30th Annual INCOSE international symposium
Cape Town, South Africa
July 18 - 23, 2020

Patterns for Success in the Adoption and Execution of Feature-based Product Line Engineering: A Report from Practitioners

Susan P. Gregg
Lockheed Martin
199 Borton Landing Road
Moorestown, New Jersey 08057 USA
+1 856 359 1636
susan.p.gregg@lmco.com

David Hartley
General Dynamics Mission Systems
12001 Research Parkway, Suite 500
Orlando, FL 32826 USA
+1 407 275 4820
David.Hartley@gd-ms.com

Morgan McAfee
General Dynamics Mission Systems
12001 Research Parkway, Suite 500
Orlando, FL 32826 USA
+1 407 275 4820
Morgan.McAfee@gd-ms.com

Randy Pitz
The Boeing Company
5775 Campus Parkway
St. Louis, MO 63042 USA
+1 314 563 5967
Randy.Pitz@boeing.com

James Teaff
Raytheon Intelligence, Information and Services
16800 East CentreTech Parkway
+1 303 344 6000
Aurora, CO 80011 USA
James.K.Teaff@Raytheon.com

Paul Clements
BigLever Software, Inc.
10500 Laurel Hill Cove
Austin TX 78730 USA
+1 512 777 9552
pclements@biglever.com

Abstract. Systems and Software Product Line Engineering (PLE) is a general approach to engineer a portfolio of related products in an efficient manner, taking advantage of the products' similarities while respecting and managing their differences. The approach manages a product portfolio as a single entity, as opposed to a multitude of separate products. Numerous resources describe the organizational benefits associated with incorporating PLE techniques and tools. Feature-based System and Software Product Line Engineering is a specific form of PLE that is powered by commercial off-the-shelf automation, fully defined processes, and a formal language of variation based on features. Many case studies show the efficacy of Feature-based PLE and the improvements in cost, schedule, and quality that can come with it. In this paper, practitioners from

four of world's six largest defense companies highlight their experience with the practices that enable and inhibit success with this powerful engineering discipline.

Introduction

Systems and Software Product Line Engineering (PLE) is a general approach to engineer a portfolio of related products in an efficient manner, taking advantage of the products' similarities while respecting and managing their differences (Krueger and Clements 2013). The key to the approach is managing the product portfolio as a single entity, as opposed to a multitude of separate products. Numerous case studies have shown the substantial advantages that this approach can bring in terms of improvements in product cost, time to delivery, and quality (see, for example, Clements and Northrop 2002). PLE has been institutionalized in the ISO/IEC 2655x family of standards (ISO/IEC 26550) and the Carnegie Mellon Software Engineering Institute's Framework for Product Line Practice (Northrop 2012).

Feature-based Systems and Software Product Line Engineering ("Feature-based PLE") is a specific form of PLE that became available in the early 2000's with the arrival of industrial-strength commercially-available tool support. The automation led to a formalization of methods and processes that underpin Feature-based PLE, including a language for defining the variation available in a product line with the concept of feature. Feature-based PLE has become widely practiced (see for example, (Clements 2015, Clements et al. 2013, Flores et al. 2017) and has demonstrated even greater improvement measures than its more general and less definitive antecedent.

In this paper, practitioners from four of the world's six largest defense contractors (CNBC 2019) – Lockheed Martin, Boeing, Raytheon, and General Dynamics – share their experience with practices that promote or inhibit success with this powerful engineering discipline. Some of the practices identify organizational issues, whereas others deal with technical practices involved in the execution of the discipline.

We begin with an overview of Feature-based PLE.

Feature-based PLE

The following description of Feature-based PLE is taken from the recently-published INCOSE primer on Feature-based PLE (INCOSE 2019).

Organizations utilizing Feature-based PLE adopt a *factory* approach to building their products. The factory is a conceptual construct that shows how the various aspects of Feature-based PLE interact with each other. Figure 1 illustrates:

- **Shared assets** are the "soft" artifacts that support the creation, design, implementation, deployment, and operation of products. A shared asset can be any artifact representable digitally: requirements, design models, source code, test cases, BoMs, wiring diagrams, documents, and user manuals, installation guides, and more. They either compose a product or support the engineering process to create a product. They can be represented digitally and configured, and are shared across the product line.

A shared asset used in the product line takes the form of a superset, meaning any asset content used in any product is included. There is no duplication or replication of asset content at all. This elimination of duplication is where Feature-based PLE derives its savings by eliminating work across duplicated assets.

The shared asset supersets contain variation points, which are places in the asset that denote content that is included, omitted, or configured according to the feature selections of the product being built. A statement of the product's distinguishing characteristics — its features — is applied to "exercise" these variation points (that is, cause the content associated with each variation point to be configured to meet the needs of the product).

A key aspect of Feature-based PLE is consistent and traceable treatment of variation across all shared asset types. Feature choices are the basis of a common language of variation across all disciplines and at all levels of the organization. This resolves the quagmire brought about by different disciplines each using its own approach to variation.

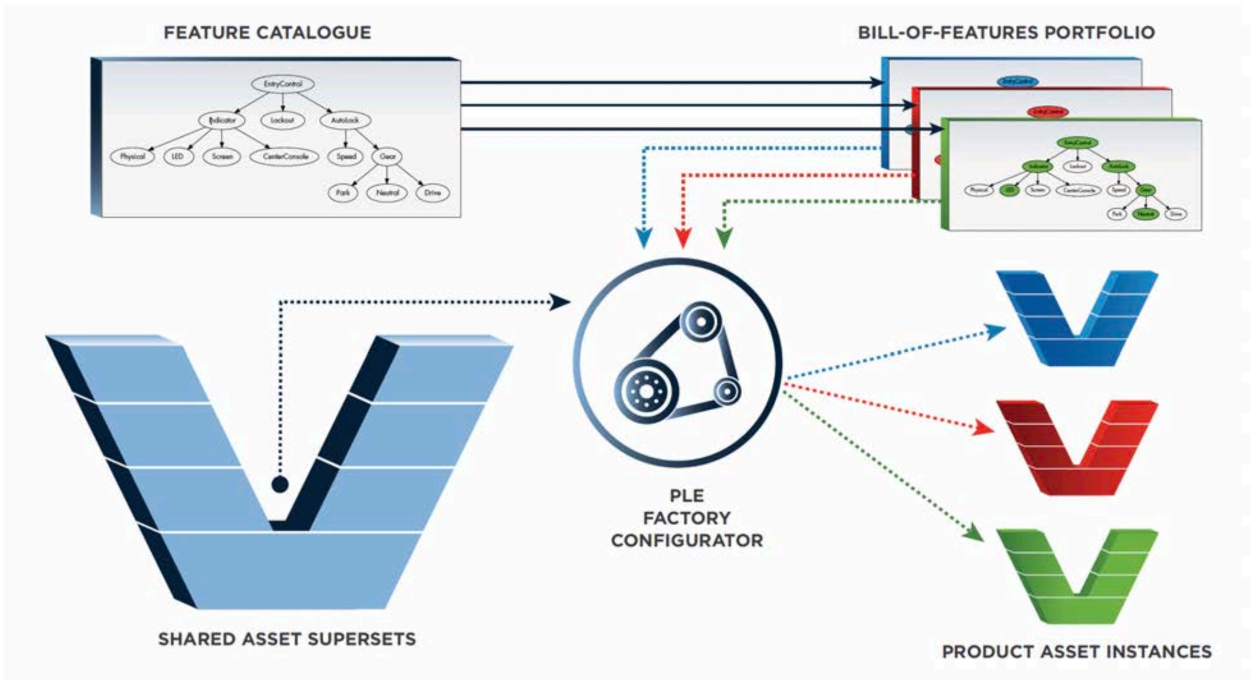

Figure 1. The Feature-based PLE Factory

- The **Feature Catalogue** captures the features that are available for each product to select. A feature is a distinguishing characteristic that describes how the members of the product line differ from each other. This provides a common language and definition of the product line's scope of variation for everyone throughout the organization.
- The features selected for each product in the product line are specified in the **Bill-of-Features** for that product.

- The **PLE Factory Configurator** is an automated software tool that applies a Bill-of-Features to all of the shared assets. It evaluates each variation point to determine if that variation point's content should be included or not.
- The PLE Factory produces as output **Product Asset Instances**, each one containing only the shared asset content suited for one of the products in the product line. Together, they constitute the artifact set for one of the products in the product line. Engineers now work on the shared asset supersets, and the Feature Catalogue, and the Bills-of-Features. Change and evolution are handled systematically through well-defined governance procedures.

Once the PLE Factory is established, engineering assets for products are instantiated rather than manually created. Feature-based PLE transforms the task of engineering a plethora of products into the much more efficient task of producing a single system: The PLE Factory itself.

Earlier approaches to PLE almost all emphasized a strong dichotomy between domain engineering and application engineering, sometimes called core asset development and product development (Krueger and Clements 2013). Application engineering involved choosing reusable domain-engineered core assets and grooming them for use in a product as needed. Application engineering includes the obligation to choose a production strategy—that is, a way to turn the shared assets into products. By contrast, the production strategy in Feature-based PLE is fixed – it is the generation of product asset instances by the configurator – which essentially eliminates application engineering entirely.

Case studies for the application of Feature-based PLE include (Wozniak 2015, Chalé Góngora 2017, Clements 2019, Flores 2017, Gregg 2015, Lanman 2011). The processes that underpin Feature-based PLE are well-defined and prescriptive, eliminating much of the guesswork that typified earlier forms of PLE; it is a prescriptive and focused specialization of the general form of PLE described by the ISO/IEC 26550x family of standards. For example, process definitions and collections of best practices for Feature-based PLE are now available to practitioners (SDTimes 2019). Accompanying the maturation of Feature-based PLE are experience reports that detail how Feature-based PLE works with Agile (Gregg 2014), works with Model-Based Development (Young et al. 2017), supports high-reliability testing (Gregg et al. 2017), works in mega-scale product lines (Flores et al. 2012), enables automation-supported intellectual property protection (Clements et al. 2013), works in high-security environments (Teaff et al. 2019) and more. Papers such as these, as well as this one, attest to the maturity and critical mass of the discipline.

Lockheed Martin: Powerful Governance

Background

The AEGIS Combat System is a system that protects assets from airborne attack from aircraft or missiles. It detects airborne threats, plans how to engage them, and launches missiles to intercept and neutralize them. AEGIS cruisers and destroyers constitute the majority of the U.S. surface Navy and will continue to form the core of the surface fleet for the next several decades. AEGIS is deployed on some 100 naval vessels in the U.S. Navy, navies of some U.S. allies across the globe, vessels of the U.S. Coast Guard, and even land-based ballistic missile defense installations.

At the heart of the AEGIS Combat System is the AEGIS Weapon System (AWS), which is a centralized, automated, command-and-control and weapons control system that was designed as a total weapon system, from detection to kill. The prime contractor for the AEGIS Weapon System is Lockheed Martin's Rotary and Mission Systems Division. There, some 1500 people work on the AEGIS program where, among other things, they maintain the over one hundred thousand AWS requirements and over ten million lines of source code used by AEGIS.

Lockheed Martin has been employing Feature-based PLE for AWS since the mid-2000's and estimates that because of the approach, over $47 million of cost is avoided each and every year (Gregg et al. 2015). The AWS product line, known internally as the Common Source Library (CSL), evolved from a series of standalone programs into a true systems and software product line that today ranks as one of the most important and successful examples of product line engineering in the U.S. Department of Defense.

Lockheed Martin views its primary objective as developing once and building and deploying many times from one set of shared assets – principally requirements, source code, and tests. Feature-based variation in requirements and code enables building a member of the product line with or without a specific capability.

Challenges that pointed to the need for governance

Critical to the successful execution of the CSL was the organizational changes which were identified as an obstacle early on and addressed. As the CSL grew the need for a governance of the CSL became evident (Gregg et al. 2014). Governance is the practice by which the product line is managed and controlled. With AEGIS, management and control come not just from the development organization, as might be expected, but from stakeholders external to that organization, including the U.S. Navy, the U.S. Coast Guard, and U.S. Missile Defense Agency. Customer pressures for development specific to their individual needs are extremely strong, yet both the customer organizations and the development organization understand the overriding advantages of the product line approach and together have worked very hard to put in place a strong product line governance structure. Another challenge is that all configurations are in different stages of development at any moment, which requires strong governance to manage the product line through development cycles.

Internal governance

The internal governance includes weekly meetings to monitor and manage product line execution through all stages of development which led to a consolidation of program unique meetings and reviews. There is one weekly software build meeting for all products and programs. There is also one weekly coordination meeting with program managers and technical leads from all the programs, and a weekly cross-program MB-SEIT meeting where technical topics are discussed. The requirements review cycles and the consolidated Program Management Team (PMT) have led to a consolidation of meetings on the customer side as well. Internal governance comprises program planning and program execution. Program planning provides approval and direction on the configurations' desired capabilities, which will be documented in a capability fielding plan. The plan will provide approval to build new capability on a development branch, and direction to merge that capability into the CSL mainline, as well as direction on which builds will be used to support the road to certification. The program execution aspect involves approving product work packages and making a decision as to the maturity of the capability for deployment to ships. Program execution produces the artifacts to support approval and decision points by the strategic and program planning levels of governance. Program execution makes a recommendation, including test results and other supporting information, as to which CSL branch or mainline load should be used to support the road to certification.

Customer-oriented governance

External customer governance is also needed. With multiple contracts, different lines of funding, and competing resource needs, it became quickly apparent that the needs and desires of each program office needed to be coordinated and communicated to the developer with one voice. Governance was required to coordinate between the various government program offices and the developer. A major role of external governance is to resolve or at least mediate the tension inherent in PLE but especially inherent in a product line this large and complex and given the importance of the capabilities of each of the configurations. While LM worked to establish processes to manage a coordinated planning approach and day-to-day development activities, the government needed to put in place a structure and associated processes to ensure clear and consistent direction was being provided to ensure success for all programs. This structure required support for decision making at three levels: Strategic, Programmatic, and Technical. To this end, three decision-making bodies were created along with a set of governing artifacts.

From a technical perspective, early expectations were that all code changes, whether new development or maintenance, i.e. defect fixes, would be conducted on a single set of source code, i.e. the mainline. It became quickly apparent that not all program offices were satisfied with the risks associated with this approach. Programs nearing the end of their development had little appetite for allowing new development efforts, to put their mature code at risk. The solution was to provide for three avenues for code development: Mainline Development, Development Branches, and Event Branches based on a risk assessment. Additionally, a process for assessing risks and deciding where to conduct each new development was required. To this end a Joint Engineering Review Team (JERT) was established and co-chaired by lead system engineers from the government, with technical representation by all product users. The charter of the JERT is to (1) provide direction on the conduct of development efforts; and (2) provide guidance to the developer in all technical matters impacting more than one program. A Joint Program Management Team (JPMT) was established with government co-chairs and representation by all product users to address the programmatic perspective, In practice, the JPMT serves as the tactical decision maker for all cross-program issues. The charter of the JPMT is to (1) render schedule and funding decisions; (2) apply programmatic considerations to JERT recommendations and turn those recommendations into decisions; and (3) provide a means of escalating unresolved technical issues from the JERT.

Strategic decision-making is the charter of the Major Program Manager (MPM) Board. This board is chaired by three O-6 program managers with representation by all product users. Future baseline content and strategic direction is set by the MPM Board. The board reports directly to its members' respective Admirals.

Artifacts to support governance

A set of governing artifacts serves to combine the strategic, programmatic, and technical decisions and document those as official planning documents: New Development Fielding Plan, Branch/Merge Plan, and the Build Plan. The New Development Fielding Plan controls planning for new and current functionality by explicitly mapping those capabilities to baselines/configurations. This provides the developer with up-front information to make intelligent and efficient design decisions. The Branch and Merge Plan provides government control over execution of development. The plan documents the development strategy and assigns it to a Development Branch or the Mainline, as well as the planning the merge point for Development Branches. The Build Plan spans all baselines/configuration and controls what functionality is in development and when it will be delivered. It time-phases all requirements allocated to software into the master build rhythm. These three artifacts, taken together, broadly define the vast majority of all of the work in the CSL and represent the collaboration, cooperation, and compromise among all of the cognizant stakeholders.

Results

The external governance was codified in an Instruction (the Navy equivalent of a policy directive) signed by the Program Executive Office Admiral. This act represents the strongest possible endorsement of the governance structures and policies put in place for AWS. The Navy feels that the underlying product line approach has served the Navy and its partners well to this point. To date, this governance structure has proven sufficiently flexible to adapt and address any circumstances and scenarios arising that were not explicitly foreseen. It is hard to imagine how AWS could have been as successful without it.

Boeing: The Right Organizational Structure

Organizational structure has a significant impact on effective Product Line Engineering adoption. Traditional program management practice often leads to isolated programs that establish initial baselines by copying from another program's baseline. This is often called clone-and-own. The products produced by these independent programs may have highly similar aspects, but are developed and maintained separately. The anti-patterns here are lack of core engineering teams, lack of shared assets, unnecessarily unique or tailored processes, and hidden duplication of effort, just to name a few.

An organization applying Feature-based PLE, however, will employ the concept of the PLE Factory, as illustrated in Figure 1. A PLE Factory must be staffed and managed so that it can execute its functions efficiently and effectively. Many organizations employing PLE utilize a matrix organization where programs (responsible for delivering individual members of the product line) are supported by product line teams that manage and operate the PLE Factory. Standing up a product line team can bring new challenges, some of which are detailed below.

Process harmonization: Ideally an organization adopting PLE would make use of common processes, with some possible tailoring to specific business or customer needs. However, when retrofitting PLE into a multi-program organization, or when rapidly adding staff to a growing organization, it may be hard to attain process synchronization across the organization. An approach to achieve process harmonization that is scalable and flexible is to apply PLE to the process documentation. A dedicated team with primary responsibility to support process development, variability, maintenance, and process configuration management is a solid strategy. On-boarding new programs, and staff becomes easier and less subject to change from opinion.

Vanity reporting: In an environment of multiple programs, it is not uncommon to see requests for the same data reported in different formats. These are called vanity reports, because many programs cannot agree on a standard and expect unique reporting to satisfy their needs. This causes additional work within the product line team that doesn't add direct value to the final products. The matrix organization should address the inefficiency and develop standards that eliminate vanity reporting.

Meeting overload: With multiple programs come a multitude of meetings. These can be anything from weekly program status, change control boards, architecture coordination, or planning sessions. While these are necessary within a complex organization structure, they can take a small army to support. It is important to ensure that these meetings bring value and are not wasting hours. It is also important to balance individuals so no one person has an inordinate number of meetings or programs to support. Additionally, there is a trade-off between requiring program representatives to attend product line team meetings or having product line teams support program meetings.

Funding challenges: Program budgets are used by product line teams to operate their PLE factory. It is important to develop well documented strategies for how and when multiple program funds are used. Strategies such as "first program pays" or cost splitting are two common approaches. The "first program pays" strategy may be simple but can create hidden schedule dependencies between programs. Cost splitting takes upfront planning and coordination, while the development team must execute with discipline. When program budgets are tied to contracted development and services, such as government funding, it is crucial to have funding discipline, but also to understand the intellectual property implications to PLE factory shared assets. It is important to have adequate markings within shared assets and understand how they are used.

Role clarity: In this matrix organization there will be managers, architects, product owners, team leads and other staff. Some of these roles will support programs, while others support product line teams. Despite the common goal of developing quality products with agility and speed, lack of role clarity can create uncertainty and confusion when conflicting direction and information is generated. It is important to have well-defined roles in the organization with clear responsibilities and authority. These well-defined roles will also avoid challenges arising from change impact assessment and adjudication.

These are some lessons learned from adopting PLE within a matrix organization. Having broad organizational team support helps overcome the challenges like these for PLE adoption.

Raytheon: Carefully Planned Adoption

Incorporating PLE results in significant business value. However, reorganizing to establish and operate a PLE Factory takes effort. Companies that expect a full and immediate ROI with little or no initial investment will shut the product line down as quickly as they attempted to launch it when business measures do not immediately change for the better. Unrealistic expectations and lack of funding lead to inadequately launched product lines that frequently face shutdown after the next quarterly results are calculated; or the next "must win" contract is lost when the full costs of establishing a product line are incorporated into a single bid.

The solution to this state of affairs is a well-planned and incremental adoption of Feature-based PLE that (a) provides incremental return on incremental investment, so that at every stage the adoption is providing positive value; and (b) recognizes the different parts of an organization that must work together to ensure the successful establishment and operation of the PLE Factory.

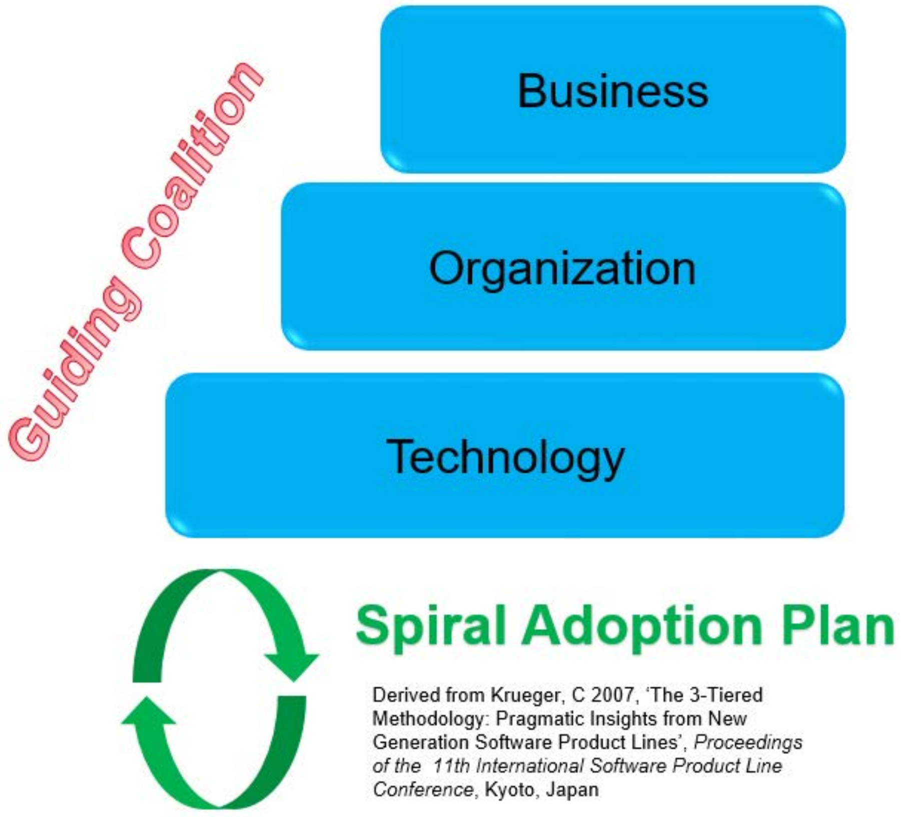
Business
Organization
Technology
Guiding Coalition
Spiral Adoption Plan
Derived from Krueger, C 2007, 'The 3-Tiered Methodology: Pragmatic Insights from New Generation Software Product Lines', *Proceedings of the 11th International Software Product Line Conference*, Kyoto, Japan

Figure 2. The PLE Adoption Methodology

While the PLE Factory plays the central role in the PLE story, it is only one part of a larger picture. Unlike other technical disciplines, PLE cannot be applied by individuals alone, but must be embraced by whole teams, projects, and organizations. PLE cannot be achieved with tooling alone, but must be part of a business strategy sponsored by executive leadership. Just as these things are true of actual manufacturing factories, they are true for PLE Factories.

For any engineering and management approach that requires an organization to change the way it does business, organizational change - the processes and steps needed to introduce and successfully use the new approach - becomes a necessity. Successful organizational change management for the incorporation of PLE techniques and tools impacts all aspects of an organization.

A full picture of PLE adoption includes:

- A Business Organization Management layer, driven by executive leadership, that focuses on the people, roles, and processes that utilize and leverage the PLE Factory to achieve the business objectives of the enterprise, as well as on the required processes for enterprise leadership to establish the PLE Factory, and to provide the necessary support for the PLE Factory during its operation. Using the analogy to a conventional factory, the Business Organizations tier provides guidance and support for the executive leadership working in the office high-rise that overlooks the factory. This layer allows the organization's management to steer the portfolio in strategic directions by defining products with new features or new feature combinations to, for example, enter a new market where the organization's ability to produce new products quickly and efficiently will give it competitive advantage.

- Technical Organization Management layer, which focuses on the people, roles, and processes that operate the PLE factory. In combination with a technological infrastructure, this provides a fully operational Feature-based PLE Factory capable of producing the System Asset Instances for all of the products in a product line portfolio. This layer concentrates on, among other things, reengineering the product assets into a collection of shared assets with variation points. In this tier, new roles specific to PLE are defined and filled, roles that move the engineers away from product specific responsibilities, and toward asset specific but product-independent roles.

- Technology that puts in place and maintains the tool and technology environment to operate the PLE Factory. Think of this as the fully functional factory but without any of the people inside to run the factory. Among other things, the Technology layer concentrates on incorporating the product configurator into the organization and beginning to use it to define a feature catalogue and shared assets for the product line.

The mastering of tiers is typically not sequential; organizations can begin building capabilities in each tier together. Further, adoption is incremental and does not happen all at once. Under a principle known as incremental return on incremental investment, each step toward complete incorporation brings commensurate benefit.

Three artifacts guide the transition and steady state operation of an organization incorporating PLE:

- A Living Business Plan owned by mission area leaders; this establishes precise need, importance, urgency, and vision for using PLE. It provides the over-arching vision for the organization's transition to PLE

- A product line Governance Concept of Operations (ConOps), executed by PLE Factory management; this is the set of processes for the day-to-day execution of PLE practices. The Governance ConOps establishes the tailored governance that will realize the Living Business Plan

- An incremental plan for creating, staffing, and operating the PLE Factory over time, known as a PLE Spiral Adoption Plan. This plan defines the path to achieve the transition from current state operations to PLE-based operations as described in the Living Business Plan.

The Guiding Coalition is a key factor for successful PLE incorporation. The Guiding Coalition is a group of leaders across all three tiers that are empowered to quickly make decision according to the team's decision making framework; and to continuously remove roadblocks for the product line teams.

Incorporation of modern PLE impacts multiple aspects of an organization, with numerous opportunities for failure if not executed in a well-informed and collaborative manner by a team. Institutionalizing and rewarding an *Ask for Help* culture is a critical success factor for high performance teams. This includes purposefully growing an organization's internal competencies thru on-the-job apprenticeships in partnership with embedded external SME's accelerating organizational change.

The model for PLE adoption that we have presented – recognizing three tiers of the organization, being guided by a Living Business Plan and a Spiral Adoption Plan, and establishing a Guiding Coalition to oversee the adoption and remove roadblocks – provides a methodical, incremental roadmap to a success roll-out of PLE that will provide positive returns at every step along the way.

General Dynamics Mission Systems: Measuring Success

When implementing Feature-based PLE in industrial practice, measuring the relative success of the PLE approach is paramount to provide the necessary evidence to support the claim that strategic, feature-based sharing is faster, more efficient, lower-cost, and higher-quality compared with the traditional approach to product development and maintenance. Business goals of efficiency and productivity often drive the decision whether or not to implement Feature-based PLE to reduce strain on limited resources including time, cost, and the human effort required to develop, deploy, and sustain products. By minimizing duplication of effort and exploiting commonality among assets across each product line, Feature-based PLE promises order of magnitude reductions in engineering overhead, development costs, defect rates, and time-to-market. We assert that these optimized business outcomes can be readily measured and tracked with four primary metrics of PLE success: product line change effectivity, cost avoided, shared asset utilization, and product line health.

Importance of measurement

Implementing effective measures of success provides PLE adopters the evidence necessary to support implementation and continued integration of the approach as a business improvement tool and force multiplier. Measurement of PLE has several purposes beyond simple business justification. One such purpose is to ensure the PLE implementation effort is meeting the intended goals; where a project may appear successful at face value, the actual outcomes may not align with the stated PLE objectives.

Furthermore, measurement of PLE implementation can reveal important data about the evolution of products, the product line, and the factory. To wit, insight about the relative growth, change, and architecture of the factory, product line, and products reveals divergence in necessary PLE processes and one-off, PLE-noncompliant development efforts. Proactive and consistent PLE measurement facilitates earlier detection of undesirable change, growth, and decimation of the factory. This measurement approach helps to manage and reinforce the adoption of PLE in the organization as well as to keep PLE on track and forestall backsliding once the factory management and processes become part of daily work and culture.

An additional purpose of PLE measurement is to inform individual-level performance appraisals. The incorporation of PLE measures at this level may have the unintended but beneficial effect of improving PLE implementation and maintenance via individual accountability for the new way of working, and potentially incentivize conformance with PLE processes. Providing individual-level PLE measurements as part of a typical performance appraisal offers employees pride in their new way of working, a language for communicating their PLE-related successes and contributions, and an opportunity to keep PLE at the forefront of important conversations among individuals, teams, programs, and organizations.

Change effectivity

One metric of PLE success is product line change effectivity, understood simply as the scope of applicability exerted by a single change to made inside the PLE Factory – either to one or more shared asset supersets, or to the feature catalogue, or to a Bill-of-Features. Every such a change will affect some number of products in the product line, either one or some or all. Change effectivity measures, on average, how many products are affected by a change made to the PLE Factory.

In practical terms, a change effectivity figure of, for example, 2.5 means that each change to the PLE Factory applies to 2.5 products. In a product line with three products, a change effectivity figure of 2.5 is impressive and indicates that a majority of the products in the product line are impacted by a single change to the PLE Factory. To calculate this measure, the total number of changes *effectively* made (to products) is divided by the number of changes *actually* made (to the PLE Factory). This requires that all changes are tracked within the product releases as well as the unique and common changes within the PLE Factory.

Cost avoided

A second metric of PLE success is the measurement of cost avoided with the approach. To determine this, historical program data must be collected, including program financial and engineering productivity data from development tools. This type of data should include requirements, pages, or source lines of code (SLOC) per hour, the average cost per change, and the number of changes made. To calculate this metric, first determine the program's average cost per change and count the number of changes actually made. Then, calculate the Change Cost by multiplying the number of changes made by the cost per change. Cost avoided is the cost of the changes effectively made minus the cost of changes actually made:

$$\text{Cost Avoided} = (\text{Change Cost})\text{Change Effectivity} - (\text{Change Cost})$$

where Change Cost = # of changes * cost per change

For example, where a program's average cost per change is \$100, and 490 changes are made to the superset, the total change cost is \$49,000 (e.g., 490 x \$100). Where that product line's change effectivity factor is 2.4, the cost avoided is \$68,600, calculated as follows:

$$\text{Cost Avoided} = (\$49{,}000)2.4 - (\$49{,}000) = \$68{,}600$$

Shared asset utilization

A third metric for determining PLE success is shared asset utilization, which indicates the size and breadth of shared assets, along with the commonality shared among them. Prior to calculating this metric, data about the size of shared assets must be collected from asset development tools, which may vary. For instance, requirements may be managed in DOORS, software in a source code IDE, test procedures in RQM, and training documentation in MadCap Flare. To calculate shared asset utilization, first measure the amount of common material in a shared asset, then divide that by the overall size of the shared asset. The formula to calculate shared asset utilization is:

$$C/O$$

where

- C = the amount of common material in a shared asset
- O = the overall size of a shared asset

We find it particularly useful to report this data in a simple table, organized by asset type, as follows:

Shared Asset Utilization			
Asset	**Superset**	**Production Capability**	**Commonality**
Software	# SLOC	# products, # SLOC	97.5%
Requirements	# modules	# products, # CDRLs	97.0%
Network Configuration	# SLOC	# products, # SLOC	94.7%
Test Procedures	# modules	# products, # CDRLs	96.3%
Training Documentation	# pages	# pages, # manuals	85.5%

Product line health

The final metric of interest is a simple assessment of product line health. To evaluate this, we monitor PLE Factory health and robustness via four primary areas: (a) production line metrics, (b) assertions, (c) guidelines, and (d) process deviation.

Production line metrics includes an assessment of the number of production lines, feature models, features, feature profiles, variation points, product profiles, and unowned feature models. Product line assertions (constraints on feature selections that are added to the feature catalogue) are counted and trended over time. Additionally, guidelines (comments and explanations inserted into the feature catalogue by engineers) are monitored in terms of the proportion of features, feature profiles, assertions, and logic for which guidelines are written, as well as the change in proportions over time. The total amount of guideline material and changes in guideline material over time also provide useful insight. Finally, process deviation can be evaluated with simple assessments of whether naming conventions are being followed and whether product owners are able to bypass factory protocols, indicating potential weaknesses in the factory.

General Dynamics Mission Systems: Best Practices for Building a Feature Catalog

We have identified a few examples of implementation patterns and anti-patterns through the course of our experience with adoption of Feature-based throughout GDMS. These patterns have either provided us a path to success or sent us red flag signals when we have seen teams going down problematic paths. The following practices apply to building a feature catalogue.

- **Top down modeling**. It is our experience that often it best to think about a program or product like the customer does. What is the language the customer uses to describe the features of the system? These are likely your highest-level features. Once these are enumerated, we find identifying subsystems helps us complete the modeling and guides our product line architecture, leading to a more composable product line.
- **Feature model size.** We hear this at nearly every PLE workshop. Are there any rules of thumb about size of the individual feature models that, together, make up a feature catalogue? When a model gets bigger then you are able to comprehend you should consider modularizing it. Software modularity and composability best practices will serve you well here. Individual feature models should exhibit a conceptual integrity and deal with a coherent area of subject matter. A large, complex, monolithic, includes-everything feature model will serve no one well.
- **Five Whys.** Features are distinguishing characteristics that members of a product line apart from each other. But they are also abstractions that capture those distinguishing characteristics in a way that is neutral with respect to any particular shared asset. When eliciting features from subject matter experts, they may describe distinguishing characteristics using shared-asset-specific language. Keep asking them why the difference exists. Sooner or later they will tell you that the reason is "just because." That's often the point at which you have arrived at a feature. We have found that the Five Whys method tends to get to the root of the feature (Wikipedia 2019).
- **Customer facing features vs engineering implementation features variation**. We want our customers involved in the process of discussing and evolving our production line. However, many features in our product lines capture distinguishing characteristics that are vital to the correctness and completeness of products, but do not capture a difference that is particularly relevant or interesting to a customer. This led us to creating a customer facing feature model that captures the way our customers discuss our products, and a separate feature model that drives the implementation of our products. When we meet with and discuss the product line with our customers, these changes that are captured in the customer facing model and these changes inform the changes in our implementation model.
- **Antipattern: Systems Model as Feature Model.** A feature is a distinguishing characteristic that makes one instance of your product line different from another instance of the product line. In a feature model, the only things we capture are features. Many people want to model their system in a feature model. They create features for elements of the system that are common – that is, the same for all members of the product line. These features are always selected for every variant of the system. Do not allow this to happen. If something is common, allow it to be common. There is no need to create features and wrap things in

variation when no variation exists. This just adds effort and costs to your implementation with no value.

- **Antipattern: Swiss Army Knife**. We have encountered a few developers who are so enamored with features and variation that they decide to attempt to model and vary every imaginable piece of their system. This leads to higher implementation costs and little realized value. Our rule of thumb is that teams should model the variation you have, not the variation you think you will have. Additionally, no feature should be modeled unless it drives some variation in one or more shared assets. A feature that makes no difference anywhere is a waste of time and effort. In our PLE implementations, you get credit for reusing something that someone else has shared. You get no credit for building something that someone might use in the future.

Conclusion

Although Feature-based PLE is becoming widely utilized in sectors other than aerospace and defense, it is enjoying a particularly widespread application in that industry. In this paper, practitioners from four of the largest A&D companies in the world have shared their experience-based ideas about some of the most important practices that enable or inhibit successful adoption and execution of Feature-based PLE.

Feature-based PLE is the subject of a forthcoming standard (ISO/IEC 26580, "Methods and Tools for the Feature-based Approach to Software and Systems Product Line Engineering") under the auspices of INCOSE's International Working Group on Product Line Engineering. This speaks to its importance throughout the engineering community as a mature, well-defined, automation-supported form of product line engineering. We hope that papers such as this one, which address real-world issues of Feature-based PLE from a practical standpoint, will further its maturation and adoption.

References

Chalé Góngora, H.G. and Greugny, F. 2017. "Where the Big Bucks (will) Come from — Implementing Product Line Engineering for Railway Rolling Stock." INCOSE INSIGHT Practitioners Magazine 22, no. 2 (2019), 15-24.

Clements, P., 2015. "Product Line Engineering Comes to the Industrial Mainstream," 2015 INCOSE International Symposium, Seattle.

Clements, P., and Linda Northrop, 2002. *Software Product Lines: Practices and Patterns*, Addison-Wesley.

Clements, P., et al. 2013. "A PLE-Based Auditing Method for Protecting Restricted Content in Derived Products," Proc. 2013 Software Product Line Conference, Tokyo.

Clements, P., et al. 2013. "Second Generation Product Line Engineering Takes Hold in the DoD," *Crosstalk, The Journal of Defense Software Engineering*, USAF Software Technology Support Center.

Clements, P., Susan Gregg, Charles Krueger, Jeremy Lanman, Jorge Rivera, Rick Scharadin James Shepherd, Andrew Winkler. 2014. "Second Generation Product Line Engineering Takes Hold in the DoD," Crosstalk, The Journal of Defense Software Engineering, Jan/Feb (2014). 12-18.

CNBC, 2019, “American firms rule the $398 billion global arms industry: Here’s a roundup of the world’s top 10 defense contractors, by sales.” Available at https://www.cnbc.com/2019/01/10/top-10-defense-contractors-in-the-world.html, accessed November 3, 2019.

Flores, R., et al. 2017. “Product Line Engineering Meets Model Based Engineering in the Defense and Automotive Industries.” In Proceedings of the 21st Software Product Line Conference, Seville, September 25-29, 2017, 175-179, New York: ACM.

Flores, R., Krueger, C., Clements, P. 2012. “Mega-Scale Product Line Engineering at General Motors,” Proceedings of the 2012 Software Product Line Conference, Salvador Brazil.

Gregg, S., et al. 2014. “Lessons from AEGIS: Organizational and Governance Aspects of a Major Product Line in a Multi-Program Environment.” In Proceedings of the 18th International Software Product Line Conference, Florence, Italy, 2014, 264-273. New York: ACM.

Gregg, S., Albert, D., Clements, P., 2017. “Product Line Engineering on the Right Side of the ‘V’,” Proc. Software Product Line Conference 2017, Seville.

Gregg, S., et al. 2016. “The Best of Both Worlds: Agile Development Meets Product Line Engineering at Lockheed Martin,” INCOSE International Symposium, 26, no. 1 (2016); 951-965.

Gregg, Susan P., Rick Scharadin, and Paul Clements. 2015. ”The More You Do, the More You Save.” In Proceedings of the 19th International Conference on Software Product Lines, Nashville, 2015. 303-310. New York: ACM.

INCOSE International Working Group for Product Line Engineering, “Feature-based Systems and Software Product Line Engineering: A Primer.” 2019. Available at https://connect.incose.org/Pages/Product-Details.aspx?ProductCode=PLE_Primer_2019, accessed November 3, 2019.

ISO/IEC 26550, Software and systems engineering -- Reference model for product line engineering and management.

Krueger, C. and Clements, P. “Systems and Software Product Line Engineering,” Encyclopedia of Software Engineering, Philip A. LaPlante ed., Taylor and Francis, 2013, available at www.biglever.com/wp-content/uploads/2019/01/Encyclopedia_Software_Engineering_PLE_Chapter.pdf.

Lanman, J., Brian Kemper, Jorge Rivera, Charles Krueger. 2011. “Employing the Second Generation Software Product-line for Live Training Transformation.” In Proceedings of Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2011.accessed May 2, 2019, http://www.iitsecdocs.com.

Northrop, Linda M., et al. “ 2012. A Framework for Software Product Line Practice, Version 5.0,” Carnegie Mellon Software Engineering Institute. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=495357, accessed November 3, 2019.

SDTimes, 2019, “New guidelines for featured-based product line engineering now available.” Available at https://sdtimes.com/softwaredev/new-guidelines-for-featured-based-product-line-engineering-now-available/, accessed November 3, 2019.

Teaff, J., and Young, B., “Applying Feature-Based Systems and Software Product Line Engineering in Unclassified and Classified Environments,” 2019 INCOSE International Symposium, Orlando, July 2019.

Wikipedia, “Five Whys,” 2019. https://en.wikipedia.org/wiki/Five_whys. Accessed November 3, 2019.

Wozniak, L., Paul Clements. “How Automotive Engineering Is Taking Product Line Engineering to the Extreme.” In Proceedings of the 19th International Conference on Software Product Lines, Nashville, 2015. 327-336. New York: ACM.
Young, B., et al. 2017. “Product Line Engineering Meets Model Based Engineering in the Defense and Automotive Industries,” Proc. Software Product Line Conference 2017, Seville.

Biographies

Susan P. Gregg is a Principal Project Engineer for the Lockheed Martin Corporation. She holds a B.A. in Physics from Rutgers University. She has over 30 years’ experience in systems and software engineering. She is currently the Technical Director for the US Navy’s Common Product Line and has co-authored many published papers on applying Feature-based Product Line Engineering at Lockheed Martin.

David Hartley started his career at FedEx where he held many roles such as Software Engineer, Technical Advisor and Enterprise Architect. In addition to working at FedEx, David spent some time doing consulting work for companies such as Citibank and Blue Cross Blue Shield of Florida. David has been at Missions Systems for 3 years working on the Pando Project, which is a corporate initiative to promote and implement Product Line Engineering (PLE). As the product owner for the PLE Center of Excellence, David leads an agile team whose mission is to maintain and execute the governance and processes necessary to execute PLE across GDMS. The CoE prepares PLE-ready product teams who understand PLE principles, use PLE tools, establish governance, become self-sufficient and assist others to adopt PLE. David holds a Bachelor of Science in Computer Science from Florida State University and a Master of Business Administration from the University of Central Florida.

Dr. Morgan A. McAfee advocates feature-based product line engineering at General Dynamics Mission Systems (GDMS) within a corporate initiative promoting and implementing PLE across many GDMS teams, projects, programs, and products. She holds a Ph.D. in Measurement, Methodology, and Analysis from the University of Central Florida, along with an M.A. in Educational Psychology.

.

Randy Pitz is an Associate Technical Fellow for The Boeing Company where he has supported several defense and commercial programs as a software engineer, team lead, principal investigator, and Chief Architect. He has over 27 years' experience in systems and software engineering. He currently acts as a PLE champion across Boeing and provides guidance and experience to interested programs. Randy holds a Masters in Computer Science from Washington University, and his Bachelor of Science from Michigan Technological University.

James Teaff wrote his first line of software as a professional in the early 80's while working for a tech startup. Subsequently over the past several decades he has worked for numerous aerospace and defense contractors across the full system development lifecycle, from principal investigator and proposal writer to IPT Lead, Chief Architect, SEIT Lead, requirements analyst, Scrum Master, programmer, tester, 2nd tier O&M support, and more. James holds a Master of Engineering in Engineering Management degree from the University of Colorado, and a Bachelor of Science degree in Computer Science from Colorado State University. He is an INCOSE Certified Systems Engineering Professional (CSEP) and is an active member of the INCOSE PLE International Working Group.

Dr. Paul Clements is the Vice President of Customer Success at BigLever Software, Inc., where he works to help organizations adopt Feature-based Systems and Software Product Line Engineering. Prior to this, he was a senior member of the technical staff at Carnegie Mellon University's Software Engineering Institute, where for 17 years he worked leading or co-leading projects in software product line engineering and software architecture documentation and analysis. Prior to the SEI, Paul was a computer scientist with the U.S. Naval Research Laboratory in Washington, D. C.