Defining and Documenting Tree Data Structures using Model-Based Systems Engineering Tools

M. Miller[1]
Northrop Grumman Corporation, Space Systems, Roy, Utah, 84067

Many systems engineering concepts can be represented as a hierarchy of nodes. For example, a system-level requirement is commonly decomposed into several lower-level requirements, which can each be decomposed further. Fault Tree Analyses (FTA) are built around a structured set of failure events. Block Definition Diagrams can define a system as a hierarchy of sub-elements. These concepts share the same underlying data structure - the tree. In conventional systems engineering, documentation such as tables, numbered lists, and linked databases are used to capture tree structures. With the advent of Model-Based Systems Engineering (MBSE), a new suite of tools has become available to system architects and designers. MBSE offers improved flexibility which could be well-suited to creating informative, maintainable trees. This paper explores conventional tree documentation tools, and compares them to MBSE methods. In particular, it focuses on each tool's ability to enforce configuration management, display data for visualization, and be used effectively by a wide user base.

I. Introduction

A. Motivation

Communication is crucial to effective systems engineering. Systems engineers convey high-level concepts and technical details to customers, implementers, and stakeholders of all backgrounds. Documenting these topics is essential in order to ensure that the correct information is conveyed in the present, and that a source of truth is preserved throughout the lifecycle of the system of interest.

Many systems engineering concepts can be represented by tree data structures. By correctly documenting these topics, systems engineers can communicate more effectively in order to benefit the entire program. This paper investigates the software tools used to document trees. Tools will be evaluated in order to understand effective documentation and ultimately achieve better communication.

B. Scope

1. Evaluation Scope

This paper evaluates various tools in a qualitative sense. Tools are evaluated against the criteria defined in Section II. This paper is not a trade study. It does not attempt select the "best" tool. The optimal tool for a specific use case depends on a number of factors – the type of program, the resources available to it, its stakeholders, etc. The evaluation criteria outlined in this paper could be used as the basis of a trade study, but the weights assigned to each category would need to be determined on an individual basis.

In addition, Section II Subsection G lists a number of factors that could be important to a program, but are not evaluated in this paper. These factors should also be considered if the reader desires to select a documentation tool.

2. Tools Selected for Evaluation

Systems engineering is a broad discipline, and there are a significant number of tools available to systems engineers. The International Council on Systems Engineering (INCOSE) and Project Performance International (PPI) are in the progress of cataloging these tools [1], but the effort is incomplete as of the time of writing. Due to this breadth, this paper does not attempt to evaluate all tools. A small subset of tools will be evaluated, selected to be representative of a larger categories of tools.

[1] Systems Modeling and Simulation Engineer.

Many applications are targeted towards a specific use case. For example, Reliasoft XFMEA [2] is a highly specialized application for running reliability analyses. While it documents trees (such as Reliability Block Diagrams and fault trees), the application was not designed as a general-purpose tool. Targeted applications such as XFMEA are not included in this investigation. Only flexible, multi-purpose tools which can be applied to a large number of scenarios are evaluated.

3. Depth of Evaluation

A number of systems engineering tools provide rich functionality that extends beyond the baseline Graphical User Interface (GUI). These extensions include macros, script executors, and third-party plugins. Extensions are noted when they enable useful functionality, but in general they are not evaluated with the baseline tool. When multiple applications are used together, or when scripting is used to enhance the functionality of a tool, the combined solution in evaluated in Section IV, Subsection D - Custom Solutions.

C. Computer Science Terminology

Trees are a computer science concept, so in this paper some computer science terminology will be used to discuss them. Textbook definitions of these terms are provided below.

Tree, *node*, *root* - "… a tree is an undirected graph with a special *node* called the *root*, in which every *node* is connected to the root by exactly one path" [3].

Child, *parent* - "When a pair of nodes are neighbors in the graph, the node nearest the root is called the *parent* and the other node is its *child*" [3].

Leaf - "A *leaf* node is a node that has no children" [3].

Level, *height* – "The nodes of a tree can be organized into *levels*, based on how many edges away from the root they are… The *height* of a tree is the maximum level of any of its nodes…" [3].

II. Evaluation Criteria

D. Configuration Management (CM)

Configuration management is commonly considered an independent process within systems engineering [4][5]. However, an effective tool provides some level of configuration management on its own. Engineers must be prepared for multiple iterations during design, including before formal configuration management is implemented, and between formal design cycles. The following subsections will be examined for each tool with regards to configuration management.

1. Index

This document will use the term *index* to refer to a field that captures the position of a node within a tree. A common indexing scheme is a list of integers separated by periods (e.g. 1.1.2.3). Adding a sibling node increments the last integer, while adding a child appends an additional integer to the list. The index depends entirely on the position of the node within the tree. If a tree is restructured, a node that has been moved will receive a new index.

A useful index conveys information about a node's position in the tree without requiring the rest of the tree to be viewed.

Indexes can be added and maintained manually, but an effective tree documentation tool must automatically enforce indexes. This guarantees that indexes will be unique, and represent the correct location of the node within the tree.

2. Identifier (ID)

This document will refer to ID as a field that uniquely identifies a node. Unlike an index, if a tree is restructured, each node retains its original ID. An ID is commonly an integer, or a string followed by an integer (e.g. 57, SD-347).

IDs are critical for tracking changes in nodes throughout the program lifecycle. In the event that the structure of the tree changes, the history of individual nodes can still be maintained by comparing IDs. In addition, IDs are commonly used to match data with external sources.

IDs can be tracked manually, but it is beneficial for a tree documentation tool to automatically assign IDs. Additionally, a useful ID schema enforces unique IDs even for deleted nodes.

3. Version History

As the values and structures within the tree change, engineers need to review and track changes. A tool should be able to compare the current version of a tree it documents to past versions.

E. Visualization

A useful tool must not only store data, but provide it to stakeholders in a consumable manner.

1. Graphical Display

Graphical display of trees is useful for providing a quick reference of a larger dataset. An important function of systems engineering is presenting complicated information for consumption by stakeholders, regardless of their level of technicality. Graphical displays are useful in achieving this goal.

2. Data Display

In addition to a graphical display, tools must also present data so that it can be used in analysis. Nodes that have numerical parameters should be able to be compared easily, and across multiple levels of the tree when appropriate.

3. Import/Export

Systems Engineers rely on a wide range of tools, especially when more than one organization is involved. An effective documentation tool must be able to import and export data using common file formats.

F. Usability

1. Ease of Implementation

In order for the tree to be documented, it must be inputted into a tool. A tool with a complicated configuration process may require additional time, funding, or resources to use. Additionally, the act of building a tree within a tool must not be overly complicated or time-consuming.

2. Accessibility

A tool that captures key program information must be used by a large number of people of all disciplines. A tool that is easy to use gives a greater number of people access to its data.

G. Criteria not considered

Not all criteria involved in documenting trees can be examined by this paper. When choosing a tool, individual programs must determine which factors are most important. A list of unexplored topics is provided below.

1. Scalability – tools could be evaluated for their ability to handle large datasets (e.g. document thousands of nodes or trees with height 50)
2. Access Control – programs may be interested in restricting which users can access or modify data.
3. Information Security – many programs deal with sensitive information. Tools could be evaluated based on their ability to encrypt data, or move information between insecure and secure systems.
4. Cost – many small programs or programs with limited enterprise architecture are restricted by the cost of tools. Free or already-purchased tools may be preferable to purchasing expensive licenses or designing custom applications.
5. Parametric properties – it may useful to a program for the tree to automatically preform some kind of operation on its nodes. For example, if mass is a parameter, a tree could automatically set the mass of a node to the sum of the masses of its children.

III. Problem Description

To assist with consistent evaluations of the tools in Section IV, a mock dataset will be used. This dataset is highly simplified – a real-world tree would almost certainly have significantly more nodes and parameters.

The tree to be evaluated represents the physical breakdown of a simplified Cordless Drill. The components of the drill are presented in Fig. 1 as an ordered list.

Cordless Drill

1. Electrical subsystem
    - 1.1. Battery
    - 1.2. Switch
    - 1.3. Motor
    - 1.4. Wiring
2. Mechanical subsystem
    - 2.1. Frame
        - 2.1.1. Shell
        - 2.1.2. Battery housing
    - 2.2. Gearbox
        - 2.2.1. Permanent gear reduction
        - 2.2.2. Adjustable gear reduction
    - 2.3. Chuck

Fig. 1 Cordless Drill Node Structure

In addition to the tree structure outlined above, each node on the Cordless Drill tree will contain the parameters laid out in Table 1.

Table 1 Cordless Drill Parameters

Parameter	Description
ID	Guaranteed unique. This field will serve to identify each node.
Index	Guaranteed unique. This field serves to specify the nodes position in the tree.
Name	The human readable name of the node.
Description	A human readable description of the node.
Mass	The mass of the node in grams, rounded to the nearest whole number.

IV. Evaluation of Documentation Solutions

A. Word Processing Tool

Microsoft Word [6] was chosen as a representative application for word processing. Microsoft Word is ubiquitous across systems engineering programs. The application is highly mature and has numerous built-in features. Even when other tools are used as the source of truth of a dataset, Microsoft Word is often used to create final documents required by a program's customer. As a result, storing trees in Word can reduce the number of steps required to manage and deliver data.

Microsoft Word has the ability to document trees is several different ways, including tables, diagrams, and ordered lists. Because Word is being evaluated as a word processing tool, only ordered lists will be evaluated.

In Word, one method of capturing trees is using headings to denote nodes, and using paragraph text to capture node parameters. This method is especially effective when parameters are in the form of long blocks of text. A screenshot of one potential Word implementation of the Cordless Drill tree is presented below (Fig. 2).

Cordless Drill

1. CD-1 Electrical subsystem
All components related to powering the screwdriver. Mass: 900.

1.1. CD-2 Battery
12 volt Nickel-cadmium battery pack. Mass: 600.

1.2. CD-3 Switch
On-off switch. Mass: 50.

1.3. CD-4 Motor
Brushed 12 volt Direct Current motor. Mass: 200.

1.4. CD-5 Wiring
Electrical wires connecting components. Mass: 50.

2. CD-6 Mechanical subsystem
All components related to the physical structure of the screwdriver. Mass: 700.

2.1. CD-7 Frame
Primary structural elements of the screwdriver. Mass: 100.

2.1.1. CD-8 Shell
Plastic, houses all other components. Mass: 70.

2.1.2. CD-9 Battery housing
Contains battery. Mass: 30.

2.2. CD-10 Gearbox
Two-stage planetary gearbox. Mass: 400.

2.2.1. CD-11 Permanent gear reduction
Planetary reduction gearbox. Mass: 200.

2.2.2. CD-12 Adjustable gear reduction
Planetary reduction gearbox with adjustable slip. Mass: 200.

2.3. CD-13 Chuck
Single-sleeve keyless chuck. Mass: 200.

Fig. 2 Word Implementation of Cordless Drill Tree

1. Configuration Management

Word provides automatic, forced indexing through its Numbered List and Outline Level numbering schemas. This implementation guarantees that even when nodes are moved, they will maintain correct indexing, reducing the potential for human error. On the other hand, Word does not support unique identifiers. The IDs in Fig. 2 were manually defined. They are susceptible to user error.

Word contains a "Track Changes" and "Compare" feature that allows different versions to be compared. However, without unique identifiers, this feature cannot properly track changes to a node. For example, Word's Track Changes tool denotes a moved node as a deletion of one node plus the addition of another.

Word does not natively support any kind of history tracking other than discussed above. To get around this problem, Word is often paired with Product Lifecycle Management (PLM) tools such as TeamCenter [7]. While this combination can be used effectively, for the intents of this paper it is considered a Custom Solution discussed in Section D**.**

2. Visualization

Storing trees as a numbered list is reader and printer friendly. As the list of nodes grows, it simply requires more pages. However, it is difficult to easily see the overall structure of the tree, especially if nodes have many parameters and there is significant space between header lines on a page.

In addition, it is difficult to compare data parameters of nodes. In Fig. 2, it is challenging to compare the mass parameters of nodes with other nodes at the same level, since it is difficult to distinguish parents and children without examining the index of a node.

Finally, while Word supports export into a number of document formats, the parameters defined above cannot be easily exported into common formats such as text, CSV (Comma Separated Value), or JSON (JavaScript Object Notation) files.

3. Usability

Implementing a tree in Word is straightforward. Unless specific formatting is desired by the program, Word is capable of creating trees without any kind of configuration. In addition, the near universal adoption of Microsoft Word ensures that nearly all stakeholders will be able to use it.

B. Spreadsheet Tool

Microsoft Excel [8] was selected to represent spreadsheet tools. Like Word, Microsoft Excel is a powerful application with numerous built in features and wide adoption. Excel can capture trees as tables or diagrams. Since it is being evaluated as a spreadsheet tool, only tables will be considered.

In excel, each node can be modeled as a row on the spreadsheet. The row "Level" was added for convenience and represents the level of the node (Fig. 3).

Inde	ID	Name	Description	Mas	Leve
1	CD-1	Electrical subsystem	All components related to powering the screwdriver.	900	1
1.1	CD-2	Battery	12 volt Nickel-cadmium battery pack.	600	2
1.2	CD-3	Switch	On-off switch.	50	2
1.3	CD-4	Motor	Brushed 12 volt Direct Current motor.	200	2
1.4	CD-5	Wiring	Electrical wires connecting components.	50	2
2	CD-6	Mechanical subsystem	All components related to the physical structure of the screwdriver.	700	1
2.1	CD-7	Frame	Primary structural elements of the screwdriver.	100	2
2.1.1	CD-8	Shell	Plastic, houses all other components.	70	3
2.1.2	CD-9	Battery housing	Contains battery.	30	3
2.2	CD-10	Gearbox	Two-stage planetary gearbox.	400	2
2.2.1	CD-11	Permanent gear reduction	Planetary reduction gearbox.	200	3
2.2.2	CD-12	Adjustable gear reduction	Planetary reduction gearbox with adjustable slip.	200	3
2.3	CD-13	Chuck	Single-sleeve keyless chuck.	200	2

Fig. 3 Excel Implementation of Cordless Drill Tree

1. Configuration Management

Without significant effort required to create and implement a formula or macro, excel is not capable of automatically generating indexes for nodes. This means that rearranging nodes risks human error, and can be time consuming to maintain. It should be noted that Excel is capable of enforcing uniqueness on a column using the Data Validation tool. While this prevents some errors, it does not guarantee that indexes are correct.

Similarly, the Data Validation tool prevents some but not all errors when defining IDs. For example, the IDs for two terms could be swapped without violating uniqueness.

Excel does not natively allow versions to be compared. Comparisons can be completed using external tools such as Beyond Compare [9], or by exporting to a more common file format such as CSV and comparing with a number of tools. Like Word, Excel does not natively support any kind of version history.

2. Visualization

As the number of parameters increases, it can be difficult to view all columns on a single screen, or fit them on a single sheet of paper. This issue is resolved by hiding columns, or splitting parameters into multiple tables when exporting the data as a document.

One of Excel's strengths is obtaining meaningful comparison between node parameters. Excel tables can be filtered and sorted. Additional columns for more in-depth analysis can be easily added and removed. This allows the user to quickly evaluate data and present it in a meaningful way. For example, it is simple to determine which Level 2 Cordless Drill nodes have the most impact on the overall mass of the system (Fig. 4).

Inde	ID	Name	Description	Mas	Leve	Mass Percen
1.1	CD-2	Battery	12 volt Nickel-cadmium battery pack.	600	2	38%
2.2	CD-10	Gearbox	Two-stage planetary gearbox.	400	2	25%
1.3	CD-4	Motor	Brushed 12 volt Direct Current motor.	200	2	13%
2.3	CD-13	Chuck	Single-sleeve keyless chuck.	200	2	13%
2.1	CD-7	Frame	Primary structural elements of the screwdriver.	100	2	6%
1.2	CD-3	Switch	On-off switch.	50	2	3%
1.4	CD-5	Wiring	Electrical wires connecting components.	50	2	3%

Fig. 4 Level Two Nodes sorted by Mass Percent

Finally, Excel supports export to CSV, a data format widely used by a number of applications.

3. *Usability*

Like in Word, implementing a tree in Excel is straightforward. Configuration is only required to ensure column uniqueness. In addition, the near universal adoption of Microsoft Excel ensures that nearly all stakeholders will be able to use it.

C. Requirement Management System

IBM Rationale DOORS [10] will be evaluated as a Requirement Management System. DOORS has widespread adoption among system engineering programs. DOORS was developed for requirements management, but it is often used for documentation of other trees as well.

DOORS will be evaluated in its most common configuration. A typical DOORS database consists of multiple *modules*. A module is a table with rows called *objects.* Each object is automatically assigned an ID and index. Like Excel, attributes can be added to objects and displayed in columns. Objects can be filtered and sorted.

DOORS primarily differentiates itself from Excel with *links* between objects in different modules. Typically, one DOORS module represents a system or sub-system. Links are used to capture trees that span various system levels. A highly simplified representation of the Cordless Drill tree is presented in Fig. 5. In this figure, modules are represented by rectangles, objects as lines of text, and links as red lines between objects.

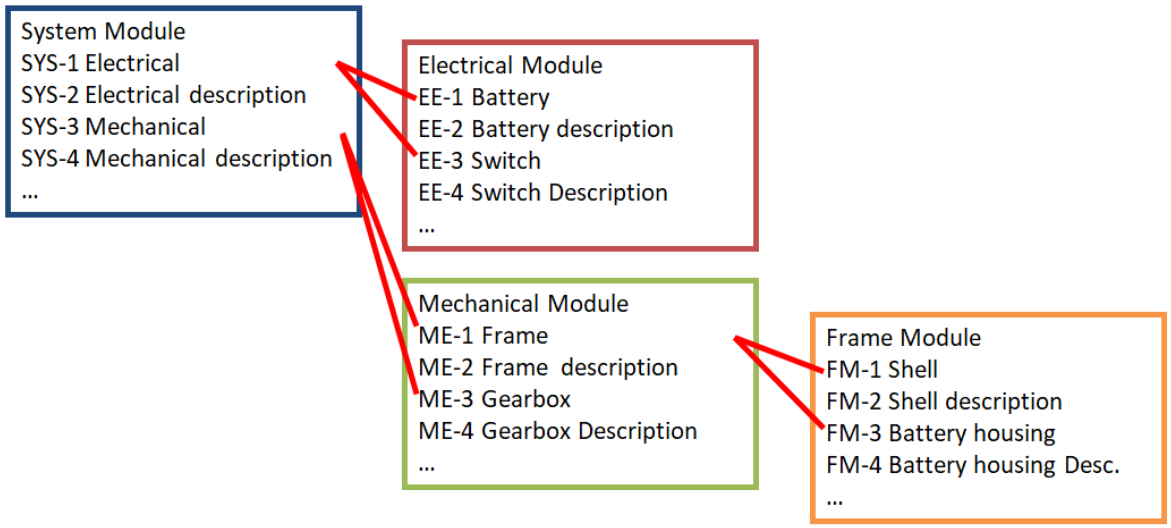

Fig. 5 Graphical Representation Cordless Drill Tree in DOORS

1. *Configuration Management*

DOORS contains strong configuration management by default. Unique identifiers are automatically enforced, even on deleted objects. Similarly, indexing is automatic.

DOORS also contains tools for evaluating version history. Each object stores every version of itself. Baselines can be taken of an entire module and can be compared with each other or with the current data.

2. *Visualization*

Within a single module, DOORS contains a variety of useful visualization techniques. As discussed above, DOORS can compare attributes, filter, sort, etc. However, data is typically stored in multiple modules. Except for the "Traceability Tree View," it is difficult to view elements from multiple modules at the same time. By default, parameters can be pulled across links, but only for one level of depth. If a user wishes to view an entire tree, or compare parameters across many modules, scripting, or exporting must be used. When used as a requirements management tool, DOORS is commonly exported into another tool for display and documentation.

DOORS supports robust import/export functionality, generating files that can be ingested into a number of applications for visual processing.

3. *Usability*

Doors requires little configuration in order to begin documentation. Creating nodes and attributes is simple. However, without scripting, creating links between nodes is extremely time-consuming. This process is repetitive and can easily lead to human error.

The learning curve for DOORS is typically steeper than the tools discussed above. While it is common among systems engineers, other disciplines are often unfamiliar with the tool.

D. Custom Solutions

When one of the tools discussed above does not adequately meet the needs of a program, a custom solution can be developed. In this paper, "custom solution" has a broad definition, referring to anything from a combination of two applications (e.g. Microsoft Word in conjunction with TeamCenter), to a custom-built application with its own GUI. This paper does not explore all custom solutions, but outlines benefits of several different approaches.

1. *Configuration Management*

A number of applications employ Structured Query Language (SQL) to access Relational Databases. This type of database stores large amounts of data without losing integrity. Relational Databases are designed with consideration for constraints such as identifiers and indexing.

Trees can also be stored in flat text files, which can be tracked in a repository such as Git [11]. Git provides comprehensive version history, and allows for simple version comparison.

A requirements management tool such as DOORS, discussed above, could also be used exclusively for version control. The data could then be exported to other tools for visualization.

2. *Visualization*

A wide range of creative visualization tools exist which can be employed to display the contents of a tree. Two examples are presented. Both examples are fully open source, and implemented in the open source programming language Python [12].

The Python module Treelib [13] stores and manipulates trees. It can also display them as names connected in a hierarchy (Fig. 6)**.**

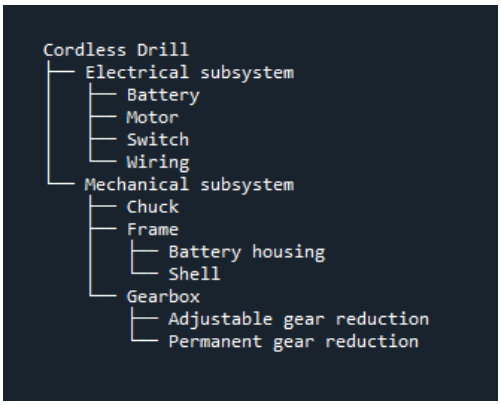

Fig. 6 Visualization of the Cordless Drill Tree using Treelib

While its scope is larger than trees, the python module Plotly provides a number of visualization tools that apply to trees. For example, the Cordless Drill nodes are displayed below in a Treemap, where the size of each node is

proportional to its mass (Fig. 7). This kind of visualization allows the viewer to see the structure of the tree while also providing meaningful data.

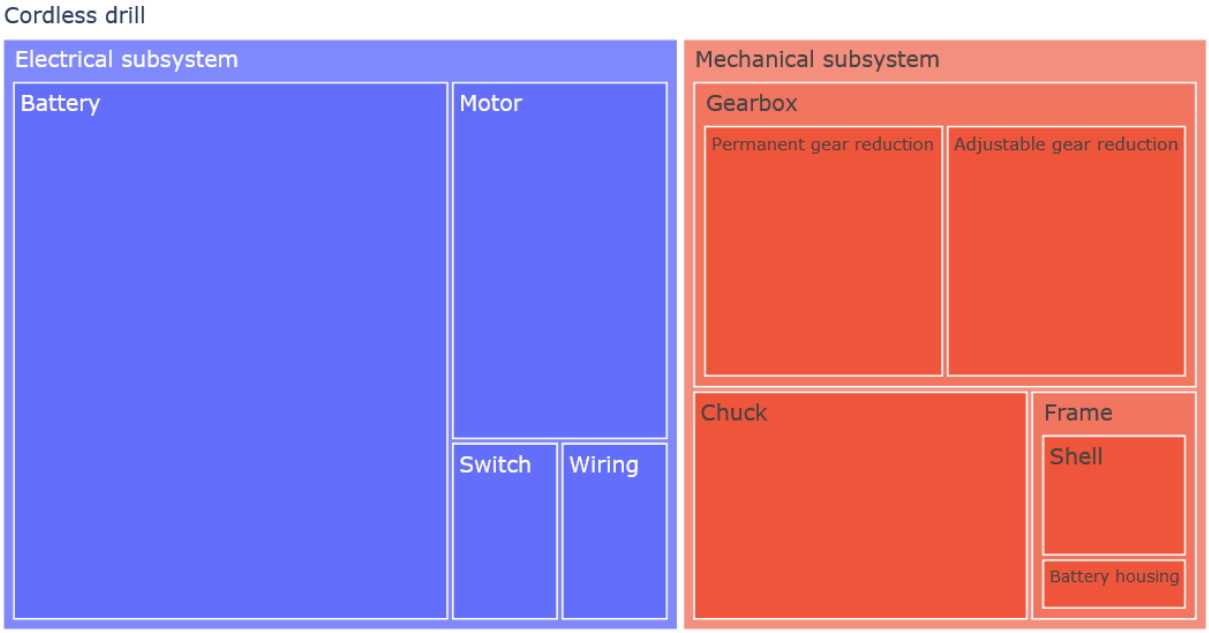

Fig. 7 Treemap of Cordless Drill Tree using Plotly

Scripting languages like Python provide a wide array of methods to display numerical data. Instead of exporting data into tables, it can also be analyzed directly.

3. Usability

The major obstacle to custom solutions is the difficulty in developing them, and the challenge associated with large numbers of people needing to use them. Creating solutions like those discussed above can require teams of experienced developers. Additionally, developmental work would need to be completed before the tool can be used. This can cause delays to other parts of the program if the tree to be documented is needed immediately. Once created, a custom tool will still be unfamiliar to people not involved in its development. This can result in a learning curve that is present throughout the life of the program.

E. Model-Based Systems Engineering (MBSE) Solution

No Magic's Cameo Enterprise Architect [15] was chosen to evaluate a potential MBSE solution. Cameo is a frontrunner in terms of MBSE tool adoption.

Cameo organizes information through a collection of objects called *elements*. Elements may have parameters associated with them, called *properties*. In addition to its properties, an element can be linked to other elements through *relations.* In Cameo, the collective set of elements, properties, and relations is referred to as *the model.* Diagrams are ways of visualizing the model. Multiple diagrams can be made which display the same elements. As a result, visualizations are easy to produce. Since each diagram is based on the same underlying model, there is no risk of various diagrams showing conflicting information. Any update to an element is reflected in updates to all diagrams containing the element.

This implementation allows for significant flexibility in modeling. For modeling the Cordless Drill tree, an approach was selected in order to implement the criteria described in Section II. The Cordless Drill tree was modeled using the following steps:

A stereotype was defined in order to capture each of the Cordless Drill parameters outlined in Table 1. Two numbering schemes were defined, one to capture ID and one for index. The numbering schemes were applied to the custom stereotype using a customization element. Nodes of the Cordless Drill tree were modeled as elements of type

class with the custom stereotype defined above applied. The nodes were structured into a tree using the *contains* relation. This type of relation was selected because it enables Cameo's automatic numbering feature.

The implementation defined above is not the only way to model a tree in Cameo. One of the advantages of MBSE is that there is often more than one solution to a problem.

1. Configuration Management

Each Cameo element is automatically assigned a unique Identifier. The identifier is a long alphanumeric string, and is hidden by default. Cameo also supports the use of custom IDs as properties. In the Cordless Drill tree example, an ID was defined as an integer with prefix "CSD-" for readability. Cameo assigns this ID when an element is created, and does not allow it to be modified. The IDs from deleted elements are not recycled, guaranteeing that no two nodes have the same ID.

Cameo does not natively support indexing, but can be configured to automatically generate indexes. The process for defining automatic numbering requires some experience with Cameo, but can be completed through the Cameo GUI, without the need for scripting or third-party software.

Cameo provides each element's history. This is useful for evaluating changes to a specific element. However, it is difficult to receive a high-level view of changes to all elements. This means that if a large tree is reworked, there could be a significant amount of overhead required to document the change.

2. Visualization

One of Cameo's strengths is its ability to display data in multiple formats. The following three visualizations (Fig. 8, Fig. 9, Fig. 10) were all automatically generated by Cameo.

For the two hierarchical diagrams, a single node was placed in a blank diagram. Using the Display Related Elements tool, Cameo automatically pulled children nodes into the diagram. The Automatic Layout tool was used to format nodes for viewing. Node parameters can be collectively displayed or hidden for the entire tree.

The flexibility of Cameo makes it easy to create a large number of useful diagrams with whatever level of detail is required. There is little overhead required to create or maintain diagrams.

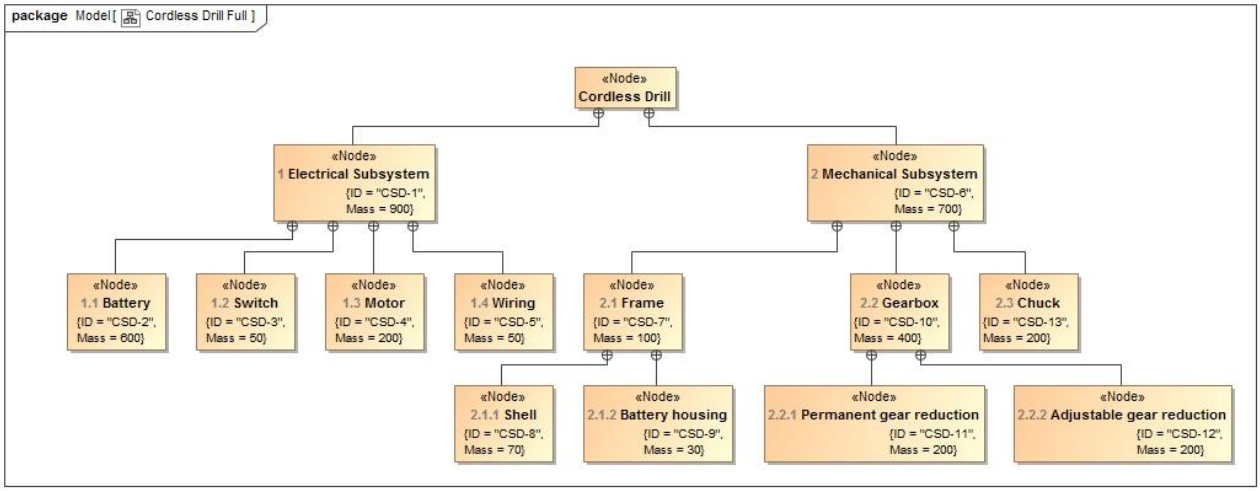

Fig. 8 Cameo Implementation of Cordless Drill Tree

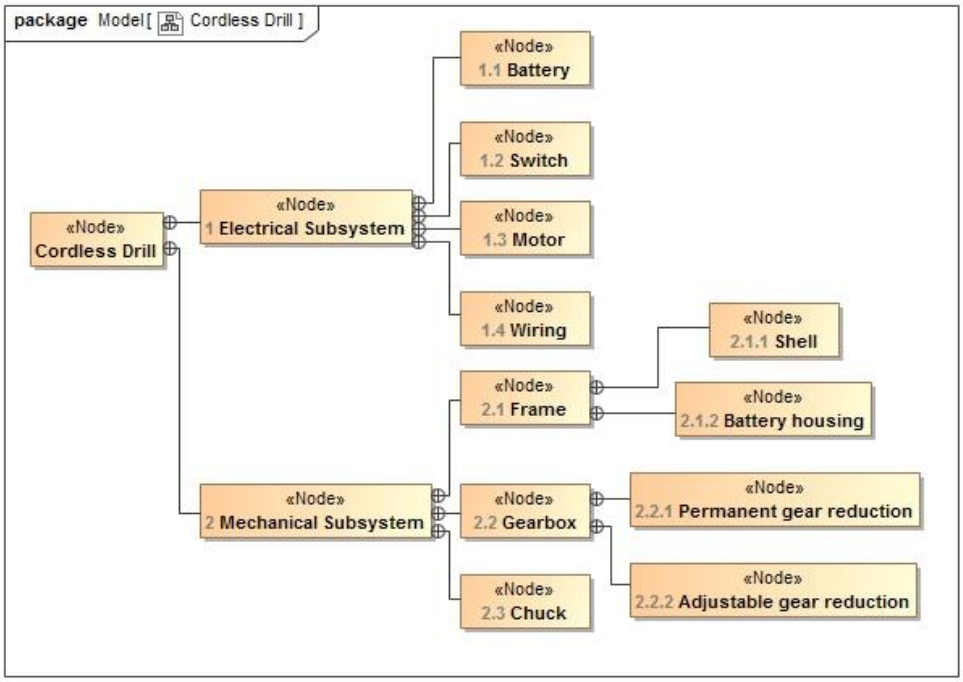

Fig. 9 Compact Implementation of Cordless Drill Tree

In order to compare parameters, Cameo is able to organize data into tables. Cameo considers tables a type of diagram. Like the diagrams above, they can be created automatically with little configuration. For example, the entire Cordless Drill tree can be pulled into a table with a single drag and drop operation. Once in table format, rows can be filtered and sorted, and additional columns can be added with functionality similar to Excel.

#	Name	Index	ID	Description	Mass
1	Electrical Subsystem	1	CSD-1	All componenets related to powering the screwdriver.	900
2	Battery	1.1	CSD-2	12 volt Nickel-cadmium battery pack.	600
3	Switch	1.2	CSD-3	On-off switch.	50
4	Motor	1.3	CSD-4	Brushed 12 volt Direct Current motor.	200
5	Wiring	1.4	CSD-5	Electrical wires connecting componenets.	50
6	Mechanical Subsystem	2	CSD-6	All componenets related to the physical structure of the screwdriver.	700
7	Frame	2.1	CSD-7	Primary structural elements of the screwdriver.	100
8	Shell	2.1.1	CSD-8	Plastic, houses all other components.	70
9	Battery housing	2.1.2	CSD-9	Contains battery.	30
10	Gearbox	2.2	CSD-10	Two-stage planetary gearbox.	400
11	Permanent gear reduction	2.2.1	CSD-11	Planetary reduction gearbox.	200
12	Adjustable gear reduction	2.2.2	CSD-12	Planetary reduction gearbox with adjustable slip.	200
13	Chuck	2.3	CSD-13	Single-sleve keyless chcuk	200

Fig. 10 Cameo Table Implementation of Cordless Drill Tree

Cameo supports import and export operations. While some configuration is needed, importing and exporting to and from CSV files is natively supported by Cameo.

3. Usability

Compared to conventional systems engineering documentation tools, Cameo requires a high cost of entry. Once Cameo is configured, trees can be defined with only a few operations. However, the task of configuring Cameo requires significant user experience.

In conventional tools, configuration is typically removed from the tool itself. Settings menus are available where the tool can be configured. In Cameo, the configuration of the model is contained within the model. Elements can be created which affect the appearance and functionality of other elements. While this provides significant flexibility, it

also means that configuration for new models can inadvertently break parts of the model that have already been defined. A program using Cameo must ensure that enough knowledge exists in its user base to prevent inadvertent conflicts.

Additionally, while Cameo can generate reader-friendly diagrams and tables, interacting with Cameo is not simple. It can take a significant amount of time for an engineer to learn Cameo and be able to effectively navigate within the tool. As an emerging technique, MBSE is not commonplace and many systems engineers do not have an MBSE background. As a result, programs attempting to implement trees using MBSE tools will face a steep learning curve as their engineers come up to speed.

V. Conclusion

A large number of tools are available to systems engineers for the documentation of tree data structures. Each of these tools has a number of advantages and disadvantages. The optimal tool for a given use case depends on the program, the users involved, the data to be documented, and numerous other factors. As a result, there is no simple answer when it comes to evaluating tools.

Although they are characterized by a steep learning curve, MBSE tools offer advantages not found in any convention systems engineering tools. Since MBSE is a relatively new pursuit, as more users could become familiar with the methodology, general MBSE literacy may increase. If this trend continues, the drawbacks of using MBSE will become less impactful. In the meantime, MBSE still presents several useful characteristics and deserves to be considered as an effective source of documentation for tree data structures.

References

[1] International Council of Systems Engineering (INCOSE) and Project Performance International (PPI), *System Engineering Tools Database* [online database], URL: https://www.systemsengineeringtools.com [retrieved 14 March 2021].
[2] XFMEA, Software Package, Ver. 2021, Hottinger Bruel & Kjaer Inc, Marlborough, MA, 2021.
[3] Fleck, Margaret, *Building Blocks for Theoretical Computer Science*, Version 1.3, University of Illinois Urbana Champaign, Champaign, IL, 2003, pp. 148-165.
[4] Walden, D. D., Roedler, G. J., Forsberg, K. J., Hamelin, D. R., and Shortell, T. M, *Systems Engineering Handbook*, Fourth Edition, Wiley, Hoboken, NJ, 2015.
[5] R.J. Cloutier (ed.), *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v. 2.3, The Trustees of the Stevens Institute of Technology, Hoboken, NJ, 2020. URL: www.sebokwiki.org [retrieved 14 March 2021].
[6] Microsoft Word 2010, Software Package, Ver. 14.0.7266.5000, Microsoft Corporation, Redmond, WA, 2010.
[7] Teamcenter, Software Package, Ver. 11.0, Seimens Product Lifecycle Management Software Inc., Plano, Tx, 2015.
[8] Microsoft Excel 2010, Software Package, Ver. 14.0.7266.5000, Microsoft Corporation, Redmond, WA, 2010.
[9] Beyond Compare 4, Software Package, Ver. 4.2.3, Scooter Software Inc., Madison, WI, 2017.
[10] IBM Rational DOORS, Software Package, Ver. 9.7.1, IBM, Armonk, NY, 2020.
[11] Git, Software Package, Ver 2.30.2, Linus Torvalds, 2021
[12] Python, Software Package, Ver 3.9, Python Software Foundation, Beaverton, OR, 2021.
[13] treelib.py, Software Package, Ver 1.6.1, Xiaming Chen, 2020.
[14] plotly.py, Software Package, Ver 4.14.3, Plotly, Quebec, Canada, 2020.
[15] Cameo Enterprise Architect, Software Package, Ver. 19.0, No Magic Inc., Allen, Tx. 2020.