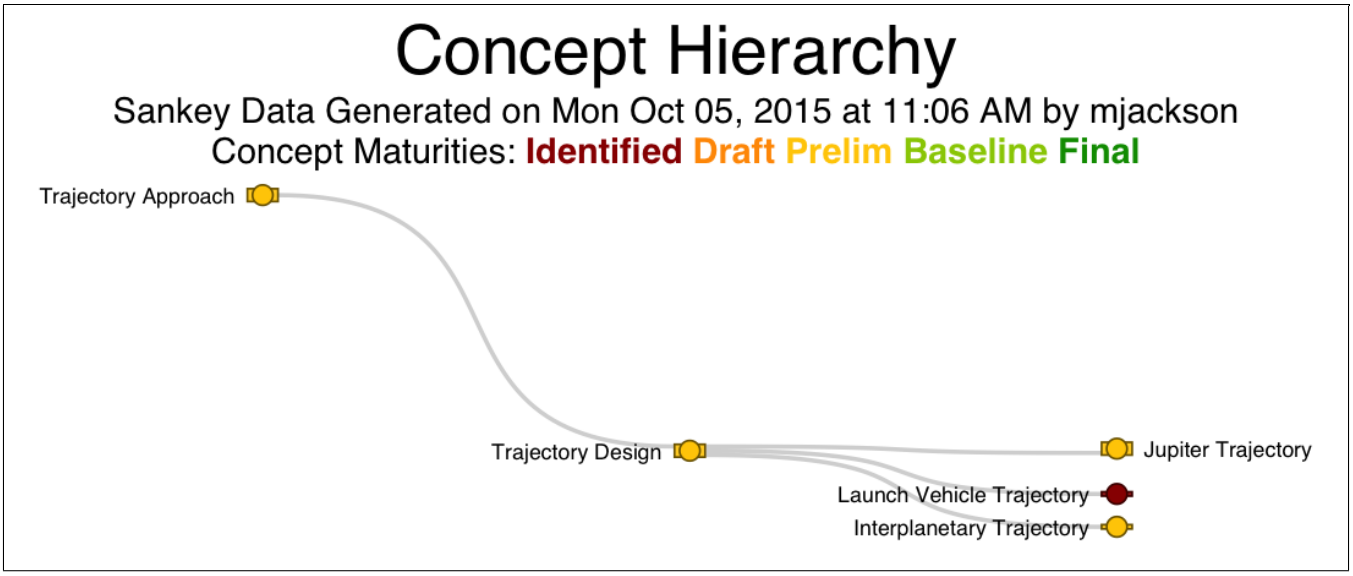

Figure 30. (Full-sized Figure 12) A filtered selection of hierarchy data showing only direct parents and children of the Trajectory Approach concept.

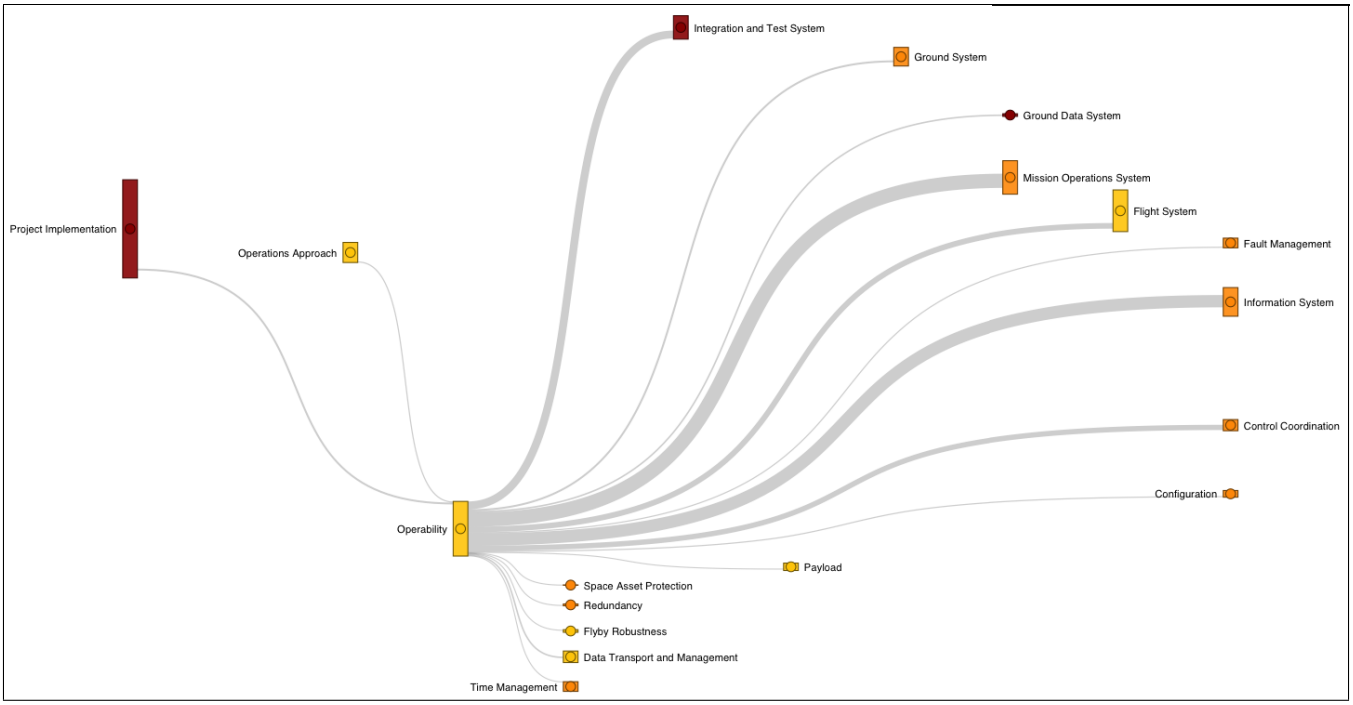

Figure 31. (Full-sized Figure 15) Filtered graph showing allocations to and from the Operability concept.

Constraint Info: Operability

Direction	AFID	Constraint ID	Constraint Name	Constraint Text	Allocation Status	Constraint Status
Outbound (Integration and Test System)	CT103.812	OPR.347	State checkpoint comparison report	The testbed shall provide the capability to compare two state checkpoints and produce a difference report.	Allocation Awaiting Review (by Parent AND Subview)	Identified
Outbound (Integration and Test System)	CT103.817	OPR.344	Testbed fault injection	The testbed shall be capable of tripping all fault protection monitors.	Allocation Awaiting Review (by Parent AND Subview)	Identified
Outbound (Integration and Test System)	CT103.803	OPR.337	Mid-test testbed state checkpoints	The testbed shall have the ability to record state checkpoints periodically during a test to enable the testbed to be re-initialized by the operator to a state captured mid-test without needing to perform a full testbed initialization.	Allocation Awaiting Review (by Parent AND Subview)	Identified
Outbound (Integration and Test System)	CT103.825	OPR.345	Testbed uplink product build	The system testbed shall provide the capability to quickly build uplink products for test use only (single commands, strings of commands, and relative timed sequences), outside of the regular flight build process.	Allocation Awaiting Review (by Parent AND Subview)	Identified
Outbound (Integration and Test System)	CT103.819	OPR.320	Testbed initialization time	The testbed (e.g., system testbed, WSTS) shall be able to be initialized to a user-defined state in under 1 (TBD) hour.	Allocation Awaiting Review (by Parent AND Subview)	Identified
					Allocation Awaiting	

Figure 32. (Full-sized Figure 16) Allocation information associated with the Operability concept. This helps users understand the particular allocations beyond the network and magnitude presented in the interactive diagram.

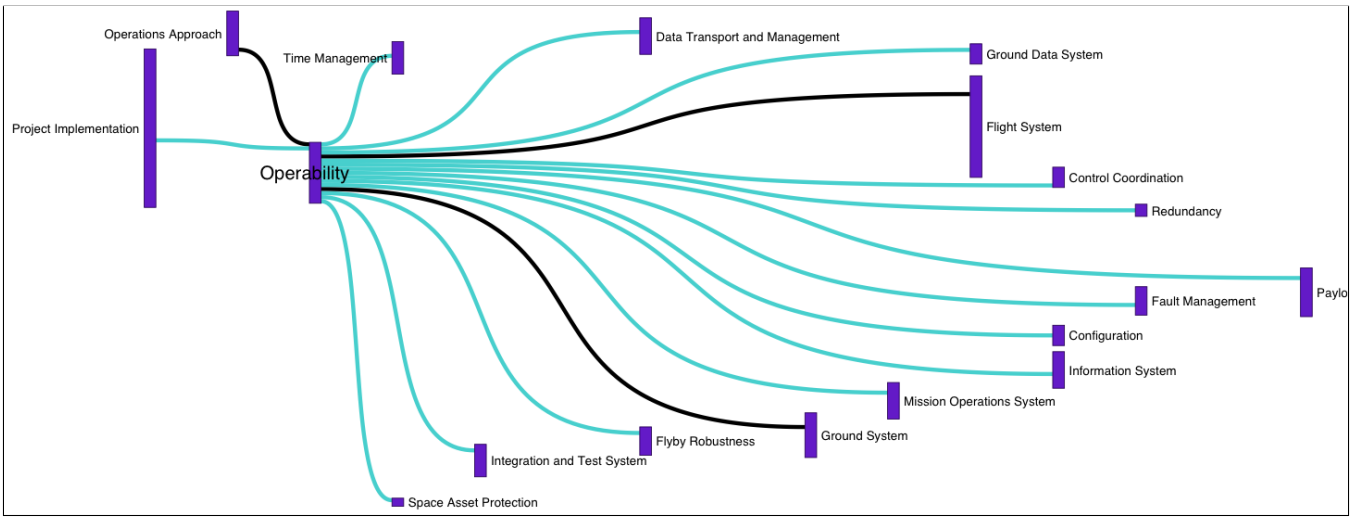

Figure 33. **(Full-sized Figure 19) From this filtered view we can assess how well the constraint allocations to Operability (Figure 15) match up to the concept hierarchy of CS1.**

Requirement Info: Temperature Control Limits during Faults

Role	Constraint ID	AFID	Constraint Name	Constraint Text	Why?	Concept Link	Constraint Maturity
Self	820	CT100.893	Temperature Control Limits during Faults	The thermal control design shall maintain allowable flight temperature limits, operating or non-operating whichever is appropriate, during system-level fault protection responses except for the following unlikely failure modes: a) S/C faults creating abnormal power dissipations and b) S/C faults at environmentally adverse attitude and/or solar distance having very small windows of vulnerability	---	Temperature	Draft
Child	821	CT100.894	Temperature Limits during Faults in Adverse Environments	The thermal control design shall maintain Protoflight/Qualification temperature limits during S/C faults at environmentally adverse attitudes and/or solar distances having very small windows of vulnerability.	**Explanation:** Separating out faults in adverse environments	Temperature	Draft
					Explanation: *no explanation text*		

Figure 34. **(Full-sized Figure 24) Table data provided to the user when an individual requirement is selected from the traceability visualization.**

ACKNOWLEDGMENTS

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

The authors wish to thank Brian Cooke and Todd Bayer for the support this work received from the Project and for their feedback, suggestions, and feature requests. Thanks also to Alek Kerzhner, David Coren and the whole MSET Team for hosting the visualizations, for developing the View Editor integration, for suggestions and for technical input. Many thanks to Navid Dehghani, Dan Dvorak, Chi Lin and Dave Nichols for their interest in this work and for providing opportunities to demonstrate and showcase it.

REFERENCES

[1] T. Bayer, S. Chung, B. Cole, B. Cooke, F. Dekens, C. Delp, I. Gontijo, and D. Wagner, "Update on the Model Based Systems Engineering on the Europa Mission Concept Study," in Proceedings of Aerospace Conference. Big Sky, Montana: IEEE, 2013

[2] G. Dubos, D. Coren, A. Kerzhner, S. Chung, J. Castet, "Modeling of the Flight System Design in the Early Formulation of the Europa Project," pending publication in Proceedings of Aerospace Conference. Big Sky, Montana: IEEE, 2016

[3] M. Jackson, C. L. Delp, D. Bindschadler, M. Sarrel, R. Wollaeger, and D. Lam, "Dynamic Gate Product and Artifact Generation from System Models," in Proceedings of Aerospace Conference. Big Sky, Montana: IEEE, 2011.

[4] Delp, C., Lam, D., Fosse, E., Lee, C., "Model Based Document and Report Generation for Systems Engineering," IEEE Aerospace Conference, IEEE, New Jersey, 2013, pp. 1-11.

[5] Open MBEE github repository: https://github.com/Open-MBEE

[6] B. Rasmussen and B. Muirhead, "A Case for Model-Based Architecting in NASA," 2012, unpublished.

[7] M. Jackson, M. Wilkerson, and J. Castet, "Exposing Hidden Parts of the SE Process: MBSE Patterns and Tools for Tracking and Traceability," pending publication in Proceedings of Aerospace Conference. Big Sky, Montana: IEEE, 2016

[8] Data driven Documents: http://d3js.org/

[9] D3 Sankey Diagrams: http://bost.ocks.org/mike/sankey/

[10] INCOSE Insight, August 2015, Vol. 18, Issue 2.

BIOGRAPHY

Maddalena Jackson *received a B.S. in Engineering from Harvey Mudd College in 2008 and has been at JPL working with MBSE since. She has served on projects both as an SE (using MBSE) and as a software developer in many domains, from human exploration to ground systems to cyber defense and now NASA's Europa Project. She is primarily a member of the Europa Projects's Flight System Requirements Team, the role for which she developed the visualizations presented in this paper. She is also Europa's Model Systems Engineering Team Software Management Lead, which oversees software developed in support of MBSE activities on the project.*

Marcus Wilkerson *received his B.S. degree in Aerospace Engineering from the University of Colorado at Boulder in 2008. He is currently the Flight System Requirements Team lead for the Europa Project. He previously served as Integration and Test lead for OPALS, an optical communications technology demonstration on the International Space Station.*