Mechanical Product Lifecycle Management meets Product Line Engineering

Charles W. Krueger
BigLever Software
10500 Laurel Hill Cove
Austin, TX 78730 USA
+1-512-777-9552
ckrueger@biglever.com

ABSTRACT

Early generation Software Product Line (SPL) engineering has evolved into Systems and Software Product Line Engineering (PLE) approaches that extend well beyond the original focus on source code, to a more holistic perspective of the engineering lifecycle. PLE tools and methods in commercial practice today support variation management in requirements, architecture, design models, source code, documentation, configuration data, test cases and more. One of the last lifecycle holdouts from PLE has been mechanical engineering, or Product Lifecycle Management (PLM). The engineering complexity of mechanical product families with embedded software has increased to a threshold where it is intractable for mechanical and software product line engineering to remain disjoint. This paper explores the convergence of mechanical, systems and software product line engineering and why it has been slow to emerge. The reasons are based both on conceptual misalignment among the traditionally distinct disciplines, as well the differences between the physics of mechanical and software systems. The Aras Innovator / BigLever Gears Bridge, an example PLM and PLE integration, is used to illustrate key concepts.

Categories and Subject Descriptors

D.2.2 [**Design tools and techniques**]: product line engineering, mechanical product lines, product lifecycle management.

General Terms

Design, Economics, Management, Measurement, Theory.

Keywords

Mechanical Product Line Engineering, Product Lifecycle Management, Bill of Features, Bill of Materials.

SPLC 2015, July 20 - 24, 2015, Nashville, TN, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3613-0/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2791060.2791109

1. Introduction

Early generation Software Product Line (SPL) engineering has evolved into second generation Product Line Engineering (PLE) tools and methods that extend well beyond the original focus on source code only, into a more holistic perspective of the end-to-end engineering lifecycle[1]. There are PLE tools and methods in commercial production use today that support consolidated variation management across requirements, architecture, design models, source code, documentation, configuration data, test cases and more[2][3].

One of the last lifecycle holdouts in PLE has been mechanical engineering, which is commercially known as Mechanical Design Automation (MDA). MDA is often part of full lifecycle support for Product Lifecycle Management (PLM). When we refer to PLM in this paper, we are referring specifically to the MDA capabilities within PLM.

The increasing engineering complexity for product families of mechanical systems with embedded software is finally reaching a threshold where it is intractable for mechanical and software product line engineering to remain disjoint. This paper explores the convergence of mechanical and software product line engineering and why it has been difficult and slow to emerge. The reasons are based both on conceptual misalignment among the traditionally distinct disciplines, as well the differences between the physics of mechanical and software systems.

We illustrate the concepts of an integrated PLM and PLE approach with the integration of a commercial PLM tool, Aras Innovator, into the BigLever Gears PLE lifecycle framework. For mechanical engineers, this integration results in a paradigm shift in the way of thinking about engineering a family of hybrid mechanical, electrical and software systems and products. Rather than use a *Bill-of-Materials* (BoM) to determine the *Features* that emerge in a system, start with a *Bill-of-Features* to help determine the *Materials* needed in a system, where materials in this case includes assets from across the mechanical, systems and software engineering lifecycle — requirements, architecture, design models, software, mechanical BoM, electrical, tests, documentation, and more.

2. From Software Product Lines to Mechanical Product Line Engineering

To help set the context for this paper, the evolution of the Product Line Engineering field can be viewed according to five evolving and converging engineering disciplines.

- Product Lifecycle Management (PLM) for mechanical-centric engineering.
- Software Product Line Engineering (SPL).
- Second generation Systems and Software Product Line Engineering (2GPLE) for software-centric Application Lifecycle Management (ALM).
- Mechanical, Systems and Software Product Line Engineering (PLE). The convergence of ALM, PLM and 2GPLE.
- Product Line Engineering and Operations (PLE&O), which extends PLE beyond the engineering lifecycle into the business operations before and after engineering.

The genesis of the product line field was in Software Product Line Engineering, which itself emerged from studies in reusing product-scale software architectures and designs[4][5][6]. The primary focus here was on software and the source code implementation phase of the engineering lifecycle.

SPL evolved into 2GPLE, where the assets from across the broader software-centric engineering lifecycle are also managed for product families[1]. The software-centric systems engineering lifecycle is sometimes referred to as the Application Lifecycle Management (ALM). 2GPLE approaches are feature-based, where a PLE *feature model* serves as the single source of feature truth for all assets across the lifecycle[3].

The focus of this paper is on the convergence of 2GPLE and PLM. The software-centric ALM perspective of 2GPLE is extended to incorporate PLM. This provides the holistic discipline of mechanical, systems and software PLE.

The final discipline in the evolution embraces the pre- and post-engineering lifecycle phases – the extended lifecycle that include business operations such as supply chain, manufacturing, sales, delivery, and product maintenance. This is addressed in the Future Work section.

3. Impediments to Mechanical Product Line Engineering

Product Lifecycle Management for mechanical products is one of the last engineering disciplines to be incorporated into the current generation feature-based systems and software PLE approach. As part of our work to address this gap, we identified several impediments that explain why PLM has been a challenging integration with PLE.

On one hand, it is certainly the case that feature-based variant management is compatible with mechanical engineering. An optional or varying feature on a mechanical system with embedded software may crosscut the entire engineering lifecycle, resulting in optional or alternative mechanical parts that apply to some members of the product family and not others. At first glance it would appear that adding PLM to PLE might be straightforward and very similar to the way that other lifecycle disciplines from ALM, such a requirements, system design models, and tests were natural extensions to in feature-based PLE paradigm.

However, there are characteristics of PLM and mechanical engineering that are substantially different from the other systems and software centric disciplines, resulting in unique challenges and the need for unique solutions. These differences come from conceptual misalignment between the traditionally distinct engineering disciplines of software-centric ALM and mechanical-centric PLM, plus differences in the real world physics of mechanical and software artifacts. These differences, described in the following subsections, have been impediments to integrating PLM into PLE and have thereby slowed the efforts to achieve holistic end-to-end lifecycle PLE for mechanical, systems and software engineering.

3.1. Differences in Product Realization and Deployment for ALM and PLM

PLE for ALM assets that are realized as bits and bytes (soft assets) is different from PLE for PLM assets that are realized as physical, mechanical parts (mechanical assets).

- The cost, time and effort required to modify, reproduce and retest soft assets is significantly less than for mechanical assets. With PLE it is possible to automatically regenerate and retest 100 different ALM product variants every night, whereas setting up and exercising a mechanical testbed for a single product variant can take days and significant expense.
- The cost, time and effort to produce, distribute and install many exact copies of software assets is negligible compared to manufacturing and shipping many exact copies of mechanical assets.
- When soft assets are updated and redeployed, it's often easiest and least expensive to replace everything rather than attempting minimal updates to only what has changed. Examples include replacing an entire electronic document, compiled application software, requirements modules in a database, a collection of design models, and zip files containing a collection of test cases. However, updates to mechanical assets requires care and consideration to minimize the total cost and risk of replacing only the necessary parts or assemblies. It makes complete sense to replace an entire software application when there is a user interface bug, but it never makes sense to replace an entire automobile when there is a defective switch on the dashboard.
- There are limited physical constraints for soft assets – for example memory space to run an executable. However, one of the core capabilities of MDA in PLM is managing a large number of different types of physical constraints for mechanical assets. Physical parts can't overlap in space. The space that must be maintained between heat generating parts and heat sensitive parts is influenced by airflow characteristics provided by the larger assembly. Electromagnetic interference (EMI) constrains the placement and selection of part and shielding variants.

The implications of all of this for PLE is that the feature-based, automated 2GPLE configuration tools and methods that are optimized for ALM soft assets need to be augmented to account for the physics and economics of variant part selection in mechanical assets. For ALM, feature-based variant and selection criteria can be generally grouped into *functional* and *non-functional* characteristics. For example:

- Passive entry for door locking and unlocking on an automobile is a *functional* variant. It is customer-facing with externally visible behavioral differences if the passive entry option is present or not present on a vehicle.
- A low-noise cooling fan for an automobile engine heat exchanger is a *non-functional* variant. The customer is not directly aware of this option on a vehicle, but engineers might select a more expensive low-noise cooling fan on more expensive vehicles in order to achieve lower overall noise levels in the passenger compartment.

PLM requires an additional type of PLE selection criteria to account for decisions based on the physics and economics of manufacturing and delivering mechanical parts. We refer to the selection criteria in this group as *operational* characteristics. For example, the same or compatible part in a mechanical BoM assembly might be sourced from multiple suppliers to lower risk and increase competition. The final selection between the part variants from multiple suppliers might then be influenced by:

- negotiated supplier volume agreements
- shipping cost between a supplier and a particular manufacturing plant
- alignment of the time when parts are needed and the time when a supplier can deliver the parts
- load balancing the percentage of parts from multiple suppliers to accommodate the output capacity of each supplier

3.2. Differences in Version and Variant Management for ALM and PLM

Like all engineering lifecycle disciplines, the idea of creating a product family is not a new to PLM. Like all engineering lifecycle disciplines, PLM BoM tools offer mechanisms to manage part *variants* in the BoM in order to support product family variants. The same is true for managing temporal *versions* of PLM BoM parts and assemblies as they are modified over time.

However, the constructs and techniques used for managing versions and variants in PLM versus ALM are conceptually different, making it difficult for experts in one domain to effectively communicate about these concepts with experts in the other domain.

ALM techniques for variant and version management tend to be based on source code configuration management constructs and techniques such as file versions, concurrent development branches, and labeled baselines. Users typically create and work in a *workspace* that contains only one version and variant of each file or object, viewing one product at a point in time.

PLM techniques for variant and version management tend to be based on database constructs and techniques such as effectivity (i.e., specifying ranges of time and/or products for which a particular part instance is valid). Users typically work in an unfiltered database view displays all of the versions and variants side-by-side or in filtered views for a particular product or range of products.

Aligning ALM versions, branches and baselines, which tend to be statically enumerated content, with PLM effectivity, which tends to be dynamically filtered content, is not straightforward because there are no clear isomorphisms. Archiving copies of assets from each lifecycle discipline is a typical technique for managing the versions and variants that were delivered in a product instance.

Another implication on PLE of the differences between ALM and PLM version and variant management is that mechanical PLM product content continues to evolve after a product is deployed, whereas ALM product content remains fixed. Automobiles undergo service and maintenance over time, possibly resulting in individual parts being replaced by new and/or improved parts. Airplanes require certain parts to be serviced or replaced after a declared number of hours in service. A particular replacement part may be incompatible with a replacement part in a different assembly, so it is essential to keep records of the as-maintained state for safety-critical products, all the way down to the serial numbers of each assembly that might be impacted by a safety recall.

4. Solutions for Mechanical Product Line Engineering

We have addressed each of the PLM-PLE impediments and gaps identified above and created a commercially available solution, based on the BigLever Gears PLE Lifecycle Framework and the Aras Innovator PLM BoM tool[7][8].

This section describes the general approach used to address each of the impediments. In the next section we describe the specific implementation details in the Innovator/Gears Bridge solution.

4.1. PLE Configurators versus PLM Configurators for Variant Management

In Section 3.1, the primary gap identified for PLM with PLE was the need for *operational* selection criteria when doing PLE variant part selection in PLM.

Two approaches were identified and considered:

- Add *operational* “features” in a PLE feature model to drive part selection with the feature-based PLE configurator. For example, if there are two suppliers for a particular part, SupplierA and SupplierB, define a mutually exclusive feature enumeration that would allow the portfolio owner to select either SupplierA or SupplierB in each product Bill-of-Features.
- Use an existing MDA variant management and constraint solving configurator in PLM for *operational* part variant selection and use the feature-based PLE configurator to pre-filter with the functional and non-functional selection criteria.

Using two configurators in the latter approach initially seems unwieldy, but separation of concerns argues strongly in favor.

- First, applying the Bill-of-Features concern of PLE, the BoM for a product family is filtered by the PLE configurator to remove parts that don’t satisfy the functional and non-function selection criteria. The result is a partially filtered BoM with different "buckets" that contain variant part alternatives. The mutually exclusive parts within any bucket differ by operational selection criteria and the partially filtered BoM is still unsolved for physical constraints.

- Next, applying the Bill-of-Materials concern of PLM, the feature-filtered BoM is solved by the MDA configurator to fully resolve the part selections based on the desired set of operational characteristics and physical constraints.

This separation of concerns has several advantages. The PLE Bill-of-Features concern assures that a single source of feature truth will consistently apply variation across the full mechanical, system and software engineering lifecycle. The PLM Bill-of-Materials concern assures that non-feature-based operational concerns do not cross into the PLE domain and it leverages the sophisticated physical constraint solvers that are one of the hallmarks of advanced PLM MDA tools.

4.2. Temporal Baseline Management

One of the three dimensions of the 2GPLE approach is the temporal management dimension, where inter-asset baselines are managed across the different tools, assets and phases in the engineering lifecycle[1][3]. We have successfully applied temporal baseline management with collections of many different ALM tools. Fortunately, temporal effectivity in PLM databases meshes cleanly into the cross lifecycle baseline management of 2GPLE. Rather than using a static ALM baseline label, PLM joins a cross lifecycle temporal baseline using a dynamic effectivity expression that matches the point in time for the baseline.

5. The Aras Innovator / BigLever Gears Bridge: PLM meets PLE

The practical realization and validation of the work in this paper is the Aras Innovator/BigLever Gears Bridge, which is an integration of the Aras Innovator PLM BoM tool into the BigLever Gears PLE Lifecycle Framework. This makes Innovator a first class member of the BigLever PLE Ecosystem. It is currently being deployed on the US Army Consolidated Product Line Management (CPM) Live Training Transformation program, an SPL Hall of Fame inductee[9].

The Innovator/Gears Bridge enables Mechanical PLE, providing a paradigm shift for engineers creating a hybrid mechanical, electrical and software product family. Rather than use a Bill-of-Materials to determine the features that emerge in system, start with a Bill-of-Features to help determine the materials needed in a system, where materials in this case includes all of the soft assets and mechanical assets across the lifecycle.

The BigLever Gears PLE Lifecycle Framework embodies the 2GPLE paradigm, where all of the bridge integrations from across the full engineering lifecycle, including Innovator/Gears, use the same single source of truth about the feature model, bill-of-features, and production line architecture, as well as share the same feature-based product configurator, variation point logic editors, PLE analysis tools, and more. The Innovator/Gear Bridge is built on the SDK, APIs and style guide from the Gears PLE Lifecycle Framework. As a result it has the same capabilities and look-and-feel as all of the other bridge integrations in the Gears PLE ecosystem.

Figure 1 shows an example screenshot from the Aras Innovator BoM tool that is extended with the Innovator/Gears Bridge. The bridge makes the Innovator BoM tool *product line aware*:

- PLE variation points are supported on parts and assemblies in the BoM
- A Bill-of-Features can be applied to the BoM superset by the PLE configurator to instantiate product specific BoM subsets
- PLE variation point editors and analysis operations are available in the Innovator toolbar

Figure 1 shows the superset BoM structure for an automobile remote control key fob. The BoM hierarchy is shown in the left column. A BoM superset contains all of the common parts that are shared across all family instances and the variation point parts that are either optional parts or mutually exclusive variants of a part.

The two parts at the bottom, KF-1370 Case Screws and KF-1380 Brand Badge, are common parts, indicated by the standard Aras part icon on the left of each BoM row. KF-1200 Key is an example of an optional variation point, indicated by the gray Gears icon badge overlaid on the upper-left of the Aras part icon. KF-1300 Case Back is a variation point with two mutually exclusive variants, KF-1301 Case Back for Key and KF-1302 Case Back No Key, indicated by the variant 'V' icon badges overlaid on the upper-left of the Aras part icons.

When a Bill-of-Features is applied to the BoM superset view by the Gears PLE configurator, variation points are visually marked to indicate which optional parts were included and excluded, as well as which mutually exclusive part variants were selected. A colorized view indicates the product-specific BoM subset that corresponds to the Bill-of-Features that was applied by the Gears configurator. As illustrated in Figure 1, optional parts will either have a red slash to indicate exclusion (e.g., KF-1100 Battery Assembly) or a yellow highlight to indicate inclusion (e.g., KF-1200 Key). The selected member from a set of mutually exclusive variants is also highlighted in yellow (e.g., KF-1301 Case Back for Key).

Of course, all common parts are always included in a product-specific subset BoM.

The variation point logic that determines the inclusion and exclusion of a variation point part is shown in the Gears Logic column of the BoM. The logic is edited with the Gears Logic expression editor that is available from the Aras tool bar (the pencil icon). The logic editor provides context sensitive help in constructing the relational expressions, based on the available features in the feature model. The variation point logic expression for KF-1200 Key says that this part should be included in the key fob BoM when the vehicle has an external KeyCylinder feature selected as one of the available RequestSource choices for electric door locking and unlocking.

6. Related Work

The idea of engineering a product family for mechanical systems is a core part of MDA and PLM[10]. PLM refers to this as option and variant management. Much like the variant management techniques in ALM tools and disciplines, these tend to focus on low level Bill-of-Materials abstractions and the interrelationships among the parts. PLM with PLE provides a higher level of abstraction in the Bill-of-Features, where product content can be defined with a part independent Bill-of-Features and then a BoM can be fully or partially derived from the Bill-of-Features.

7. Future Work

An unanticipated outcome of the integration of PLM and PLE is how the pre- and post-engineering business operations phases of the PLM lifecycle — for example, portfolio planning, supply chain management, manufacturing, sales, and product service — become natural candidate extensions for feature-based PLE. Each of these business operations require and implement their own approach to product variant management techniques. Leveraging PLE feature models as the single source of feature truth for both engineering and business operations offers intriguing value and we are currently implementing proof-of-concepts at some of our PLE customers. We refer to this extension to PLE as Product Line Engineering and Operations (PLE&O). We see this as the next important step in the evolution of the product line field.

References

[1] Charles Krueger , Paul Clements . Systems and Software Product Line Engineering. In Encyclopedia of Software Engineering. Taylor and Francis: New York, Published online: 03 Sep 2013; 1-14.

[2] Beuche, D. Modeling and building software product lines with pure: Variants. In Proceedings of the 15th International Software Product Line Conference, Limerick, Ireland, Sept 08–12, 2008, ACM Press, 2008; SPLC ’08, 358.

[3] The Systems and Software Product Line Lifecycle Framework. BigLever Software Technical Report #200805071r4. 2014. http://www.biglever.com/extras/PLE_LifecycleFramework.pdf

[4] Clements, P., Northrop, L. Software Product Lines: Practices and Patterns, Addison-Wesley, 2002.

[5] Neighbors, J. The Draco approach to constructing software from reusable components. Workshop on Reusability in Programming, Sep 1983, 167-178.

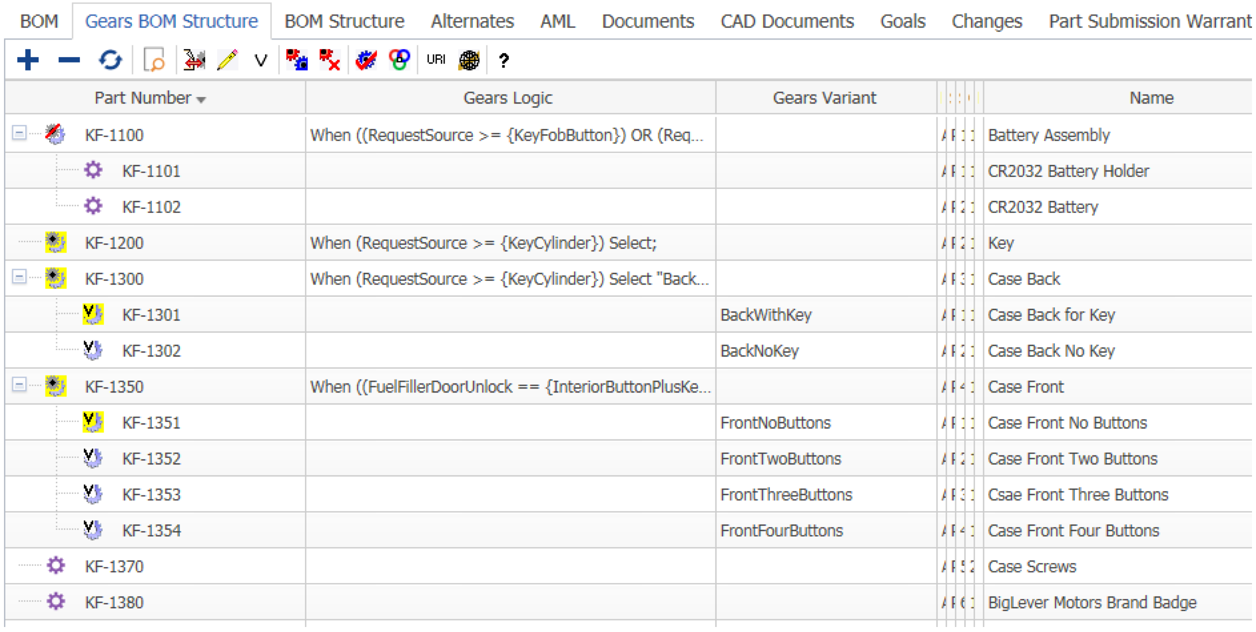
[6] Krueger, C. Modeling and Simulating a Software Architecture Design Space. PhD Thesis CMU-CS-97-158, School of Computer Science Computer Science, Carnegie Mellon University. December 1997.

[7] BigLever Software, “BigLever Software’s Product Line Engineering Solution,” http://www.biglever.com/solution/solution.html

[8] Aras Corp, “Bill of Materials,” http://www.aras.com/solutions/plm-solution.aspx?name=bill-of-materials

[9] Dillon, M., Rivera, J., Darbin, R., Clinger, B., “Maximizing U.S. Army Return on Investment Utilizing Software Product- Line Approach,” Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), 2012.

[10] Kreimeyer M. “A Product Model To Support Plm-Based Variant Planning And Management,” Proceedings of DESIGN 2012, the 12th International Design Conference, Dubrovnik, Croatia. May 2012. 1741-1752.



Part Number	Gears Logic	Gears Variant					Name
KF-1100	When ((RequestSource >= {KeyFobButton}) OR (Req...		A	F	1	1	Battery Assembly
KF-1101			A	F	1	1	CR2032 Battery Holder
KF-1102			A	F	2	1	CR2032 Battery
KF-1200	When (RequestSource >= {KeyCylinder}) Select;		A	F	2	1	Key
KF-1300	When (RequestSource >= {KeyCylinder}) Select "Back...		A	F	3	1	Case Back
KF-1301		BackWithKey	A	F	1	1	Case Back for Key
KF-1302		BackNoKey	A	F	2	1	Case Back No Key
KF-1350	When ((FuelFillerDoorUnlock == {InteriorButtonPlusKe...		A	F	4	1	Case Front
KF-1351		FrontNoButtons	A	F	1	1	Case Front No Buttons
KF-1352		FrontTwoButtons	A	F	2	1	Case Front Two Buttons
KF-1353		FrontThreeButtons	A	F	3	1	Csae Front Three Buttons
KF-1354		FrontFourButtons	A	F	4	1	Case Front Four Buttons
KF-1370			A	F	5	2	Case Screws
KF-1380			A	F	6	1	BigLever Motors Brand Badge

Figure 1. Innovator BoM with Innovator/Gears Bridge