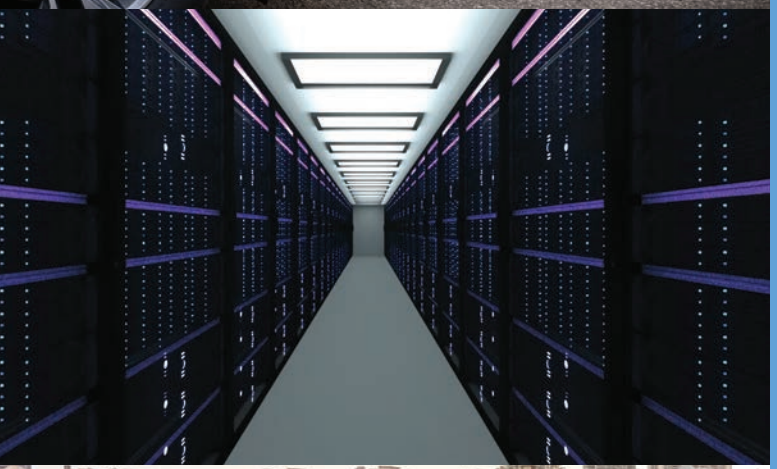
INCOSE

International Council on Systems Engineering

A better world through systems approach

Feature-based Systems and Software Product Line Engineering: A Primer



Feature-based Product Line Engineering lets you build your product line portfolio as a single production system rather than a multitude of individual products.

Why Product Line Engineering?

INCOSE's Product Line Engineering International Working Group spearheaded the ISO standard 26580 on Feature-based Product Line Engineering. This primer serves as an introductory companion to that standard.

Product Line Engineering (PLE) has long been known for delivering significant improvements in time to market, quality, and cost of systems.* *Feature-based Product Line Engineering* is a proven, well-defined, repeatable, automation-centric approach to PLE that is now delivering even greater improvements throughout some of the most challenging engineering industries.

Virtually all systems and software engineering is performed in the context of a product line. Hardly anyone builds just one edition, just one flavor, of anything. Product lines are found in every industry, including aerospace, defense, automotive, medical, consumer electronics, computer systems, energy, telecommunications, semiconductor fabrication, software applications, e-commerce, and industrial automation. Product lines occur under every business model, including retail, government contracting, OEM, business-to-business, value-added reselling, and custom development.

As businesses everywhere strive to achieve competitive advantage and greater profitability, the need to elevate systems engineering to system family engineering is universal.

There are constant pressures to decrease cost and time to market. The exponentially growing complexity of product line portfolios and how they are created and produced are pushing organizations to the edge of their capability. As the mundane tasks of managing this complexity increasingly consume engineering teams, they lose the opportunity to create and fully leverage new product innovations.

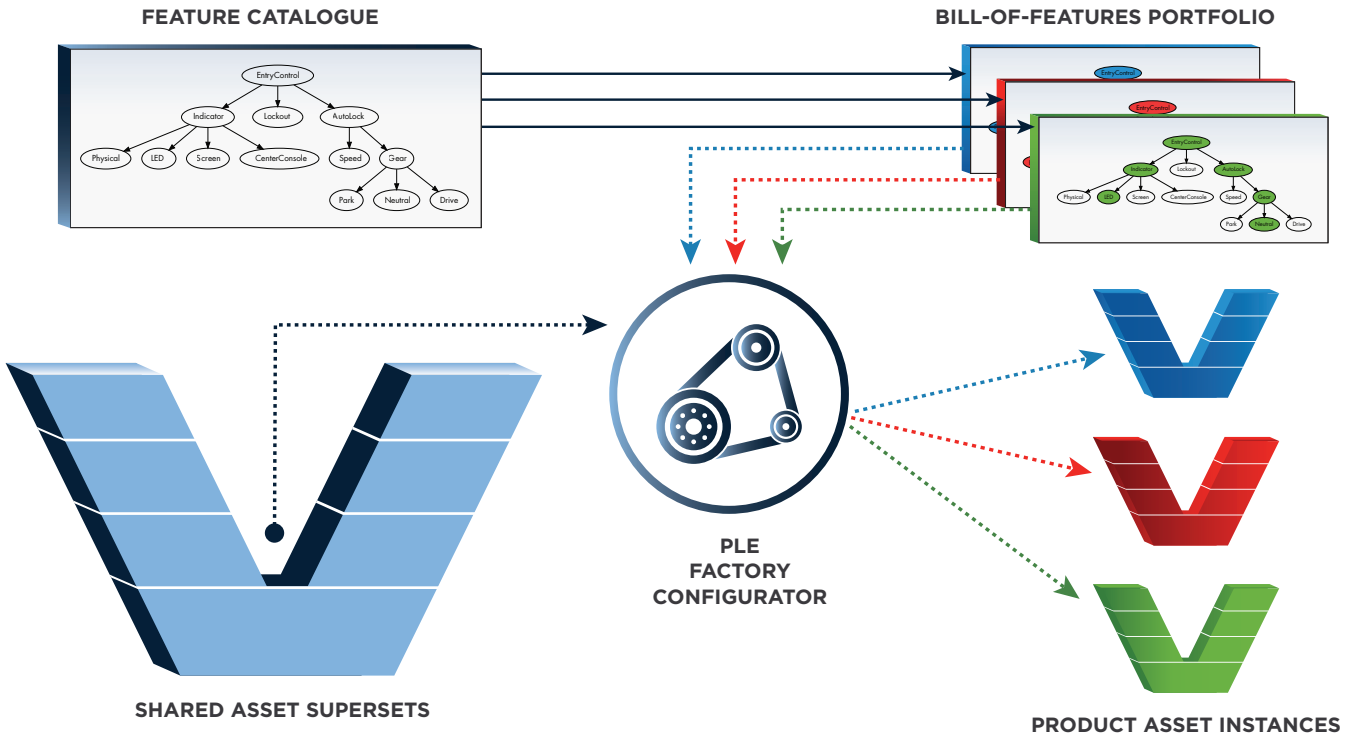
This primer provides a starting point to learn about this powerful approach. It offers a brief introduction to PLE, explains how it works, and provides sources of information to learn more.

Even in the world of Aerospace and Defense, gone are the days when government customers are satisfied with the traditional "clone and own" reuse approaches where each product variant is custom fit to a customer's needs. This traditional reuse method does not give rise to agile and faster deliveries. Customers are demanding frequent and modular upgrades, lower development costs, and faster deliveries to keep pace with their competition and changing operational contexts.

* Linden et al. *Software Product Lines in Action: The Best Industrial Practice in Product Line* Engineering. 2010

"The Department [of Defense] is transitioning to a culture of performance and affordability that operates at the speed of relevance… We will prioritize speed of delivery, continuous adaptation, and frequent modular upgrades."
— James Mattis, U.S. Secretary of Defense, April 2018
https://www.defense.gov/News/Article/Article/1503359/mattis-asks-house-committee-to-build-on-recent-dod-successes/

The solution:
The Feature-based PLE Factory



Organizations utilizing Feature-based PLE adopt a factory approach to building their products. The figure above illustrates the important concepts of a Feature-based PLE Factory in operation, as defined in the ISO standard.

- **Shared assets** are artifacts that support the creation, design, implementation, deployment, and operation of products. They can be represented digitally and configured, and are shared across the product line.

  Each shared asset is a superset that contains variation points, which are pieces of content that should be included in or omitted from a product based on the features selected for that product.

- 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.

Shared assets:
A key concept of Feature-based PLE

Shared assets are the "soft" artifacts associated with the engineering life cycle of the products. A shared asset can be any artifact representable digitally: requirements, design models, source code, test cases, BoMs, wiring diagrams, documents, and more. They either compose a product or support the engineering process to create a product.

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 configured according to the feature choices 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).

Configuration options typically include selection or omission of the content; selection from among mutually exclusive content alternatives; generation of content from feature specifications; and feature-based transformation of content from one form into another.

Requirements

Perhaps one of the easiest shared assets to understand is requirements. A superset of requirements combines individual product requirements to establish the product line requirements. Variation points achieve inclusion and omission, define mutual exclusion, and transform requirement wording in the product specification – all based on feature selections. Requirement transformation can replace constants, units, or other text with values that are derived from the feature model. Requirements that have no variation are part of every product.

Models

When models are used in product development, they can be developed as supersets and instrumented with variation points for the product line. For example, system design or architecture models using SysML or UML can be instrumented through variation points, which apply to structural elements such as processes, objects, classes, states, use-cases, packages, and others.

Electrical and Mechanical Designs

Shared assets for Electronic Design Automation (EDA), and Mechanical Design Automation (MDA), and Computer-aided Design (CAD) for electronic, mechanical, mechatronic, and cyber-physical systems take the form of supersets of parts, properties, and relationships in Bills of Materials (BoM), assemblies, subsystems, circuit boards, wiring harnesses and more in EDA, MDA, and CAD models. Variation points instrument optional, mutually exclusive, and varying content in the models.

Source Code

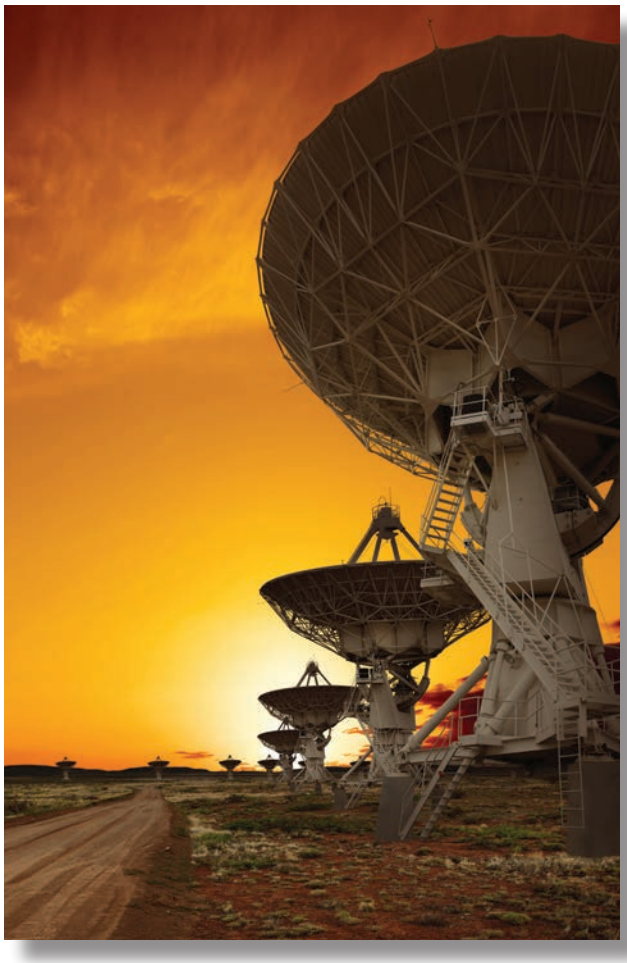
In software systems, shared assets can include the source code, the models used to generate the source code, build files, and automated unit tests. Source code can be configured in several ways including individual blocks of code, files, or build files. A modular software architecture may assist greatly with the feature instrumentation but is not necessarily required. Features might align closely with the system's software, electrical, and/or mechanical modularity, or they might be cross-cutting where a feature affects several different modules.

Test Plans and Test Cases

For all types of systems there should be validation and verification artifacts that include test plans and test cases. These may use automated or manual techniques, but in either case should be supersets instrumented with variation points along with the other shared assets in the product line. Eliminating sources of unnecessary testing is often one of the first visible benefits of PLE. Furthermore, it is possible to streamline or even eliminate redundant testing of common capability across the product line.

Others

There are many other types of shared assets that might serve in a product line, such as product budgets or cost models, schedules and work plans, user manuals and installation guides, process documentation, marketing brochures, wiring diagrams, simulation models, engineering drawings, product descriptions, and contract proposals. These assets in their simplest form may be documents, spreadsheets, or presentations, but more complex or customized systems may be used to formalize them. In any case, these assets can be integrated into the PLE factory.



A Consistent Approach to Variation

Imagine that a requirements engineering team has embraced a variation management technique based on tagging requirements in a requirements database with attributes that differentiate feature variations in requirements. Further, the design team has adopted a SysML tool and has embraced inheritance as the mechanism for managing design variations.

The software development team is using an informal feature model drawn in a graphical editor, plus macro directives, build flags, and configuration management branches to manage implementation variations. Finally, the test team has adopted clone-and-own of test plan sections, stored in appropriately named file system directories to manage their PLE test plan variations.

Now imagine what would be needed to create a complete PLE life cycle solution that integrates into a larger business process model. How do the requirements database attributes and tagged requirements relate and trace to the subtypes and supertypes in the design models? How do these attributes and supertypes relate and trace to the macro flags, CM branches, and test case clone directories? Trying to translate and synchronize among the different representations and characterizations of features and variations creates dissonance and chaos at the boundaries between stages in the life cycle.

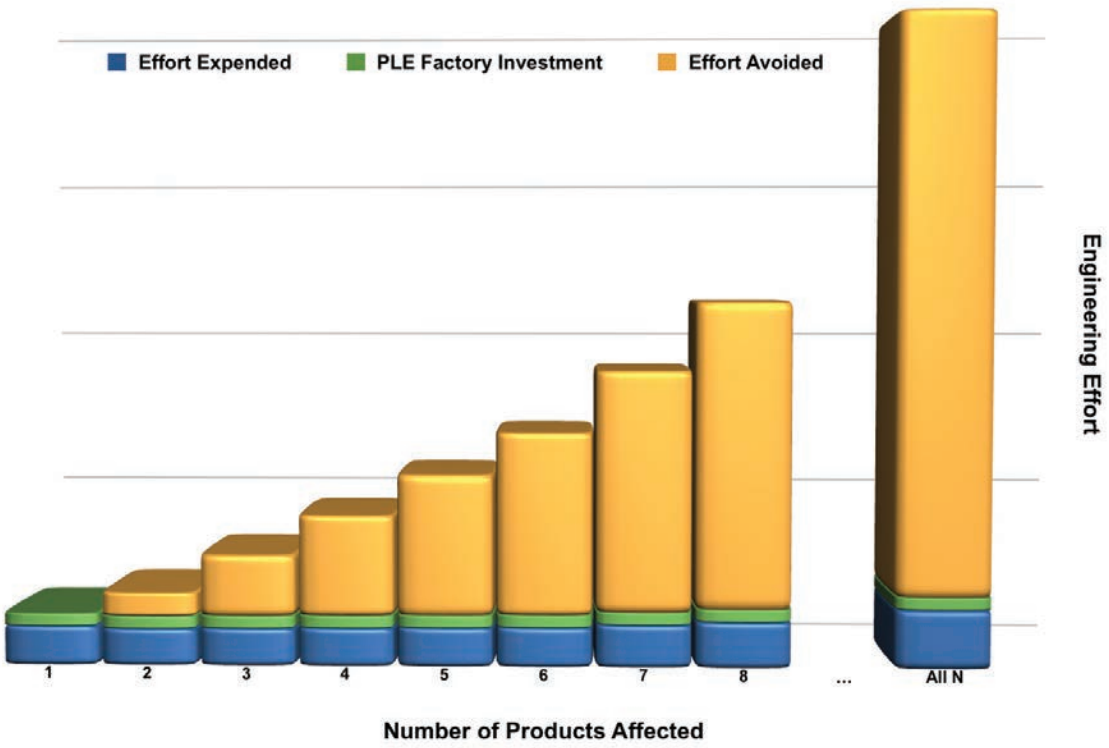
To resolve the quagmire brought about by different disciplines each using its own approach to variation, a key aspect of Feature-based PLE is consistent and traceable treatment of variation across all shared asset types. Feature choices are the basis of a common language of variation across all disciplines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

The economics of Feature-based PLE

Organizations that adopt PLE as a key business strategy consistently garner significant competitive advantage in the form of more wins, more innovative and higher quality systems and products, faster engineering and business velocity, and higher revenue and profits.

Feature-based Product Line Engineering eliminates duplication in artifacts and replication of work, resulting in the leanest, most efficient engineering effort possible.

As an organization carries out its daily engineering work, that work can be characterized by how many products in the organization's portfolio each piece of work affects.



Capturing a new requirement, repairing a defect in a piece of software source code, writing a more comprehensive test case, adding a new piece of hardware to the Bill of Materials, altering a design model: Each of these tasks, and more, can be characterized by how many products in the portfolio it affects.

Suppose a task affects four products. In a product-centric environment, each product's team will apply that task. Under Feature-Based PLE, if the organization undertakes a task again affecting, say, four products, that task is carried out once, inside the factory. The task will involve changing or adding to the shared asset supersets, or the Feature Catalogue, or the Bills-of-Features for the products. Then, the configurator is used to apply the work to each product that needs it.

In the figure above, each bar represents work that applies to a certain number of products. The blue segment of each bar measures the engineering effort of a task. No matter how many products benefit from the task, the task is only done once, consuming one "unit" of effort. The gold segment of each bar measures the engineering effort avoided through the automation of the configurator. The small green segments represent the cost of applying Feature-based PLE: building the Feature Catalogue, the shared assets with variation points, etc.

The engineers carrying out the task in our example have done the work of a team four times their size under the old approach. They can claim with absolute justification that they are four times as productive as their pre-PLE-factory counterparts.

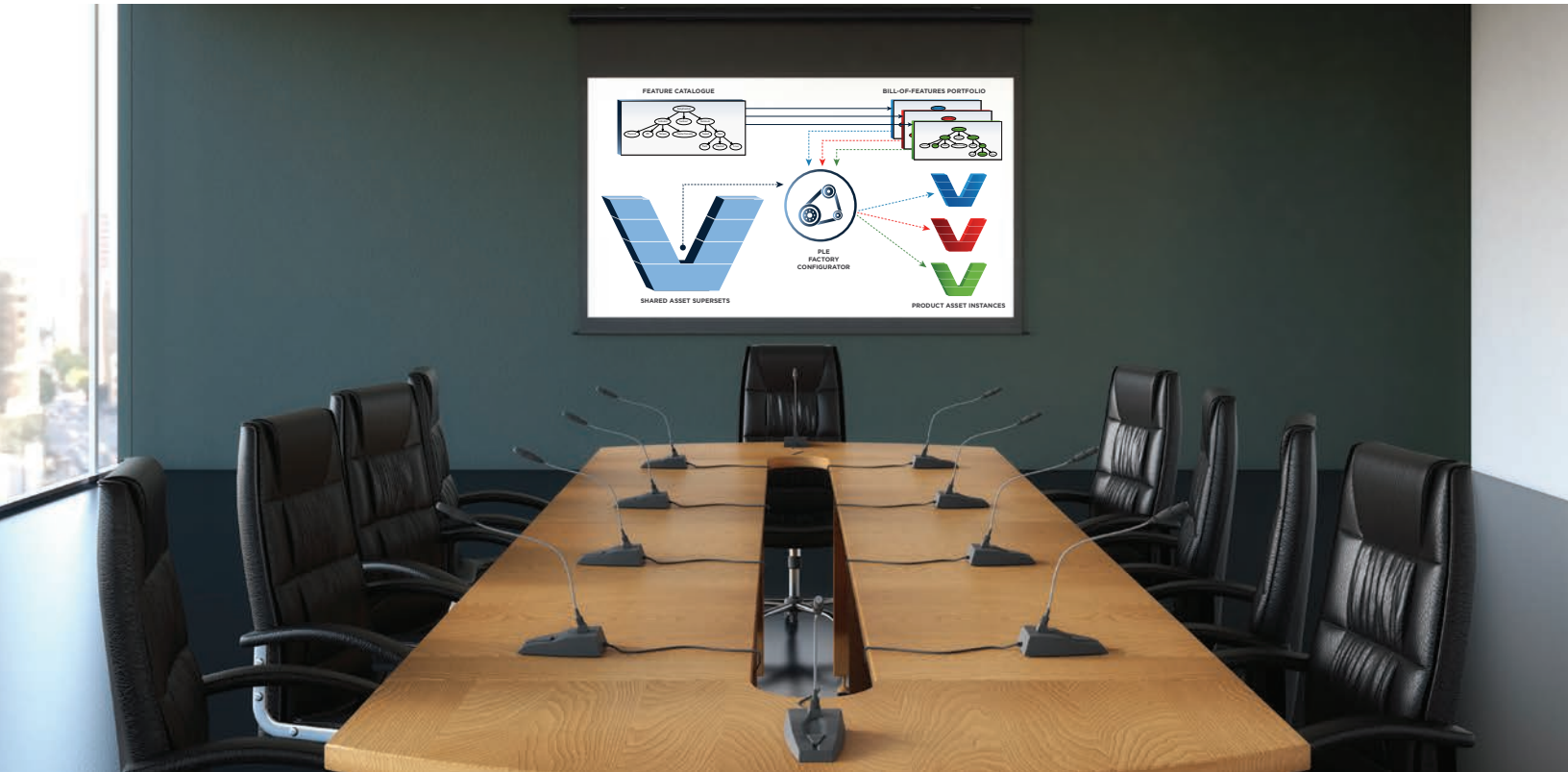
Feature-based PLE for the enterprise: PLE from the break room to the boardroom

PLE earned its wings, and its ongoing reputation for substantial savings, in engineering. However, Feature-based PLE should not be sequestered in just the engineering sphere. Organizations expend enormous amounts of time and effort dealing with product feature diversity to manage manufacturing and supply chains, certification and compliance documentation, product marketing and product portfolio planning, e-commerce web system deployments, sales automation, training, support, service, maintenance, disposal, and more. Feature-based PLE, with its central paradigm of feature selections driving variation realization, works not only in organizations' engineering arms, but in their operations arms as well.

In many companies, the marketing, capture of opportunities, design, development, and delivery of products is the output of different teams who work in silos and follow different processes. This often leads to a misalignment between the benefits that are actually obtained and the business strategy of the company (which translates into lower profits), and between the delivered product and the services the product is supposed to provide (which usually translates into customer dissatisfaction).

To avoid these problems, alignment and collaboration among many functions in the organization is crucial: from strategy functions to customer-facing functions, program management functions, technical functions and industrial functions. In other words, all roles — marketing, sales, bid teams, project management, systems engineering, product policy, domain and specialty engineering, as well as procurement, manufacturing and installations — must agree upon, share, promote, and comply to the same vision of the product line scope. In all of these areas, features can formalize the product line's domain of variability.

To the extent it is adopted throughout an organization, Feature-based PLE provides the opportunity to create that alignment and collaboration. Its central concept of feature powers diversity management at all levels, from the most minute and mundane to the most strategic and company-defining. Decision-makers can use features to make strategic decisions about portfolio management, entering new markets, pricing, and more, and those feature decisions can flow seamlessly down through product realization.



Organizational PLE adoption

Effectively adopting Feature-based PLE involves much more than just installing a tool to manage the Feature Catalogue and configure products. Leadership commitment — not just tacit approval — is a critical ingredient for success.

Feature-based PLE involves a change in mindset, from product-centric thinking to product line thinking, as well as an organizational change: to create, staff, and operate the PLE Factory. While it in no way fundamentally changes the engineering processes in an organization, it does extend them in specific ways.

For example, requirements engineers will continue to work on requirements, as always — except now they will work with a superset and create variation points. The same applies for software engineers, test engineers, technical writers, analysts, architects who build design models, and so on.

Governance is key. Change control boards continue to function as before, but changes now involve shared asset supersets and the feature catalogue — and they are reviewed and vetted from the perspective of the entire product line.

These and other changes must be introduced into the organization's culture, and the organization's natural tendency to continue working as before must be overcome.

As a result, tenets from the organizational change community* are appropriate and helpful in a PLE adoption. Among other things, that community teaches us that:

- Strong, effective, and repeated communication throughout the organization of the importance and (especially) the urgency behind the adoption of PLE is essential.
- An incremental, purposeful, goal-oriented, step-by-step adoption plan is necessary, to ensure that the organization's transition to PLE happens in a gradual, manageable fashion. Each increment should include a short-term win, an improvement brought about by PLE, that can be advertised and celebrated, increasing the momentum of the adoption.

* For example, see Kotter, J. P. Leading Change. Boston: Harvard Business School Press, 1996.

Tools are easy. People are hard. The hardest obstacle to overcome when adopting Feature-based PLE is not technological, but cultural. Fortunately, the high ROI on PLE makes the organizational change pill easier to swallow.

Feature-based PLE in practice

Feature-based PLE is currently in use in organizations of all sizes and across a wide range of industry sectors. Examples include:

- A global aerospace and security firm providing the US Navy with a critical strategic defense system is saving tens of millions of dollars every year, equal to the cost of their entire engineering staff.
- A leading aviation supplier is producing certification packages for their safety-critical flight products eight times faster than before.
- A major network storage company is enjoying 300% to 500% improvements in scalability, time to market, and product quality.
- A global aerospace and defense company has saved more than $800 million over a twelve-year period, while increasing the productivity of their engineers to 280% of previous levels.
- One of the world's largest automotive manufacturers has calculated cost savings of tens to a hundred million dollars per year, from configuring just one type of shared asset.

Published case studies include:

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.

Signs you're not practicing Feature-based PLE

You're not practicing Feature-based PLE if:

- You don't have a Feature Catalogue, explicitly owned and managed, and under revision control. Different functions and factions in your organization have diverging opinions on what the product line is and is not.
- You aren't using a feature-based configurator tool that configures your engineering artifacts. You're creating asset instances by hand, or by building and maintaining ad hoc scripts.
- You aren't using a consistent variation management approach, based on features, across your engineering artifacts.
- Products are not defined in terms of features.
- You are permitting, either explicitly or through casual process enforcement, clone-and-own practices, increasing your technical debt and burdening the organization with repetitive work.
- Products in your product line have their own development teams. The PLE Factory and product development teams are fighting for funding.
- Your product roadmaps and investment plans are hampered by technical decisions taken unilaterally by customer project teams.
- Engineers are applying manual changes to multiple copies, not supersets.
- You have no PLE-based governance structures in place.
- Your customers are dictating changes to products that bypass the PLE change control authority and governance structures.
- Your development teams hesitate to perform modifications because they are tangled up in a branching misery of product versions.

Additional resources:

1. ISO/IEC 26580 Software and systems engineering - -Methods and tools for the feature-based approach to software and systems product line engineering.
2. Beuche, D. 2008. “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, 358-367. New York: ACM.
3. 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.
4. 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.
5. Krueger, C., et al. 2017. “An Enterprise Feature Ontology for Feature-Based Product Line Engineering,” INCOSE International Symposium, 27, no. 1 (2017); 951-965.
6. 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.

Feature-based Systems and Software Product Line Engineering: A Primer is offered as a COMMUNITY SERVICE from the International Council on Systems Engineering (INCOSE). INCOSE’s intention is to introduce and explain the ISO standard 26580 on Feature-based Product Line Engineering to the world’s systems engineering community.

We encourage the document’s widest use, including reproductions, translations, adaptations/ derivatives with only three restrictions:

1. Permission for use of images, unless indicated as in the Public Domain, must be acquired for derivative works. Please contact INCOSE for Image contact information.
2. Please mark your material: derived from Feature-based Systems and Software Product Line Engineering: A Primer © 2023 by INCOSE.
3. Commercial uses of this document require INCOSE’s prior approval.

In view of the minimal restrictions for any use of this Primer, please send an electronic information copy of any document created with or from this Primer to our INCOSE Administration Office at info@incose.org

INCOSE

International Council on Systems Engineering

A better world through systems approach