Product Line Engineering Meets Model Based Engineering in the Defense and Automotive Industries

Bobbi Young
Raytheon Integrated Defense Systems
Bobbi.Young@raytheon.com

Judd Cheatwood, Todd Peterson
General Dynamics Mission Systems
{Judd.Cheatwood, Todd.Peterson}@gd-ms.com

Rick Flores
General Motors
rick.r.flores@gm.com

Paul Clements
BigLever Software
pclements@biglever.com

ABSTRACT

Product line engineering and model based engineering are two powerful engineering approaches that each bring significant advantages to system engineering projects. This paper explores how three companies – Raytheon, General Dynamics, and General Motors – are combining these two paradigms in unique and innovative ways in very challenging application domains to achieve engineering goals of critical importance to them.

CCS CONCEPTS

• Software and its engineering → Software product lines;

KEYWORDS

Product line engineering, model-based engineering, feature models, feature profiles, variation points, product configurator, feature-based product line engineering, PLE factory,

ACM Reference Format: Bobbi Young, Rick Flores, Judd Cheatwood, Todd Peterson, Paul Clements, Product Line Engineering Meets Model Based Engineering in the Defense and Automotive Industries. *In Proceedings of SPLC '17*, Sevilla, Spain, September 25-29, 2017, 10 pages.

DOI: 10.1145/3106195.3106220

1. INTRODUCTION

Model based engineering is "an approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and/or product throughout the acquisition life cycle" [6]. A model, in turn, is "a physical, mathematical, or otherwise logical representation of a system, unity, phenomenon, or process" [2]. Model based engineering is held in contrast to approaches in which informal prose or diagrams serve as the basis for the information exchange among stakeholders in a systems engineering process. Models, because of their physical or mathematical formulation, tend to be less ambiguous and more amenable to high-confidence analysis, thus reducing errors and re-work in the systems engineering process.

The purpose of this paper is to discuss the combining model-based engineering (MBE) with product line engineering (PLE), and to describe ways in which the combination is already in industrial. The examples originate with three Fortune 150 companies: Raytheon, General Dynamics, and General Motors. While their technical approaches are similar, their goals and results are (we hope) interestingly different and together illustrate a broad picture of some of the ways that PLE+MBE is being used in industry today.

2. FEATURE-BASED PRODUCT LINE ENGINEERING

All three companies profiled in this paper employ a very specific approach to PLE. The approach has often been referred to as "second generation PLE" [5] but we will refer to it here as "Feature-Based PLE," to align with a forthcoming ISO standard that will describe the approach by that name.

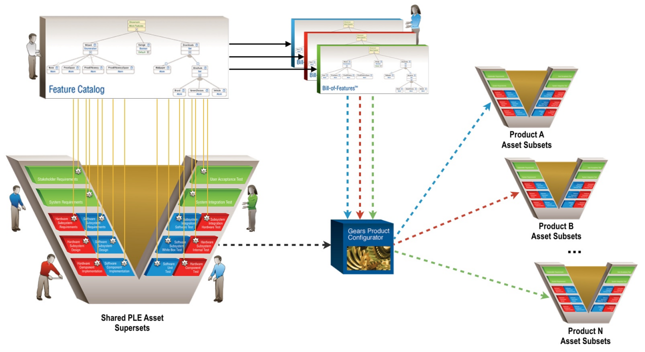

Feature Catalog
Bill-of-Features™
Stakeholder Requirements
System Requirements
Hardware Subsystem Requirements
Software Subsystem Requirements
Hardware Subsystem Design
Software Subsystem Design
Hardware Component Implementation
Software Component Implementation
User Acceptance Test
System Integration Test
Subsystem Integration Software Test
Subsystem Integration Hardware Test
Software Subsystem White Box Test
Hardware Subsystem Internal Test
Software Unit Test
Hardware Component Test
Shared PLE Asset Supersets
Gears Product Configurator
Product A Asset Subsets
Product B Asset Subsets
...
Product N Asset Subsets

Figure 1 PLE as a factory (figure used with permission of BigLever Software)

In Feature-based PLE, the engineering assets are shared across the product line. These shared assets can be whatever artifacts are representable digitally. These shared assets are created and maintained as supersets, meaning that they contain any content needed to support any of the products. A configurator (in this paper we use Gears [1]) produces product-specific instances by actuating a product — that is, exercising variation points in the supersets according to the feature choices for that product. A feature is a distinguishing characteristic that sets products in a

ACM ISBN 978-1-4503-5221-5/17/09…$15.00

http://dx.doi.org/10.1145/3106195.3106220

product line apart from each other. A variation point is a specification attached to a piece of content in a shared asset that stipulates the feature choices under which that content is needed in the product-specific instance of that shared asset. The collection of feature choices for a product is called a Bill-of-Features, and is drawn from all of the available feature choices for the product line, which are captured in a feature catalog. Figure 1illustrates.

3. COMMON APPROACH FOR MBE AND PLE TOGETHER: MODELS BECOME A PLE SHARED ASSET

The three companies represented use MBE to accomplish specific goals for the design and engineering of their respective systems – plural. And therein lies the motivation for combining MBE with PLE. In that light, it becomes apparent that the MBE models need to efficiently and systematically manage variation. One model won't do. To support their respective product lines, each company's models need to be able to support product-specific model instances, just like their requirements, code, tests, and other shared assets do.

Therefore, each company described turns to MBE to handle the modeling chores important to them, and turns to PLE to handle the variation inherent in the models that reflects the diversity in the products they are building. All three companies use Feature-based PLE to configure their models by attaching them as shared assets (along with other shared assets) to their respective PLE factories, and configure them using the same feature selections that drive the configuration of their other shared assets.

4. RAYTHEON: INTEGRATED AIR/MISSILE DEFENSE

The Raytheon Company is a technology and innovation company specializing in defense, civil government, and cybersecurity throughout the world. They employ 63,000 employees worldwide. MBE and PLE are both in use, to varying degrees, throughout Raytheon. We will relate the experience of one product area in particular, in the radar domain, where the two are being applied together.

To protect confidentiality we will use a fictitious but representative example called GloboShield. GloboShield's mission is to protect a theater from air and missile attack by detecting, tracking, identifying, and destroying airborne threats. A system includes sensors, displays, planning functions, threat evaluation, health and status monitoring, communication with friendly systems for information exchange, and more. Customers can order GloboShield in different configurations, and with different levels of capability. For example, GloboShield provides options for its Threat Assessment capability, which we will use to illustrate how model variation is handled. The customer may

- Choose or omit the Threat Determination service to identify a threat that could be an air-breathing target (ABT) and/or a theater ballistic missile (TBM).
- Choose or omit a Threat Ranking and/or a Threat Warning service.

Figure 2 is a sketch of a system architecture for GloboShield, identifying its major subsystems along with some explication about each. (Figure 2 does not tell the whole architectural story, of course, but for the purposes of our discussion, we will let Figure 2 stand in for the entire range of useful architecture documentation.) Each instance of GloboShield will have its own system architecture that reflects the product choices outlined above as well as many others, but in all cases one that is derived from the architecture illustrated in Figure 2. (For the purposes of this paper, it is not important to understand the details of the "master" or derived architectures.)

Product architectures are subsets of the master architecture. For example, a product architecture may omit some of the components that populate the master architecture (and therefore the relationships or "connectors" that tie those components to other components). This typically occurs when those components provide a capability that has not been chosen for a product. A product architecture may vary from the master in other ways; for example, a component may exist in both the master and a derived product architecture, but that component will be of a different "flavor" in the product, or bind certain choices about it that are available in its master-architecture analog.

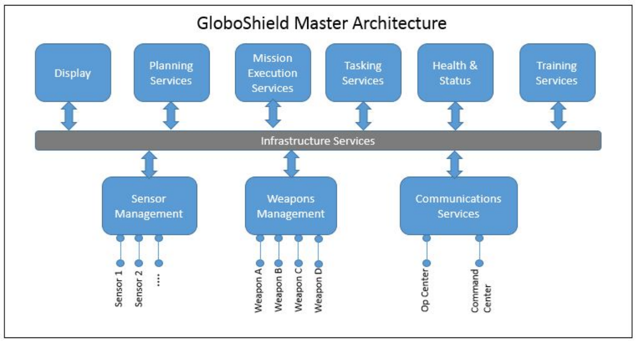


Figure 2 GloboShield master architecture view identifying major architectural components

To move from the documentation-centric realm into the model-based realm, these architectural designs need to be captured in a more formal representation. Raytheon, like many systems engineering organizations, uses the System Modeling Language (SysML) as its preferred language in which to represent system architectures. The master and derived architectures can be represented in SysML. Raytheon uses Rhapsody from IBM Rational as the modeling tool with which to capture SysML models. In alignment with MBE, these choices enable analysis and derivation (for example, code generation) to be brought to bear, whereas any utilization of the architectural information before was purely manual and fraught with error-prone and labor-intensive work.

To treat our architecture models in accordance with the PLE Factory model of Figure 1, the systems engineering team developed a superset view of the architecture to capture those systems that would be variant. Figure 3 captures the variations identified for the structure of the enterprise view. The gear icon on some of the blocks (along with the Variation Point stereotype in the model for these elements) denotes model elements that are variation points – that is, they do not appear in every member of the product line.

In order to identify the variant features, a feature model was developed. Raytheon has chosen the Gears PLE tool and framework [1] to serve as its feature modeling tool and the PLE Configurator shown in Figure 1.

The feature model captures distinguishing characteristics that set products apart – that is, only those features that represent variation are modeled. The feature model is a hierarchical decision tree that identifies and defines opportunities for variation; making selections in such a decision tree defines a particular product.

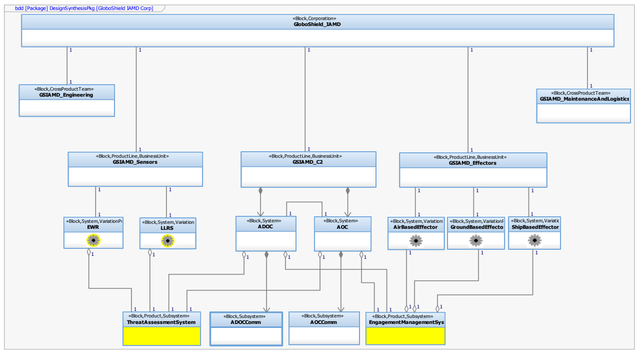


Figure 3 GloboShield block definition diagram superset with variation

Figure 4 shows part of the GloboShield feature model. It describes the variant features a customer may choose from for a specific product. Features defined in the feature model can be capabilities (i.e., functional, operational, presentation, implementation techniques, operating systems, and operating platforms) that represent variation among products. Some options may be dependent on other options and need to be included or cannot be combined with other options. These dependencies and constraints are defined as assertions and are captured as rules.

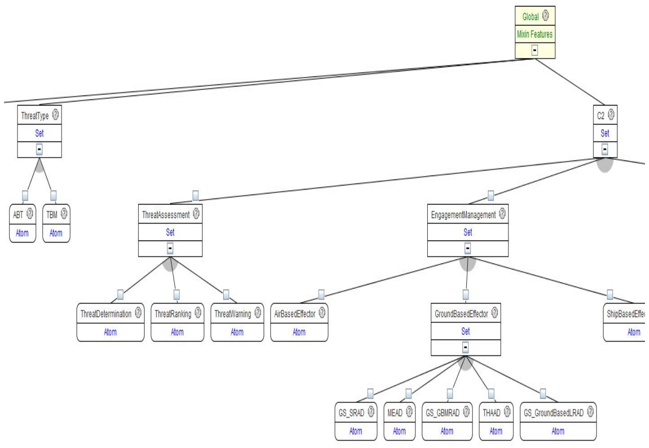
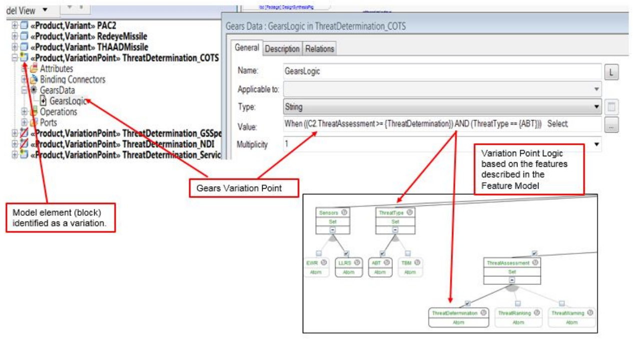


Figure 4 Feature model (excerpt) for threat assessment

Feature profiles are defined from the feature model by selecting the feature options that define a specific product; for instance, a feature profile that chooses the Threat Determination option for a GloboShield product instance, or another that selects Threat Determination, Threat Warning, and Threat Ranking for a different GloboShield product instance. Feature choices must be mapped to the variation within shared assets, so that the configurator can reflect the feature choice in the configuration of the shared asset. This is accomplished by adding variation points to the assets. Figure 5 shows a variation point that has been added to the ThreatDetermination block in Rhapsody. The variation point logic is written in a logic language that maps to the feature declarations. The logic tells Gears what feature choices or combination of feature choices will cause that block to be included in the projection, or product-specific instantiation of this shared asset. In the Rhapsody model, when Gears produced the Threat Determination ABT product configuration (Figure 6), the model elements with the gear icon are either highlighted in yellow if included, or a red slash if excluded, from the projection. All other model elements not tagged with variation points are common and will always be included in the projection.


Gears Variation Point
Model element (block) identified as a variation.
Variation Point Logic based on the features described in the Feature Model

«Product,Variant» PAC2
«Product,Variant» RedeyeMissile
«Product,Variant» THAADMissile
«Product,VariationPoint» ThreatDetermination_COTS

Gears Data : GearsLogic in ThreatDetermination_COTS

Name: GearsLogic
Type: String
Value: When ((C2.ThreatAssessment >= {ThreatDetermination}) AND (ThreatType = {ABT})) Select
Multiplicity: 1

Figure 5 Rhapsody view of Gears variation points

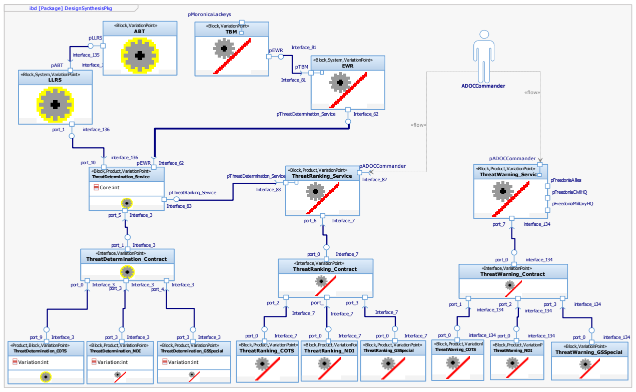


Figure 6 Threat assessment block diagram after actuation for the threat determination ABT product

A single actuation step will produce not only a block diagram but also any other diagrams that have been similarly imbued with variation points. The result is a holistic set of models, consistently configured to represent the same product instance.

When the PLE factory produces a product-specific architectural model in Rhapsody, we can then carry out all of our MBE-based analysis and downstream transformations. The derivation of the product architecture(s) from the master architecture of Figure 2 is now fully automated, happens in a few seconds, and is not prone to the errors of manual derivation. Moreover, we have only the superset to store, maintain, and update. The derived architectures are not maintained on their own, but merely re-generated when the superset changes. Thus, we have cut the artifacts we need to manage and store by 80% (from 5 to 1).

5. GENERAL DYNAMICS: MBE+PLE+PLM

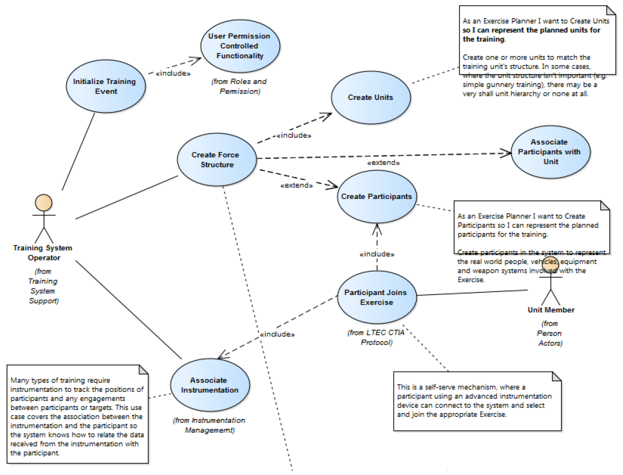
General Dynamics employs 99,500 people world wide, is listed as #88 on the 2016 Fortune 500, and is the world's fifth largest defense contractor. The product line from which this MBE+PLE story is taken is Live Training Transformation (LT2), a family of training systems developed primarily for the US Army but also used by other US military services.

Each member of the LT2 product line carries out a training scenario for one or more warfighters[1]. Generally speaking, the

[1] "Warfighter" is a generic term for soldier, sailor, airman, or Marine.

system records activities, scores actions, help produce after-action reports for review, and helps trainers carry out a training scenario. Systems at the small end of the spectrum could involve a single warfighter qualifying with a weapon. Systems at the large end involve whole brigades on maneuvers, engaging each other with lasers and wearing laser vests.

In LT2's PLE factory of Figure 1, the primary shared assets include software source code, a bill-of-materials (BOM) for each system being deployed to training facilities around the world, and extensive maintenance and training documentation. LT2 has to date accumulated more than $746 million in cost avoidance based on its product line engineering approach [4]. LT2 employs Enterprise Architect as its tool of choice to capture the design models for the systems in the product line and also uses SysML as its modeling language. LT2 engineers use their models to capture and convey the designs of the various systems in the product line.


User Permission Controlled Functionality
(from Roles and Permission)
Initialize Training Event
«include»
Create Units
Create Force Structure
«include»
Associate Participants with Unit
«extend»
«extend»
Create Participants
Training System Operator
(from Training System Support)
«include»
Participant Joins Exercise
(from LTEC CTIA Protocol)
Unit Member
(from Person Actors)
«include»
Associate Instrumentation
(from Instrumentation Managememt)

As an Exercise Planner I want to Create Units **so I can represent the planned units for the training**.

Create one or more units to match the training unit's structure. In some cases, where the unit structure isn't important (e.g. simple gunnery training), there may be a very shall unit hierarchy or none at all.

As an Exercise Planner I want to Create Participants so I can represent the planned participants for the training.

Create participants in the system to represent the real world people, vehicles, equipment and weapon systems involved with the Exercise.

Many types of training require instrumentation to track the positions of participants and any engagements between participants or targets. This use case covers the association between the instrumentation and the participant so the system knows how to relate the data received from the instrumentation with the participant.

This is a self-serve mechanism, where a participant using an advanced instrumentation device can connect to the system and select and join the appropriate Exercise.

Figure 7 LT2 use case, rendered in SysML with Enterprise Architect (figure used with permission of General Dynamics)

LT2 engineers focus on use cases, captured as Use Case Diagrams (Figure 7). The plan is to add variation points to these representations so that use cases specific to a member of the LT2 product line can be automatically generated by the LT2 PLE factory. The plan is to also add variation points to the system logical elements and physical elements such as constituent assemblies and parts. When General Dynamics deploys a training system to a site, all of the parts and pieces (computers, cameras, equipment racks, laser vests, cables, labels, radios, and much more) arrive in shipping containers. Those shipping containers have to be packed, and include exactly the components needed by that training system – no more and no less. So, a critical part of LT2's PLE factory is a bill of materials (BoM), and a critical part of LT2 engineering is the Product Lifecycle Management (PLM) that manages the BoM as part of the "digital twin" of the system as it evolves from design through deployment, and on-site maintenance.

Like all shared assets, the BoM is maintained as a superset. Based on feature choices that describe the system being deployed, the configurator produces a BoM subset specific to that system, which can be used to assemble the collection of physical parts that make up a system.

LT2's over-arching vision is a marriage of design with matching physical implementation; their vision is a vision of MBE integrated with Product Lifecycle Management (PLM), which is the realm of parts and manufacturing. Because both realms exist in a product line environment, Feature-based PLE is being used to align those two worlds. This vision is taking MBE into the PLM world of manufacturing and physical system deployment and maintenance, with PLE making it all work seamlessly in the context of multiple products.

6. GENERAL MOTORS: SEAMLESS ENGINEERING

General Motors is one of the world's largest automotive manufacturers, producing vehicles in 37 countries. It employs 209,000 people and is currently ranked #8 on the Fortune 500 list. General Motors' product line of vehicles has been referred to as "mega-scale product line engineering," [3] meaning an extremely large product set (over 9 million per year in GM's case) with extremely complex individual products, and extremely complex feature variation and interaction in those products.

GM is employing PLE and MBE together to achieve at least three specific goals, which we will address in turn.

Models to facilitate seamless end-to-end systems engineering: GM's primary goal is to achieve a truly integrated systems engineering activity to enable a seamless and unified systems engineering approach to automotive manufacturing that combines the electrical/electronic, software, and hardware realms.

To run the software (up to 10 million lines in some cases), a car may have dozens of electronic control units (ECUs), distributed around the vehicle on multiple networks, which leads to complexity in terms of optimal deployment, or assignment of functionality (and the software that provides it) to individual processors.

Thus, the feature variation alluded to above needs to be realized in software, in electronic control units, in serial data messages flowing across networks that connect the electronic modules, and more. GM system engineers see this as a continuum between the functional and the physical realms, requiring flawless translation on every vehicle. But vehicle designers do not choose ECUs, data messages, or software code; rather, they choose features that are much more customer-facing such as active safety or interior lighting packages. The goal, then, is to flow feature choices correctly down to decisions about what units of functionality to deploy on what ECMs, and have the ECMs operate and communicate correctly.

Figure 8 shows the levels involved in the feature realm. Primitive features are grouped into cohesive and coherent feature models, which in turn are conjoined into models of entire systems and subsystems, which produce choices from which a vehicle designer can select. Each layer bundles features at its layer into a much smaller number of available choices, reducing the complexity from literally astronomical at the bottom to a space on the order of hundreds of choices at the top.

This structure is mirrored in the design space. A vehicle's modeled functionality is divided into domains (for example, lighting), which offer features (for example, "guide me to my car" lighting) that exhibit behaviors. System components (for example, built-in daytime running lights) exist that are made to behave according to functions, which are units of feature behavior. System components are allocated to ECUs, and need to communicate with system components allocated to other ECUs.

All of this information is modeled using (in GM's case) Rhapsody; variation is modeled throughout the levels using features and feature profiles using Gears. All of the levels of the model mentioned above exist as supersets with variation points that Gears uses to configure based on feature choices.

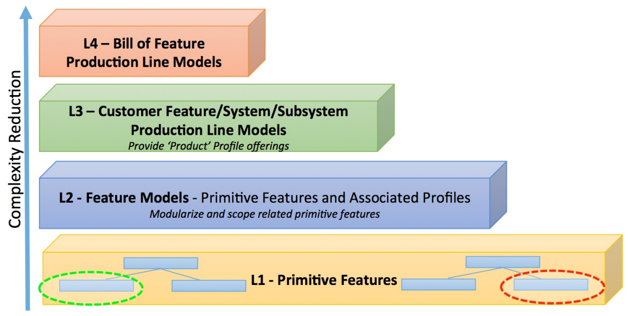

L4 – Bill of Feature Production Line Models
L3 – Customer Feature/System/Subsystem Production Line Models
Provide 'Product' Profile offerings
L2 - Feature Models - Primitive Features and Associated Profiles
Modularize and scope related primitive features
L1 - Primitive Features
Complexity Reduction

Figure 8 System engineering modeling levels (figured used with permission of General Motors)

The variation points act, as a group, to produce a consistent flow-down from features to ECU allocation, based on the selections for a particular vehicle.

Although the following is a simplification, it describes essentially the process of combining PLE with MBE: If a feature is not selected for a vehicle, then the system components to implement that feature are de-selected. If a system component is de-selected, then the functions to realize it are de-selected. If a function is de-selected, then it is not allocated to any ECU, and the serial data items for that function are not part of the ECU's repertoire.

Models to facilitate automatic calculation of calibration parameters: For reasons having to do with high-volume manufacturing, it is not possible for each vehicle to have its own customized software to match its chosen feature configuration. Instead, generic software is pre-loaded onto the various electronic control units, software that is capable of handling any feature configuration on the vehicle. The software's behavior is determined by a set of *calibration parameters*, whose values are loaded into a physical memory block during manufacturing. Feature-specific code is written to be predicated on the value of the appropriate calibration parameter: *if* (calibration for this feature is true) *then* (execute the code for this feature).

It takes many thousands of calibration parameters to configure the software for a vehicle. An error in a calibration parameter will mean that a feature for a vehicle is turned on, or off, inappropriately, and these errors can lead to costly maintenance actions. Calibration parameters are also part of the Rhapsody design models. So, to the list of ramifications of omitting a feature from a vehicle, we can add "If a feature is de-selected on a vehicle, set the corresponding calibration parameter in that vehicles Cal file to false."

GM estimates that the value of automatic generation of calibration parameters, and the reduction in errors from manual work is measured in hundreds to thousands of man/years per year, worth tens to hundreds of millions of dollars per year [7].

Models to facilitate performance analysis and network management: Since the serial data items needed to implement a piece of functionality are derived, it is now possible to derive a vehicle-specific data dictionary. Data items that are not needed are omitted. This enables much more precise architecting of networks and network topologies. Instead of putting a network on a vehicle that can handle the maximum possible communication load of any vehicle, it is now possible to put a network on a vehicle that can handle the maximum communication load for *that* vehicle. This can lead to more economical vehicle architectures.

GM Summary: Overall GM sees this approach as yielding better consistency, fewer defects, higher quality of data, a reduction of repetitive work, better connectivity among the various engineers who own different parts of the models, and a unified language for all of systems engineering within General Motors. The PLE aspect is essential. With a product line of a few, or even a few dozen, members it might be possible (if undesirable) to handle the models as separate instances. Not so at GM, where the product line numbers in the millions.

7. CONCLUSIONS

We have shown how a simple and effective way to combine two powerful engineering paradigms, MBE and PLE, is being used in practice in two extremely challenging system engineering industries. The approach is fully supported by off-the-shelf tooling and automation, all of which is in widespread use today. The combined paradigm uses the PLE Factory concept of a shared asset superset with variation points, automatically configured to produce product-specific instances.

Each company came to MBE+PLE through their own respective contexts, and with their own specific and quite different goals for the approach. PLE and MBE have, on their own, each reached industrial levels of maturity, backed up by robust technologies and methodologies that work at large scales. We hope to have shown that MBE and PLE together has now arrived on the scene fully formed and benefitting from the maturity of each of its parents, and providing the benefits of both.

8. REFERENCES

[1] BigLever Software, "BigLever Software Gears," http://www.biglever.com/solution/product.html

[2] Department of Defense, Directive 5000.59, August 8, 2007, http://www.dtic.mil/whs/directives/corres/pdf/500059p.pdf

[3] Flores, R., Krueger, C., Clements, P. "Mega-Scale Product Line Engineering at General Motors," *Proceedings of the 2012 Software Product Line Conference*, Salvador Brazil, August 2012.

[4] General Dynamics, "Training and Simulation," https://gdmissionsystems.com/c4isr/training-simulation/

[5] Krueger, C. and Clements, P. "Systems and Software Product Line Engineering," *Encyclopedia of Software Engineering*, Philip A. LaPlante ed., Taylor and Francis, 2013

[6] National Defense Industrial Association, "Final Report, Model-Based Engineering Subcommittee," Feb. 2011, http://sebokwiki.org/wiki/Final_Report_of_the_Model_Based_Engineering_(MBE)_Subcommittee

[7] Wozniak, L., Clements, P. "How Automotive Engineering Is Taking Product Line Engineering to the Extreme," *Proc. SPLC 2015*, Nashville, 2015.