2019 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
SYSTEMS ENGINEERING TECHNICAL SESSION
AUGUST 13-15, 2019 - NOVI, MICHIGAN

AUTOMATING THE VISUAL NARRATIVE: DYNAMIC ADORNMENTS AND LEGENDS

Michael J. Vinarcik, P.E., FESD

Chief Solutions Architect, SAIC, Detroit, MI

ABSTRACT

The Department of Defense's Digital Engineering Strategy (adopted in June 2018) has five goals:

1. *Formalize the development, integration, and use of models to inform enterprise and program decision making*
2. *Provide an enduring, authoritative source of truth*
3. *Incorporate technological innovation to improve the engineering practice*
4. *Establish a supporting infrastructure and environment to perform activities, collaborate, and communicate across stakeholders*
5. *Transform the culture and workforce to adopt and support digital engineering across the life cycle.*[1]

For this strategy to succeed, stakeholders must willingly participate in the cultural transformation and use system models as they are intended: as living, dynamic, integrated sources of information that communicate intent with rigor and clarity. Moving from disjointed documents and air-gapped information sources to a single source of truth is important. However, the explosive growth in system complexity is causing a new, unwanted emergent behavior:

> *In an information-rich world, the wealth of information means a dearth of something else: a scarcity of whatever it is that information consumes. What information consumes is rather obvious: it consumes the attention of its recipients. Hence a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it.*[2]

Because stakeholders are becoming overwhelmed with information from multiple sources, effective system modeling cannot solely focus on the creation of competently executed, integrated system models. It must also facilitate the creation of visualizations and derived products that allow stakeholders to efficiently identify and consume relevant information.

Citation: M. Vinarcik, "Automating the Visual Narrative: Dynamic Adornments and Legends," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2019.

[1] [1, p. 4]
[2] [2]

1. INTRODUCTION

1.1. The Transition from DISE to MBSE

As systems engineering transitions from Document-Intensive Systems Engineering (DISE) to Model-Based Systems Engineering (MBSE), both systems engineering and their stakeholders (program managers, subject-matter experts, and other program personnel) must adapt to the new paradigm to gain full value from the model effort.

One of the most pernicious issues facing the system engineer or modeler is the desire (from many individuals) to simply model or automate the *status quo*. While this is possible, in most cases, simply automating the production of traditional documents robs any modeling effort of much of its impact and value. Although model-generated documents have the advantage of being internally consistent (assuming the model was executed competently), they also typically reflect compromises made in the past due to the limitations of document-intensive methods.

When a program is fully transformed into a model-driven program, not only do stakeholders routinely consume the model directly and view it as the key deliverable (not the derived work products), but they also interact with the model in purposeful, novel ways that empower them to make effective decisions based upon an aggregation of relevant information. Stakeholders must learn to ask for work products that help them achieve their desired outcomes and modelers must not be afraid to suggest deviations from former best practices. Training Within Industry, which was used by the United States Department of War to fill personnel shortfalls in matériel production during the Second World War, understood that "Periods of standardization should be punctuated by periods of innovation, which are then translated into new standards." [1] It is time for systems engineers to deliberately adopt that approach to innovate and develop a new generation of best practices for model-based work products.

1.2. QED

One of the first steps in facilitating this innovation is working with stakeholders to uncouple their *needs* from the *forms* that were previously used to satisfy them. In *The NeMO Orbiter: A Demonstration Hypermodel*, the author posits the QED mnemonic to help structure these discussions:

"In traditional geometric proofs, QED was the final line a student wrote to indicate he was finished. *Quod erat demonstrandum* means "that which was to be proven" in Latin.

This acronym has been adapted for hypermodeling:

- What is the **Q**uestion we need to answer?
- How can we **E**xtract relevant information from the model?
- How should we **D**isplay it to stakeholders in a meaningful, easy to consume way?

By appropriately harnessing and answering these three questions, a competent system modeler is able to provide value for his program by allowing the program team's engineering staff to have insight into the system of interest.

As Frederick Brooks wrote:

"Show me your flowcharts and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowcharts; they'll be obvious. [2]"

Tables (or matrices or relationships maps) are often much more useful and clearer than diagrams. For this reason, competent modelers should carefully select *how* information should be presented (remembering that the model and its content is independent of its display). Subject matter experts, modelers, decision makers, and stakeholders may have different cognitive styles and preferences. Modelers must be willing to adapt without compromising model integrity. There is a fine line between fruitful challenge of the *status quo* (such as abolishing swimlanes) and fruitless conflict.

Observing QED principles requires that the modeler(s) find a way to represent all relevant

information in a well-defined structure so that it can be found and serve as the authoritative source of technical truth; for each piece of useful information:

- Should it be owned by an element?
- Should it be owned by a relationship?
- Should it be owned by a usage?" [3, pp. 4-5]

1.3. The Elegance Equation

These discussions must be held in the context of the elegance of the modeling effort.

"Every modeling effort has several factors that may be used to describe it:

η = Efficiency factor = output/input ($0 < \eta < 1$)

ε = Effectiveness factor = ability to accomplish intended outcome ($0 < \varepsilon < 1$)

φ = Elegance value ($0 < \varphi < 1$)

$$\eta\varepsilon = \varphi$$

Language, tool, and method each have their own contributions to this equation:

$$\eta_{language}\, \varepsilon_{language}\, \eta_{tool}\, \varepsilon_{tool}\, \eta_{method}\, \varepsilon_{method} = \varphi$$

Once the tool and language are selected, those terms are effectively constants… so any modeler is only able to directly influence $\eta_{method}\, \varepsilon_{method}$.

Therefore, productivity, effectiveness, and elegance depend heavily upon the methods used to construct the descriptive system model. One critical, inescapable fact is that every model element has a cost associated with its elicitation, creation, definition, and maintenance. Therefore, if a system can be described rigorously and completely with *n* elements, each $n + i$, where $i > 0$, element adds no value and only increases cost.

…A corollary of these principles is directly applicable to system modeling: *Don' t Create What You Can Infer or Query*. If these inferences and queries are unambiguous, leveraging them has a significant and direct impact on reducing the number of modeling elements." [3, p. 2]

As the modelers and stakeholders work to identify ways to satisfy the information needs of the latter, the modelers are best able to judge η and the stakeholders understand ε.

2. MAGICDRAW

This paper is focused on the use of MagicDraw, the Dassault Systèmes / No Magic system modeling tool (it is also available bundled with various plugins in the Cameo or Magic product lines). These techniques will work in any of these configurations (version 19.0 or later) and may also have analogues in the tools offered by other vendors (such as IBM's Rhapsody, PTC's Integrity Modeler, or others). All examples are rendered in the Object Management Group's System Modeling Language (SysML); the principles illustrated may also apply to Unified Modeling Language (UML) or Unified Architecture Framework (UAF) models. However, it is outside the scope of this paper to provide instructions for other tools and languages. Interested readers are directed to those tools' manuals, language metamodels, and subject-matter experts for assistance.

3. CUSTOMIZATIONS

MagicDraw exposes many of the capabilities, options, and settings that its developers use to construct the various profiles. This allows end users to make significant customizations to the model and the tool environment, including:

- Adding properties (including derived)
- Assigning icons to an element or relationship type
- Creating new element or relationship types
- Determining if/where elements appear in toolbars, palettes, and pop-ups
- Controlling how new properties are displayed within the tool's specification window.

Displaying customized elements and relations in the palettes streamlines model creation and creating clusters of custom properties in the specification window helps modelers and end-users easily interact with the content. The modest effort needed to add yields productivity benefits and a perception of a well-formed model. See Figure 1, Figure 2, and Figure 3.

4. STRUCTURED EXPRESSIONS

MagicDraw can execute a variety of languages (BeanShell, JavaScript, Groovy, Jython, and others) but one of the most powerful is its internal Structured Expression language. This language is mapped to the profile/language in use and writes complicated executable code using patterns driven by the user interface. It allows users who understand the tool's underlying data model and principles to rapidly create queries, tests, and other element and property operations. These may drive legends, validation rules, custom properties, custom tables, dependency matrices, and other enhancements.

Structured Expressions also include operations focused on:

- Collections
- Comparison
- Date
- Logical
- String

Particularly useful examples include:

- AnyMatch
- Exclude
- First
- Intersect
- IsEmpty
- Union
- IfThenElse
- Not

Interested users are directed to the Systems Architecture Guild YouTube channel (http://videos.systemsarchitectureguild.org) for detailed examples and how-to explanations.

5. DYNAMIC LEGENDS

Dynamic legends are a tool feature that allows MagicDraw to adorn elements and paths (in essences, shapes and lines) with custom fill, font, color, icon, and other properties. Unlike manually applied legends, which allow users to apply these graphical enhancements on case-by-case basis, dynamic legends automatically apply adornments based upon rules established by the modeler.

These legend items may be prioritized (to ensure they are applied in a given order) and are applied by *element condition.* These *conditions* are tested, and the adornments are applied to each element for which they are true. See Figure 4.

Although dynamic legends are most often used to adorn diagrams and tables that are intended for stakeholder use, they may also be used to assist modelers. Carefully crafted dynamic legends can be used as a rapid quality check if the creation of a validation suite is not warranted. Applying a dynamic legend temporarily to a diagram can highlight elements and paths that did not respond to the adornment rules; this allows the modeler to correct errors and then remove the legend.

6. PROPERTY-DRIVEN ICONS

MagicDraw provides a provision for changing an element's icon based upon its properties. The *iconholder* stereotype may be applied to an *enumeration*. Different icons may be assigned to each literal within the enumeration. Stereotype tags that are typed by those iconholder enumerations now "drive" the icon for the element to which they are applied. When the value of the enumeration is changed, the icon is automatically updated to match. See Figure 5.

7. DISPLAYING AT-RISK ELEMENTS

Derived properties may be used to help identify at-risk elements. For example, if cost and mass budgets are established, the mass and cost reserves may be automatically calculated. Dynamic legends may be used to highlight these when displayed in tables or diagrams (e.g., shading the mass reserve cell red if it is less than zero or coloring a part property yellow if its cost reserve is less than 10%). See Figure 6.

Custom tables may also be created that only contain at-risk elements. Structured expressions may be used to manage which elements are included or excluded. This enables stakeholders to

focus their attention and eliminates the need to sort exhaustive tables or export them to Microsoft Excel for filtering.

8. VALIDATION SUITES

MagicDraw provides a validation engine that may be used to automate quality checks and enforce style guides. Care should be used not to overuse active (constantly running) validation suites, since these can hamper performance in large models. However, the use of on-demand validations (and those run automatically before committing the model) are encouraged.

Modelers may use the structured expression language to design tests that make up each rule; these should return a Boolean value. If the result is *True*, the element passes validation. If the result is *False*, the element fails the test and the failure is displayed to the user.

It is a best practice to include details in the validation rule that help the user correct the error (e.g., "All use cases must be traced to a user need."). Different levels of severity may also be assigned (from *debug* to *fatal*); this allows some granularity in response.

9. MODEL HOUSEKEEPING

One of the hallmarks of a well-crafted, competently executed model is that it is *tidy*. Elements are thoughtfully organized, content is easy to find, and no spurious elements are present. It takes constant effort from the modeler(s) to keep a model in this state, but the results facilitate ease of use.

One best practice is to organize elements, tables, diagrams, and other content by considering its context. For example, if a table lists the interfaces associated with a block, it is a good practice to have the block own the table. In this way, all the customized reporting elements are easy to find (they are owned by the element on which they provide insight). This also allows the use of context-scoped tables (this allows a simple copy/paste/rename of an existing table and it automatically updates to reflect information about its new owner). See the associated video at http://videos.systemsarchitectureguild.org for details.

10. NAVIGATION BEST PRACTICES

MagicDraw allows the creation of hyperlinks between elements; a double-click on the element follows the link. Careful use of this feature can simplify the navigation within a model (e.g., hyperlinking a package to the most important element or diagram within it or hyperlinking an activity diagram's initial node to its parent state machine). URLs may also be associated with elements; this allows the model to serve as a directory to external content (such as standards or source content).

Elements within the containment tree may also be set as "Favorites" by clicking the star at the top of the containment pane. This is a feature best used sparingly but it can facilitate personal navigation (Favorites are not shared globally).

Finally, content diagrams and package diagrams may be used as model "switchboards" to help users navigate them. See Figure 7.

11. CONCLUSION

The pragmatic use of dynamic adornment and legends can automate the visual narrative, ensuring that stakeholders are guided to relevant information and that these cues are always in synch with the underlying model content. Appropriate use of these techniques will help overcome the difficulties stakeholders encounter in finding relevant model content to support analyses and decisions.

12. REFERENCES

[1] D. Dinero, Training Within Industry: The Foundation of Lean, New York: CRC Press, 2005.

[2] F. P. Brooks, Jr., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, 1975, 1995.

[3] M. J. Vinarcik, "The NeMO Orbiter: A Demonstration Hypermodel," in *Ground Vehicle Systems Engineering and Technology Symposium*, Novi, 2018.

[4] Office of the Deputy Assistant Secretary of Defense for Systems Engineering, "Department of Defense Digital Engineering Strategy," Department of Defense, Washington, 2018.

[5] H. A. Simon, "Designing Organizations for an Information-Rich World," in *Computers, Communication, and the Public Interest*, Baltimore, The Johns Hopkins Press, 1971, pp. 40-41.

APPENDIX A: Figures

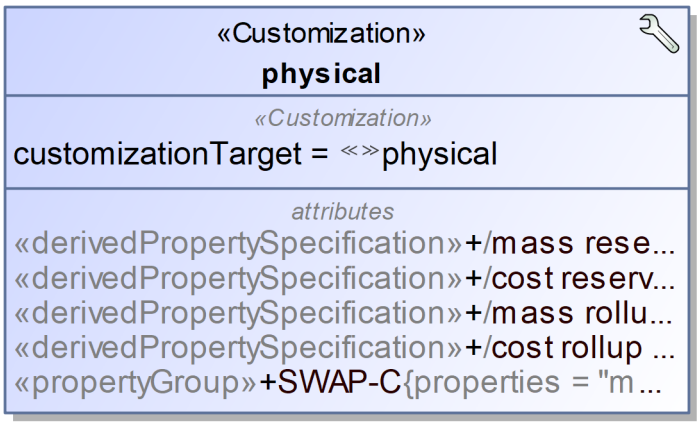

Figure 1:Customization Example

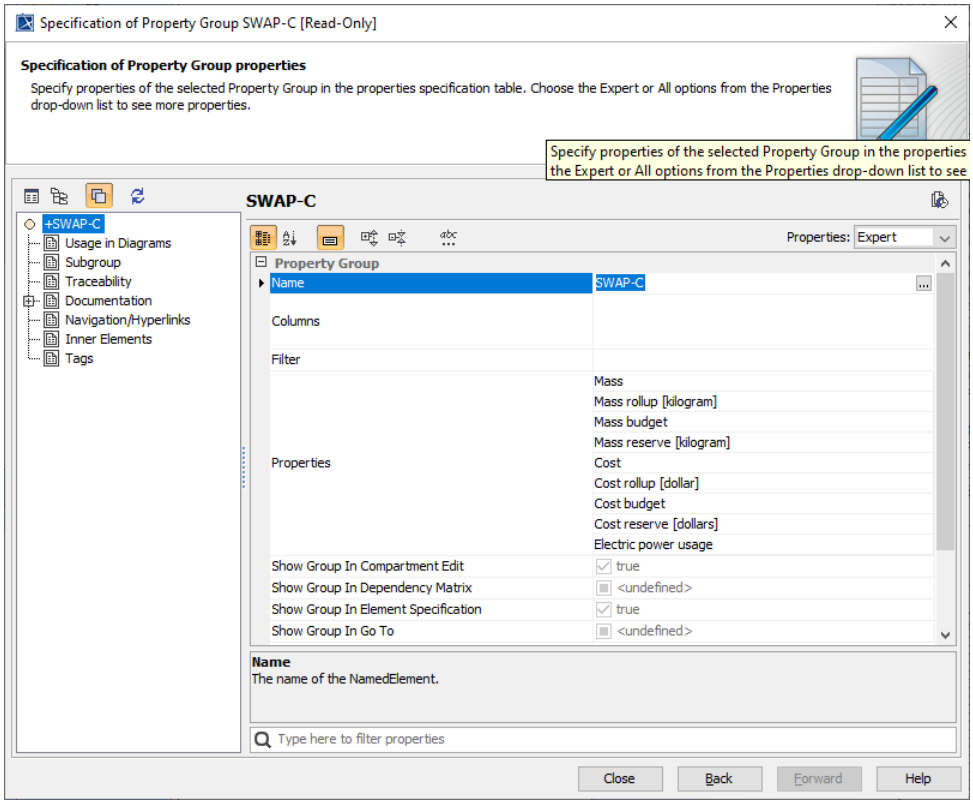

Figure 2: Property Group Example

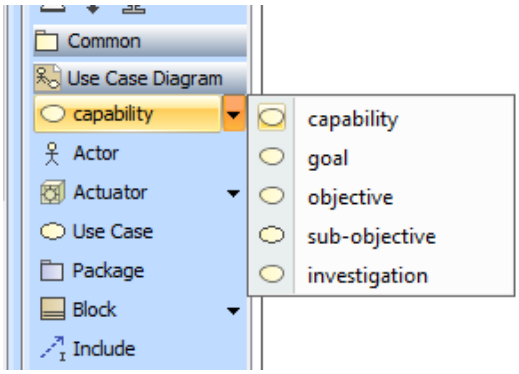

Figure 3: Custom Palette

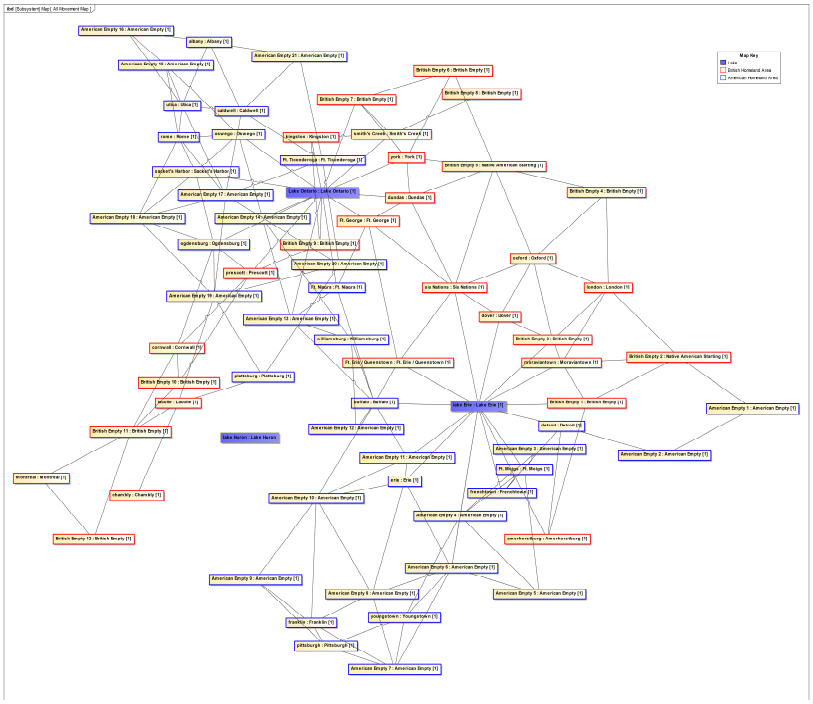

Figure 4: Dynamic Legend Example

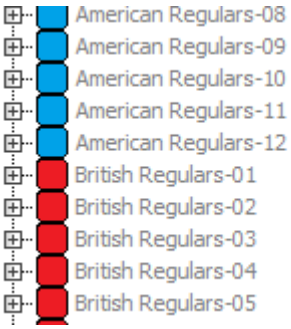

Figure 5: Notional IconHolder Example

#	△ Name	Has Parts	Mass	Mass rollup [kilogram]	Mass budget	Mass reserve [kilogram]	Cost	Cost rollup [dollar]	Cost budget	Cost reserve [dollars]
1	A	true	5	62	60	-2		50	200	150
2	A1	false	8	8	10	2	5	5	5.1	0.1
3	A2	false	10	10	11	1	10	10	10	0
4	B	true		28	50	22		24	25	1
5	B1	false	8	8	10	2	8	8	12	4
6	B2	false	20	20	20	0	16	16	16	0
7	C	true		18	20	2		18	20	2
8	MCES Example System	true		108	125	17		92	100	8

Figure 6: Notional Cost and Mass Reserve Table

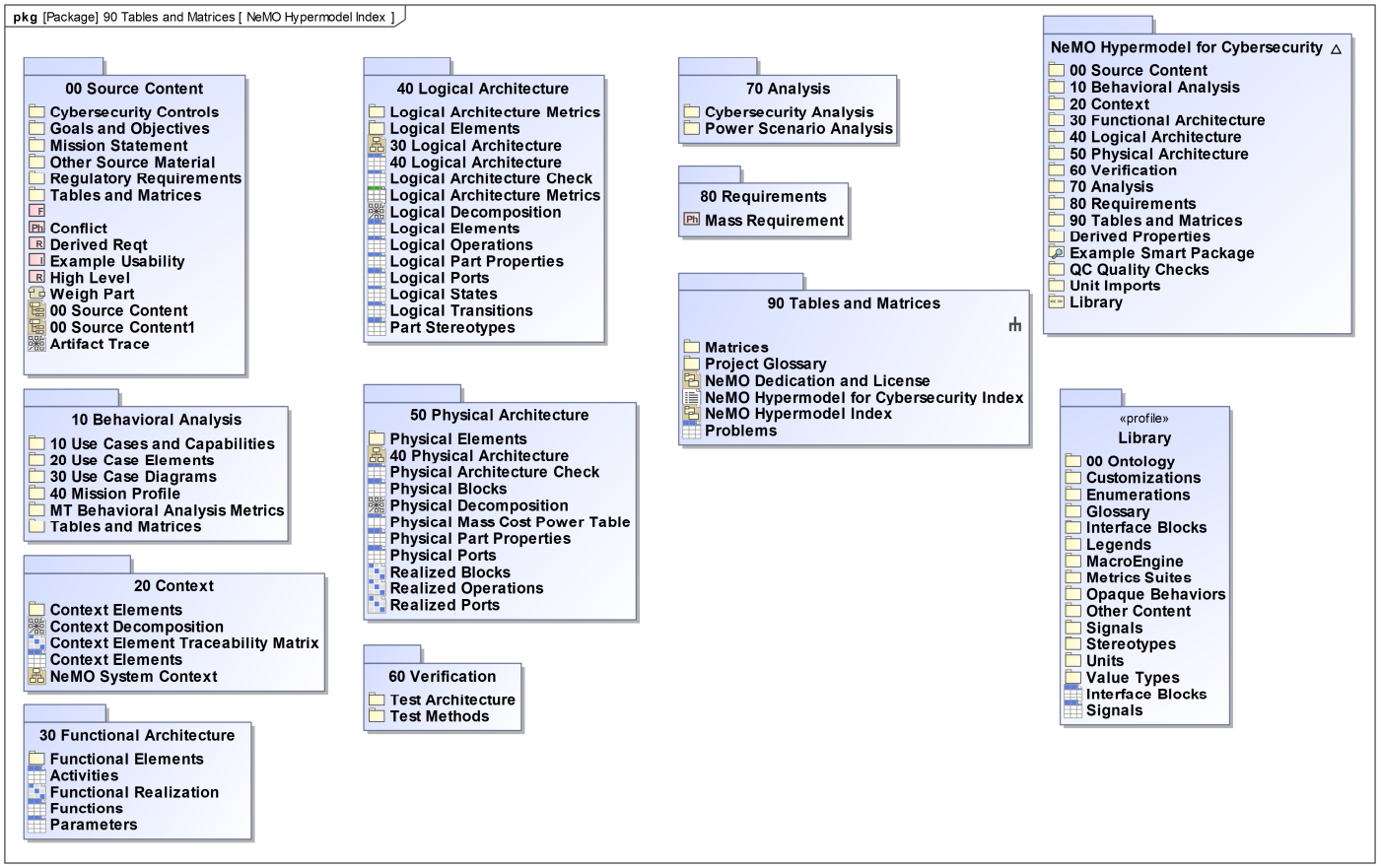

Figure 7: Notional Package Diagram

.