Efficient Management of Configurations in the Model-Based System Development of a Common Submarine Combat System

Steven W. Mitchell
Lockheed Martin MS2
9500 Godwin Dr.
Manassas, VA 20110 USA

***Abstract*[1] - Efficient management of the product configuration process is a challenge in the evolution of any industrial scale product family. This is particularly true with current standards-based system modeling tools, as the standards themselves are just beginning to address this problem in a scalable fashion.**

In the case of the SWFTS common submarine combat system, dozens of product configurations must be managed in parallel, with many of those baselines being updated several times a year. To handle this task a new SysML modeling technique has been developed. It extends the concepts of libraries with SysML Catalogs to bound the complexity of the configuration task, improving the quality and efficiency of the systems engineering process.

General Terms: Systems Engineering, System Evolution

Additional Key Words and Phrases: Product Family, Model Based Systems Engineering, SysML, SysML Catalog, Configuration Management

I. INTRODUCTION

As described greater detail in [1], the Submarine Warfare Federated Tactical System (SWFTS) Common Submarine Combat System (CS2) is comprised of 39 subsystems which are configured to support six distinct submarine classes. As the term "Federated" implies [2, 60-61], these heterogeneous subsystems are developed autonomously by various organizations, each with their own requirements, architectures, and funding streams, and then loosely coupled through semantically and syntactically controlled interfaces. The subsystems consist of a large number of software modules hosted on a smaller number of common Commercial Of-The-Shelf (COTS) hardware components augmented with a substantial number of subsystem-specific – and in some cases variant-specific – hardware components. Much of the software is also COTS[2].

Host platform variability means that the CS2 is in reality a product family with different variants for each class, flight, and in some cases individual submarine. While the core components that make up a given baseline release of the CS2 are common across all host platforms, there are host-specific variants that have to be developed, integrated, installed, supported, and then replaced with the next version on a regular upgrade cycle.

The CS2 integration program – the SWFTS program – handles the hardware side of the COTS management problem with biennual baseline updates that allow each ship-set of equipment to be state-of-the-market when it is installed. This process is referred to by the program as Technology Insertion (TI). These TI baseline changes occur in even years, leading to the nomenclature TI08, TI10, etc.

To provide incremental improvements in system capability and to avoid COTS software obsolescence issues, the application software running on this hardware platform is also upgraded biennually. For historical reasons [3] this process is called Advanced Processor Builds (APB). The APB baseline updates occur in odd years, leading to the nomenclature APB09, APB11, etc. This TI/APB cycle is in reality a double-helix collaborative spiral development life cycle [4] where the hardware and software spirals are one year out of phase with each other. The APB process is managed not by the SWFTS program, but by the Program Executive Officer (PEO) for Integrated Warfare Systems (IWS) (PEO IWS5A), which provides the SWFTS systems engineering products to the individual acquisition programs that produce the subsystems that are integrated by SWFTS.

Given that there is an unavoidable coupling of the APBnn and TImm updates, fleet-wide baselines installed aboard ship are referred to as TImm/APBnn. The overlap of the TI and APB update cycles leads to an annual change in baseline installations. Thus submarines that are upgraded in 2011 will receive TI10/APB09, while those upgraded in 2012 will receive TI10/APB11, etc. These annual baselines must go through the full systems integration process to ensure that the system installed aboard ship is operationally suitable, effective, and interoperable.

The baseline management problem does not end when an annual baseline and all of its variants are installed and certified. The changes between the older, still supported baselines and the new baseline are assessed for

[2] In this usage, COTS includes Free or Open Source Software (FOSS) as well.

applicability to SWFTS CS2 systems already in the fleet. When those changes fix operational problems in the deployed systems they may be rolled into incremental updates for those previous baselines. This leads to the ongoing evolution of up to four baseline trees.

The engineering problem of managing the evolution of the CS2 goes beyond simply tracking bills of materials for the various baselines. Since this system is installed on a submarine, power, cooling, mass properties, and physical layout are tightly constrained and must be tailored to the idiosyncrasies of the various host platforms. Other materials characteristics that have safety implications for the crew under casualty conditions, such as the presence of various hazardous materials, must be tracked. These systems process classified information, and so must be documented, tested, and certified to the appropriate standards for information assurance (IA). The CS2 has direct impacts on safety of ship and is a core part of a weapons system: both of those things require additional documentation, verification, and certifications. The CS2 interfaces with larger military command, control, communications, computers, and intelligence (C4I) networks, which entails interoperability testing and certification. All of the information supporting these various certifications and accreditations must be managed for each variant of the CS2.

As discussed in detail in [5] and [6], the SWFTS program is currently converting from a traditional document-centric systems engineering process to a model-based systems engineering (MBSE) [7] process. The focus of this paper is the management of variation in the Systems Modeling Language (SysML)[3] system models that support the new SWFTS MBSE process, and efficient construction and documentation of CS2 product family variants.

II. Considerations For Modeling Variability In System Configurations

Efficiently representing system variation is a key issue in applying MBSE to the systems engineering of product families. This is important both to minimize duplicative data that needs to be maintained and synchronized within the system models [8] and [9], and to minimize the conceptual complexity of the system model [10]. There is a large body of literature on the subject of managing variation, much of it coming from the software product family community.

Building on Jacobson, Griss, and Jonsson's notion of variation [11], Webber [12] introduced the Variation Point Model. She describes this utilizing an extension to the Unified Modeling Language (UML) with Gomaa in [13] for variation involving inheritance, parameterization, information hiding, and call backs. Bachmann, et al. [14] generalize this notion to accommodate variation points throughout both the architecture and the design, although without providing a specific implementation in any standard modeling language. Critically, they observe that implementing this approach for managing variation in realistic product families will require a very sound methodology for configuration management.

De Oliveira, Gimenes, and Huzita [15] developed their own design methodology utilizing variation points. This methodology requires both an external requirements database and UML notes embedded in diagrams to document key aspects of product family variation, which has the potential for creating problems in automated model processing (such as constraint checking). It also raises questions about model portability between tools as different UML modeling tools treat the content of comments quite differently.

Compliance with industry modeling standards such as UML and SysML [16] is a critical consideration, because SWFTS is not a program to invent new modeling tools and languages but a program to build submarine combat systems. Both engineering best practices and customer direction are to minimize the likelihood of getting locked into any particular tool or tool vendor, and conformance to standards is the best way to avoid such lock-in. Where the current standards are inadequate to the task, as in variation point modeling, this means working with the standards organizations to extend the standards, and with multiple tool vendors to incorporate those updated standards in their products.

A recurring theme in the software product line variability literature is of variability as a means for deferring design decisions until late in the implementation process. This is a central concern of van Gurp, Bosch, and Svahnberg [17, 18], who systematically address the problems of identifying the most appropriate technique for implementing a specific kind of variability in a given software product family. This makes sense for product family lines designed from scratch, but seems less applicable to programs such as SWFTS where the problem is managing variability in the context of on-going evolution of a product family which essentially coalesced. However, many of the heuristics identified in [18] for the effective use of variation points are directly applicable to the problem of managing the evolution of the CS2.

The concept of catalogs as an organizing mechanism is included in the draft Service Oriented Architectures Modeling Language (SoaML) standard [19]. Unfortunately, the SoaML standard has not progressed beyond beta status since introduction in April 2009, and there is no clear timeline for its finalization.

[3] The current specification of the SysML is at http://www.omg.org/spec/SysML/

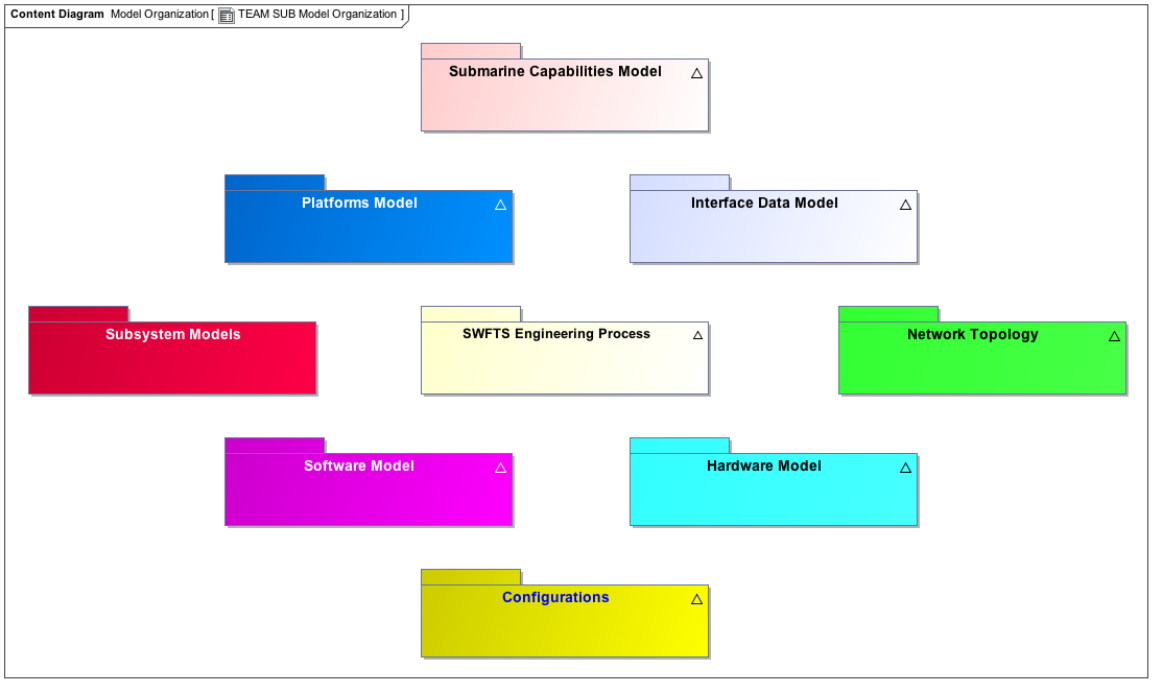

Figure 1 Top-level organization of the CS2 Engineering Data Model

III. Structure of the CS2 Engineering Data Model

The high level design of the CS2 Engineering Data Model (CS2EDM) is described in [5]. As illustrated in Figure 1, the CS2EDM is divided into a collection of nine smaller models. This figure depicts only the top-level partitioning of the system model – there are additional models at lower levels as needed to support the evolution of the CS2. In particular, the *Subsystem Models* package contains a package for each category of subsystem (SONAR, Electronic Support Measures (ESM), Navigation, Communications, etc.), which in turn may eventually contain a complete subsystem model organized like the overall CS2EDM for each alternative implementation of that subsystem. Thus the ESM package might contain a subsystem version of the CS2EDM for the AN/BLQ-10 electronic surveillance system used on the United States Navy Virginia class, one for the Condor CS-5600 system used on the Royal Australian Navy Collins class, etc.

The words 'may' and 'might' are used deliberately, as some of the subsystem implementations are either COTS or other types of Non-Developmental Items (NDI). Commercial vendors are unlikely to provide this level of design documentation. Further, the degree to which even government subsystem acquisition programs embrace MBSE will depend on multiple factors, not the least of which is a clear demonstration of the return on investment of MBSE vs. traditional document-centric systems engineering by the SWFTS program.

Aside from the Subsystem Models, the CS2EDM will be populated only with sufficient information to support the SWFTS systems engineering process, along with additional information to support the requirements of the customer acquisition process. This process, the Joint Capabilities Integration and Development System[4], requires specific architectural information presented in a particular format. As discussed in [1], there is substantial overlap between that architectural information and the information used by the SWFTS systems engineering process, so the SWFTS customer has directed the CS2EDM be populated sufficient additional information, particularly in the Submarine Capabilities Model, to satisfy both programmatic needs.

Clearly, selection of a particular implementation of a given subsystem is a key point of variability in the CS2, but associated with that selection are model elements scattered across the entire CS2EDM. Thus the capability for launching Tomahawk cruise missiles from specific submarine platforms brings with it higher echelon network security requirements that impact the partitioning of the combat control LAN into network enclaves in the Network Topology model. The unique data

[4] https://dap.dau.mil/aphome/jcids/Pages/Default.aspx

requirements of the Tomahawk also impact the interfaces between various subsystems, etc. documented in the Interface Data Model. Since the Tomahawk is a certified weapon system, there is associated hardware and software that impacts the Software Model and the Hardware Model. Thus adding a single element to the Submarine Capabilities Model can have ripple effects throughout the CS2EDM. This ripple effect might be an argument that the overall model organization is flawed, as it does violate the heuristic in [18] that variations should be associated with a minimum of variation points. However, in the imperfect real world other organizational considerations often trump theoretical optimality.

The specific partitioning of the CS2EDM depicted in Figure 1 was selected both to support the existing SWFTS program IPT structure, which is a reflection of organizational considerations both within the SWFTS program and in the larger Team Submarine acquisition community, and to facilitate the construction and evolution of SWFTS configuration baselines. This structure both simplifies programmatic coordination and the configuration management problem, since each of the sub-models is the focus of attention of only one or two Integrated Product Teams (IPT), and reduces the effective memory footprint of the model being used by the IPTs.

This later feature is an important if transitory practical consideration as it enables a common office computer to load the modeling tool and the sub-model for editing without being bogged down by constantly paging virtual memory to disk, with deleterious effects on engineer productivity. The specific model size that triggers paging will change over time as the standard configuration of office computers and the modeling tools evolve, but given the scope of the CS2EDM the basic consideration of managing computer resource demands will likely persist.

While the CS2EDM is partitioned into multiple sub-models, as demonstrated in the Tomahawk missile example above those partitions are not independent. Figure 2 illustrated the key relationships between the various subordinate models. In particular, it indicates how system configurations, a primary product of the SWFTS systems engineering process, are built upon the rest of the CS2EDM.

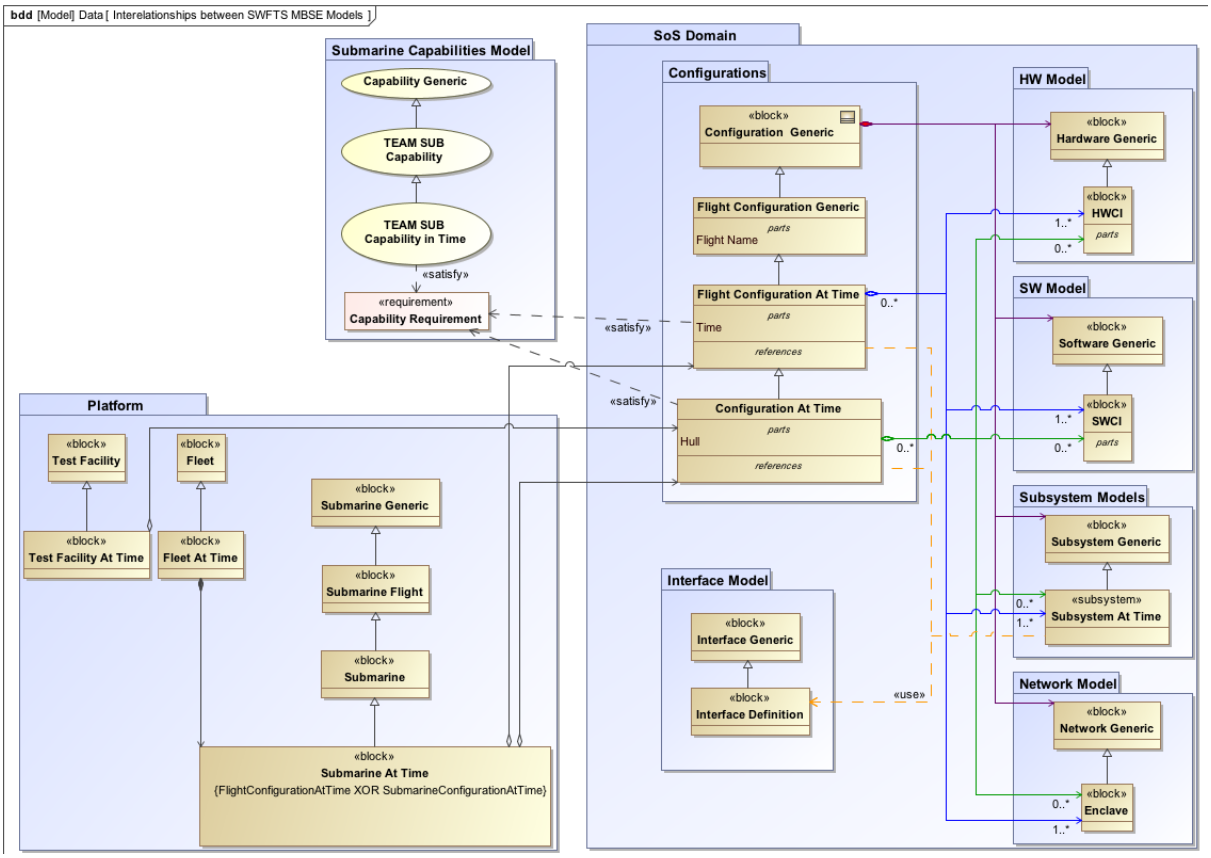

Figure 2 Interrelationships between the elements of the CS2 EMD sub-mod

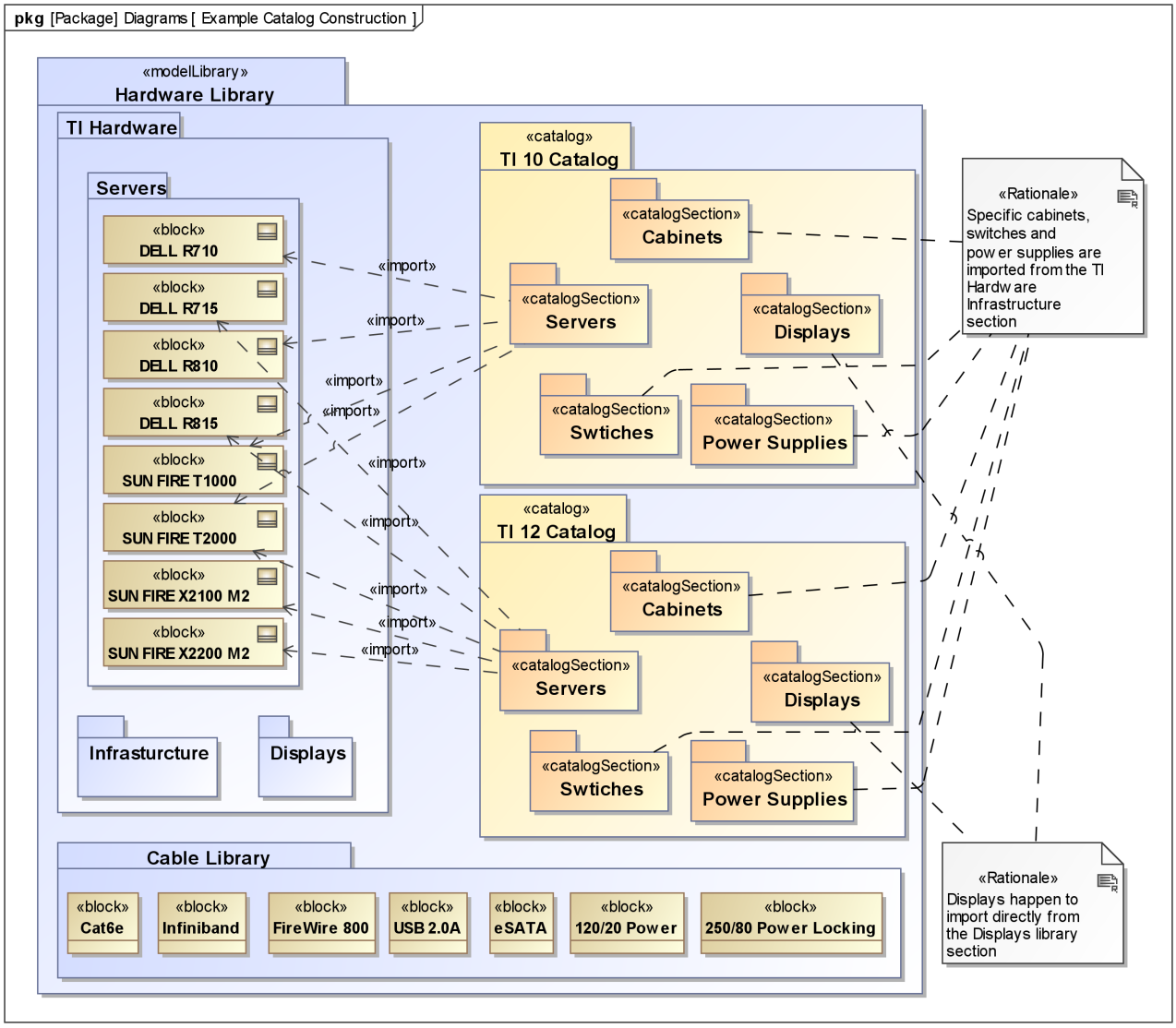

Figure 3 Constructing catalogs of approved components from libraries of available components

IV. Constructing Configuration Baselines

Constructing those baseline system configurations is a technically challenging task. Given the large number of baselines that must be managed, the total number of software and hardware components, interface specifications, etc. that are used in one or more baselines at any given time is quite large. For an engineer sitting down to construct a new baseline, the need to hunt manually through dozens of server and switch models or tens of hundreds of versions of interface specifications would be so laborious and error-prone as to defeat the productivity and quality objectives of introducing model based systems engineering to the SWFTS program. Traditional UML/SysML modeling tool support for variation points is sufficient for the toy problems used in vendor demonstrations, but totally inadequate for an industrial problem of this magnitude.

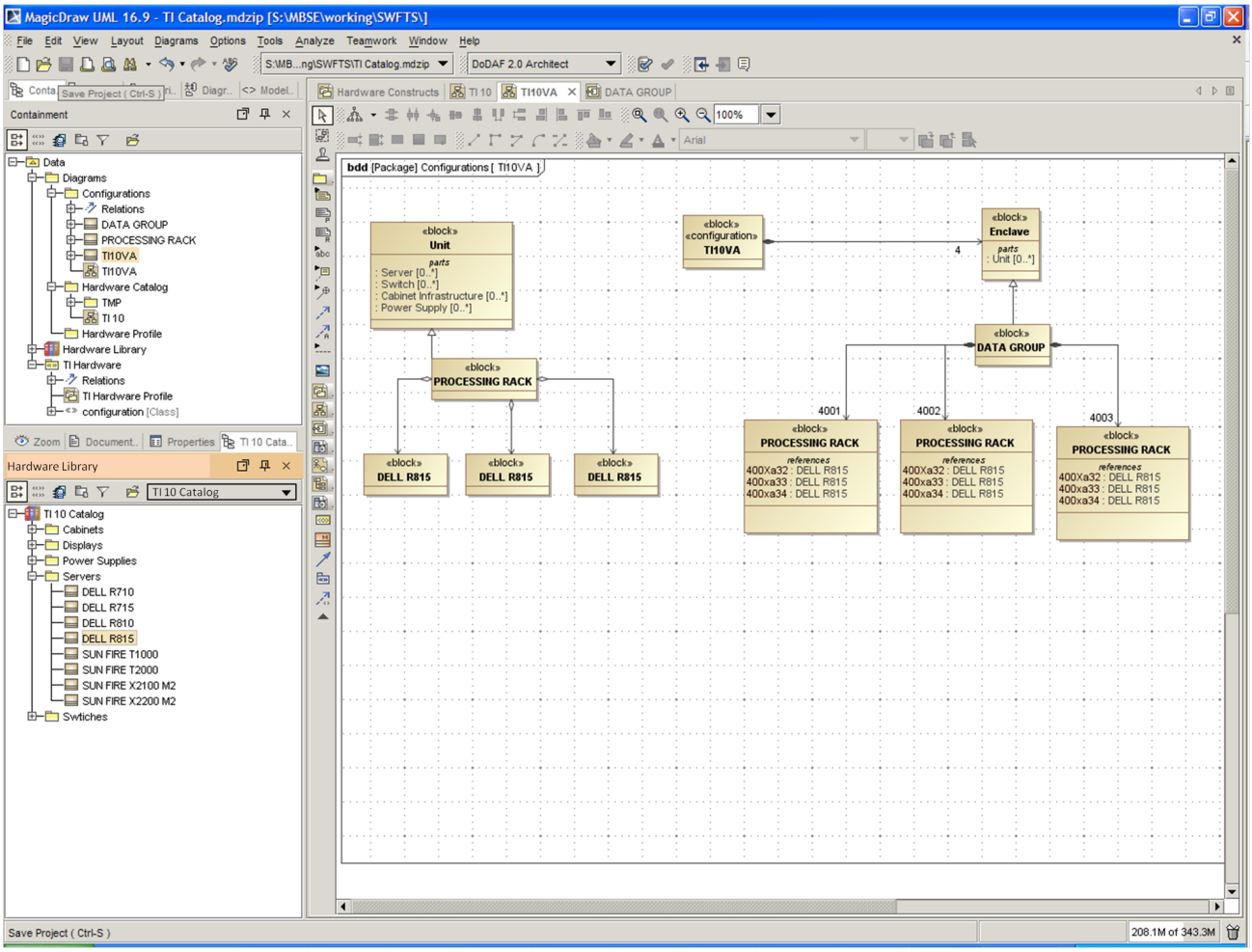

Figure 4 Constructing a system configuration from catalogs of approved baseline components

Thus it is necessary to create some mechanism for appropriately restricting the scope of objects available to the engineer constructing or modifying a given baseline. If the totality of servers, switches, displays, etc. that are included in the CS2EDM Hardware Model are considered as a library of candidate hardware components, what is needed is a catalog containing only those components which are approved for baseline use in the configuration at hand.

It is precisely this concept of libraries and catalogs that is currently being used by the SWFTS program to model the CS2EDM. As shown in Figure 3, the approved subset of servers from the list of all servers used in any existing or planned version of the CS2 is imported into a catalog for a specific baseline (TI10 or TI12 in the example). Similarly, these catalogs are populated with other hardware components approved for those baselines. Similar catalogs are constructed for COTS and application-specific software components, for interface specifications related to specific capability requirements, etc. Each catalog restricts the scope of the configuration to those components approved for the specific TInn/APBmm baseline.

The actual process of constructing a baseline from a set of catalogs is shown in Figure 4. In this case a variant configuration from the TI10/APB09 baseline is being constructed for a specific class of submarines. The TI10 hardware catalog is open in the browser on the left side of the screen capture, and specific servers are being configured into processing racks that will be installed on the submarines. It should be noted that the specific hardware components shown in Figure 3 and Figure 4

are purely illustrative, as these specific screen captures were generated while working out the library and catalog modeling technique rather than during production engineering. But the tool support shown in Figure 4 is critical to the productivity and quality gains that are projected for the conversion of SWFTS from a document-based to a model-based systems engineering process. In this case that support is currently unique to a particular tool vendor (No Magic, Inc. with their tool MagicDraw UML5), but that sort of user-interface feature is likely to be imitated by other tool vendors as a natural side-effect of competition, so its use is not considered as creating a high risk of tool vendor lock-in.

V. Future Work and Conclusions

The structure of the CS2EDM described in this paper and in [1] is firm, although of course the details are likely to evolve. The SWFTS program is currently populating that model with all of the information required to build a full CS2 baseline. Once the model is fully populated, the library and catalog methodology for constraining the construction of configurations described above will be used to construct a full set of baseline and variant configurations in parallel with current document-centric systems engineering process to validate the model and the model-based systems engineering process.

As mentioned in Section III, the scope of the current project is limited to populating the CS2EDM with just enough data to support SWFTS systems engineering and the US Department of Defense Joint Capabilities Integration and Development System (JCIDS) acquisition process. In particular, subsystems are being treated to the maximum extent practical as black boxes. As the projected benefits of moving to a model-based systems engineering process materialize, it is anticipated that at least some of the subsystem program offices will join in the migration to MBSE, and the scope of the CS2EDM will expand.

As the recent SWFTS experience with variation points shows, modeling language and tool support for this important concept is still quite rudimentary. As more details are worked out and additional dimensions of the data are filled in, it is likely the CS2EDM will stretch the bounds of the current UML and SysML language standards, and potentially will require inventing extensions to those languages. This will be done in collaboration with the developers of the respective standards so that those extensions are defined in the spirit of the languages, and can be incorporated in future versions of those evolving standards. Along that line the SysML Revision Task Force is considering OMG Issue Number 13928, which proposes adding the `ElementGroup` construct to the SysML standard [20]. If adopted, this will provide much of the functionality of the SysML Catalog construct discussed above. If and when that change occurs in the standard and is implemented by the tool vendors, the SWFTS program will review its use of the Catalog construct.

In addition to building the CS2EDM, an ecology of tools is being defined and developed around the model to make it efficient for the many IPTs that collaboratively evolve, implement, and support the CS2 to access those portions of the model that they need without requiring all of those engineers to become proficient with the tools, languages, and schemas used to build the model. Most of those engineers are currently using various Microsoft Office products to manage their specialized nexi of the overall information space that will be subsumed into the federated CS2EDM. It is anticipated that various web services will be built around the UML/SysML model to provide familiar interfaces to those nexi and to eliminate the cost of climbing the learning curve that would be necessary if every engineer supporting the CS2 were required to become expert in a UML/SysML modeling tool.

[5] http://www.magicdraw.com/

Acknowledgments

This research was supported by NAVSEA contract N00024-06-C-6272. The modeling techniques described in this paper were the result of vigorous discussions with Mr. Greg Bussiere and Mr. Shawn Murphy of the Naval Undersea Warfare Center, with Mr. Robert Sylvia of Sonalysts, with Mr. Daniel Brookshier of No Magic, Inc., and with Mr. Sanford Friedenthal, Mr. Brandon Gibson, and Mr. Mathew Rhodes of Lockheed Martin. The useful results are the fruits of that collaboration, while the errors remain my own.

References

1. Mitchell, Steven W., "Complex Product Family Modeling for Common Submarine Combat System MBSE," Third International Conference on Model Based Systems Engineering, Fairfax, VA, Sept 2010.
2. Sage, Andrew and Rouse, W. B., Handbook of Systems Engineering and Management, 2nd ed., Wiley 2009.
3. Jacobus, P., P. Yan, and J. Barrett. “Information management: the Advanced Processor Build (Tactical).” *Johns Hopkins APL Technical Digest* 23, no. 4 (Jan 2002): 366-372.
4. Boehm, B., and P. Bose. “A collaborative spiral software process model based on theory W.” *Proceedings of the Third International Conference on Software Process, Vol.* 3, 59-68 (1994).
5. Mitchell, S., "Model-Based System Development for Managing the Evolution of a Common Submarine

Combat System." AFCEA-GMU C4I Center Symposium on Critical Issues in C4I, May 18 - 19, 2010, http://c4i.gmu.edu/events/reviews/2010/GMU-AFCEA-Agenda-2010.php

6. Gibson, B., S. Mitchell, and D. Robinson, "Bridging the Gap: Modeling Federated Combat Systems," Third International Conference on Model Based Systems Engineering, Fairfax, VA, Sept 2010.
7. International Council on Systems Engineering (INCOSE), Systems Engineering Vision 2020, Technical Report INCOSE-TP-2004-004-02, Sept. 2007, http://www.incose.org/ProductsPubs/products/sevision2020.aspx
8. Callahan, Sean M. "Extended generic product structure: an information model for representing product families." *Journal of Computing and Information Science in Engineering* (ASME) 6 (Nov 2006): 263-275.
9. Jiao, J., and M. M. Tseng. "An information modeling framework for product families to support mass customization manufacturing." *CIRP Annals-Manufacturing Technology* (Elsevier) 48, no. 1 (Jul 1999): 93-98.
10. Roshandel, R., A. V. D. Hoek, M. Mikic-Rakic, and N. Medvidovic, "Mae---a system model and environment for managing architectural evolution." *ACM Transactions on Software Engineering and Methodology (TOSEM)* (ACM) 13, no. 2 (Apr 2004): 240-276.
11. Jacobson, I., M. Griss, and P. Jonsson, Software Reuse- Architecture, Process and Organization for Business Success, ACM Press, New York, NY.
12. Webber, Diana L., "The Variation Point Model for Software Product Lines", PhD Dissertaion, George Mason University, 2001.
13. Webber, Diana L., and Hassan Gomaa, "Modeling variability in software product lines with the variation point model", Science of Computer Programming, Vol. 53, No. 3, 2004.
14. Bachmann, Felix., Michael Goedicke, Julio Leite, Robert Nord, Klaus Pohl, Balasubramaniam Ramesh, and Alexander Vilbig, "A meta-model for representing variability in product family development", Software Product-Family Engineering, LNCS 3014, 2004, pp. 66–80.
15. de Oliveira, Edson Alves, Itana M. S. Gimenes, Elisa Hatsue Moriya Huzita, and Jose Carlos Maldonado, "A variability management process for software product lines", Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research (CASCON'05), 2005, pp. 225-241.
16. Friedenthal, Sanford et al. *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann, 2008.
17. Van Gurp, Jilles, Jan Bosch, Mikael Svahnberg, "On the notion of variability in software product lines", Proceedings of the Working IEEE/IFIP Conference on Software Architecture, 2001, pp. 45-54.
18. Svahnberg, Mikael, Jilles van Gurp, Jan Bosch, "A taxonomy of variability realization techniques", Software: Practice and Experience, Vol. 35, No. 8, pp. 705–754.
19. Object Management Group, "Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS)" , Draft (Beta 2), Dec. 2009.
20. Friedenthal, Sandy, private communication, Oct. 21, 2010.