BigLever's 3-Tiered Product Line Engineering Methodology and Organizational Adoption Strategy

Contact Information:

info@biglever.com
www.biglever.com
512-777-9552

Successful product line engineering involves a technical strategy but also an intentional, focused, and comprehensive organizational adoption strategy. BigLever's 3-Tiered Product Line Engineering Methodology provides the definitive model for a fully-functioning product line organization, while our Spiral Model for Organizational Adoption is the key to designing and following a successful incremental path to PLE adoption.

1. Introduction

Product line engineering (PLE) has shown itself to be an engineering paradigm unmatched in delivering improvements, often up to ten-fold, in cost savings, time to market, engineering productivity, product quality, and more. Numerous case studies give ample testimony to PLE success in some of the most challenging business sectors, application domains, and market segments, and in organizations that range in size from among the world's largest to those with an engineering staff of twenty or less.

PLE savings come from sharing across a family of similar products or systems: Strategic, planned sharing of design, development, testing, defect detection and repair, deployment strategies, project management and planning, portfolio management, market analysis, and much more. The more sharing an organization can initiate and manage across their product line, the greater the savings. Maximizing sharing requires an organizational as well as a technological re-thinking of how products in a product line are produced.

Successful product line engineering involves an organizational as well as a technical strategy and change. There needs to be a clearly articulated vision of the end state, and a model for systematically and incrementally achieving it.

This report deals with an industry-proven methodology for achieving a high-payoff product line operating capability, and a model for incrementally rolling out that methodology across an organization that aspires to achieve the many benefits of PLE.

PLE Technology

BigLever Software's approach to PLE relies on a feature-based factory approach, centered around the Gears configurator. Figure 1 illustrates the important concepts of the factory, which we call the Gears Product Line Engineering Lifecycle Framework:

- *Features* are defined to describe the products in the product line in terms of the features each product presents; a *feature profile* defines the set of feature choices associated with one product.
- *Shared assets* (such as requirements, source code, test cases, and much more) are endowed with *variation points,* which are places where the assets differ to support different products. The shared assets are maintained as supersets.
- The Gears product *configurator* exercises the variation points to configure the shared assets to support the feature profile of a chosen product.
- This whole system — a configurator using feature profiles to exercise variation points in shared assets — is a *production line.*

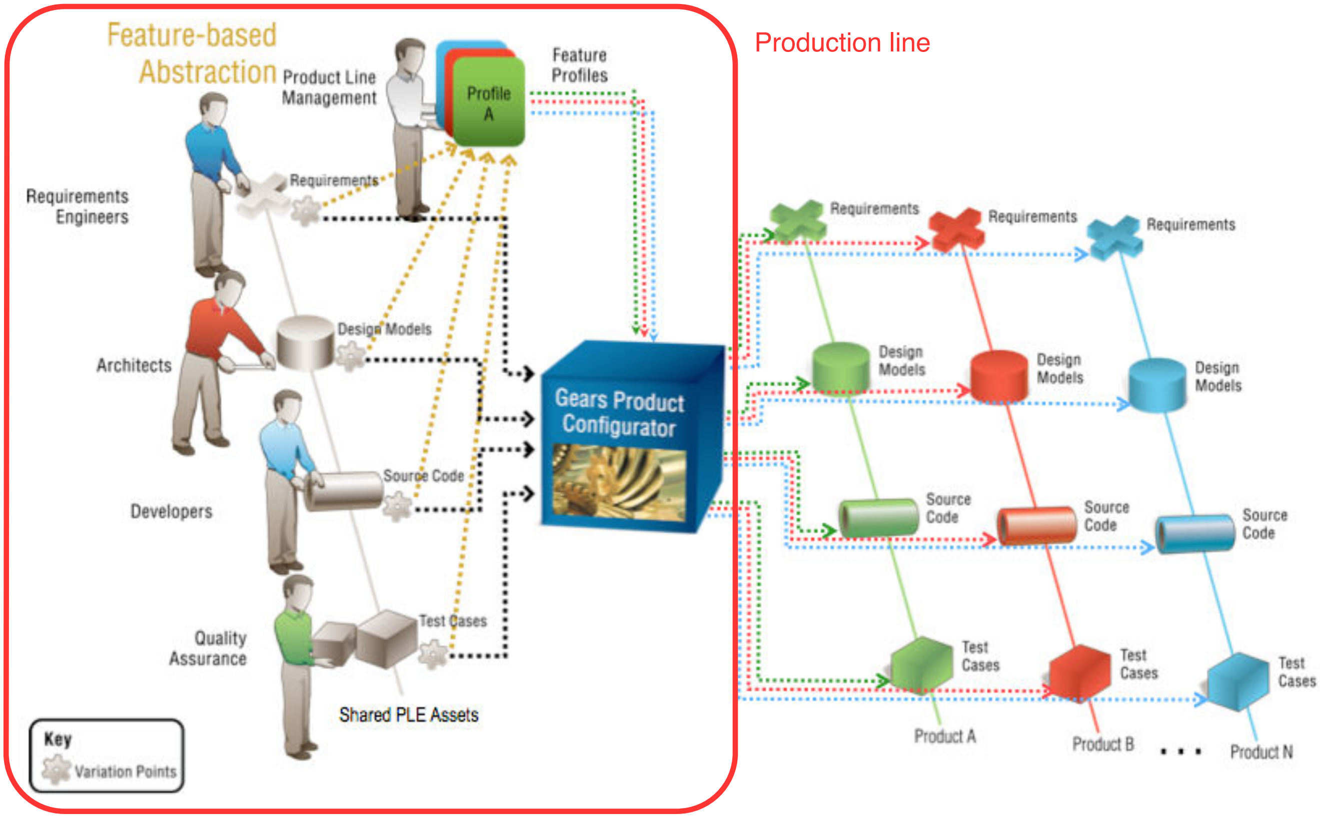

Figure 1: The Gears configurator uses feature profiles to exercise variation points in shared assets, to configure them to support products.

Automated production of products from feature profiles and shared assets allows a single consolidated change to a product asset to be automatically reconfigured into all products in the product line portfolio. This eliminates the need for manual merging of shared asset modifications into multiple products.

Successful transition to PLE requires a shift in thinking from product-centric approaches to a portfolio-wide sharing perspective.

The definition of "shared asset" in the methodology is very liberal. Essentially, any legacy artifact supporting a product can serve as a shared asset at this level, so long as it consolidates commonality (i.e., eliminates duplication, branching, and other forms of divergence), contains zero or more variation points, and can be used by the product configurator to instantiate products. Shared assets can (and do) include requirements, design models, test artifacts, work plans, source code, user manuals, bills of materials, planning documents, and more.

PLE Organizational Change

While the Gears PLE Lifecycle Framework provides the technological approach to sharing, this must be complemented by an organizational approach. Sharing will not happen effectively unless the appropriate organizational structures, processes, workflows, training, management incentives and remediation, and more are put in place. Organizations who wish to transition to PLE must overcome inertia, the tendency to continue carrying out their current practices. Successful transition to PLE requires a shift in thinking from product-centric approaches to a portfolio-wide sharing perspective.

To manage such a shift, organizations need a vision of the desired end state, and a realistic step-by-step plan for achieving it. Lacking one or the other, an organization is likely to flounder, trying ad hoc approaches or approaches that lead to failure. Further, it will be extremely difficult to motivate the engineering staff to change the way they carry out their tasks, and equally difficult to create a business case that convinces executive leadership to sponsor and support the activities necessary to bring about the needed change.

This report addresses both needs: The need for an end-state vision, and a plan for achieving it. Both are based on BigLever's broad-based success in bringing organizations of all sizes and domains to the PLE vision.

2. The 3-Tiered PLE Methodology

BigLever's 3-Tiered PLE Methodology provides a vision of an organization in PLE "steady state." That is, it delineates the capabilities that an organization will be able to execute on a daily basis in the running of its production line capabilities.

As companies shift from conventional product-centric engineering to PLE-based approaches, three tiers of capabilities and benefits are established, sometimes in sequence and sometimes in parallel.

As companies shift from conventional product-centric engineering to PLE-based approaches, three tiers of capabilities and benefits are established, sometimes in sequence and sometimes in parallel. Each tier builds upon and is enabled by the capabilities and benefits of the lower tier. That is, the capabilities at each tier provide direct benefits, but also enable increasingly more strategic capabilities and benefits in the higher tiers. The base tier provides a set of engineering technology capabilities and benefits, which enables a middle tier of engineering management capabilities and benefits, which ultimately enables the top tier of highly strategic capabilities and benefits in the executive and business operations.

This partitioning of capabilities and benefits into three distinct tiers provides a modular methodology that is easy to understand and explain to the practitioners who will be tasked with executing it. The well defined relationship between the tiers reduces the number of options and clarifies choices when defining and adopting a PLE approach.

The 3-Tiered PLE Methodology, shown in Figure 2, provides a holistic view of a product line organization in operation, a view that incorporates the PLE methods and techniques described in Figure 1. This report will describe the purpose and function of each tier in turn.

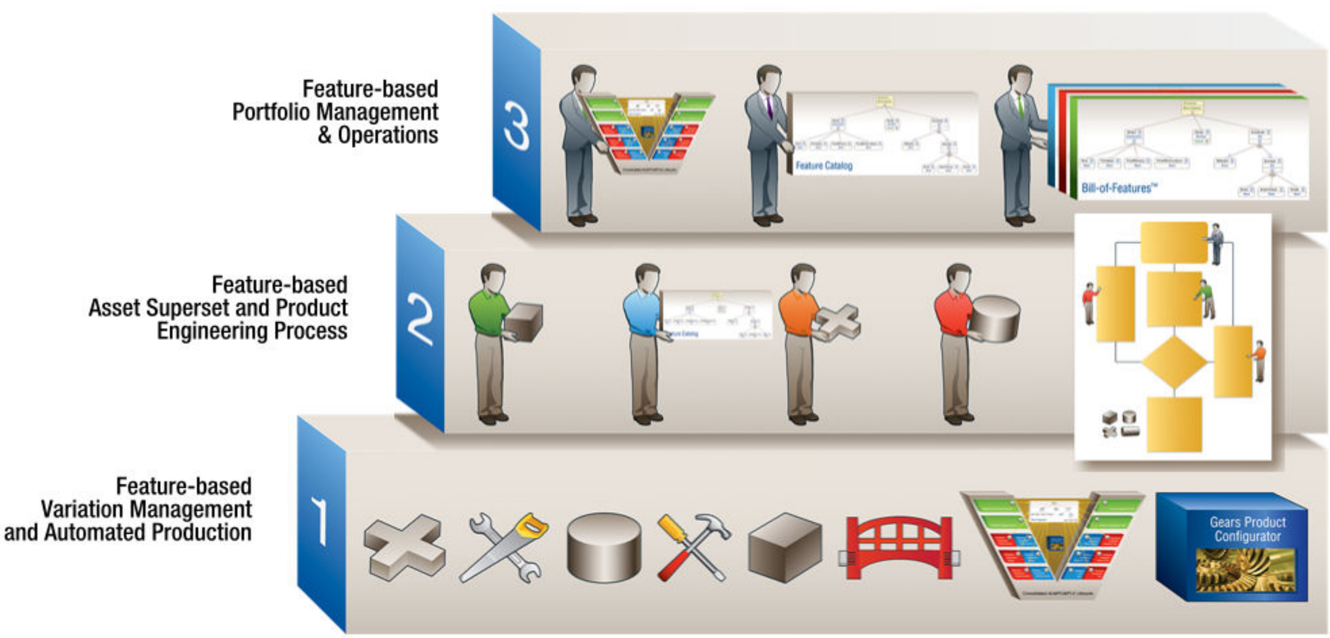

Figure 2. The 3-Tiered Methodology for PLE

Base Tier: Feature-Based Variation Management and Automated Production

The base tier, *Feature-Based Variation Management and Automated Production*, provides the foundation for PLE practice.

As shown in Figure 1, Gears takes two types of inputs – *shared assets* and product *feature profiles* – in order to automatically create product instances. The shared assets can include requirements, architecture and design models, source code, test cases, user documentation, and much more. The product feature profiles are concise abstractions that characterize the different product instances in the product line portfolio, expressed in terms of a *feature model*[14].

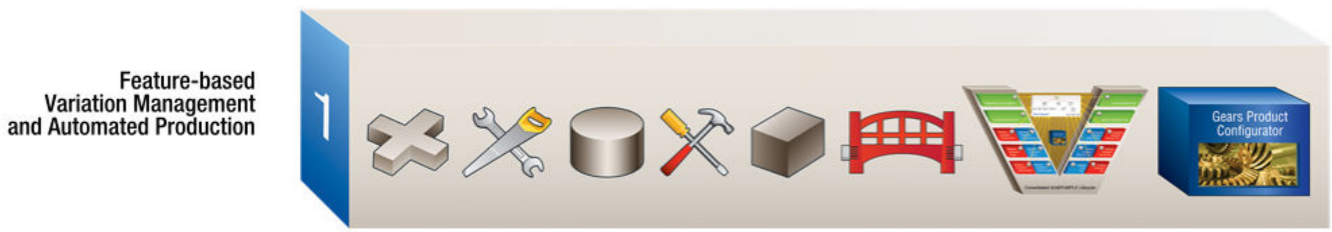

Figure 3. Base Tier: Feature-Based Variation Management and Automated Production, with a focus on automation and technology installation

In the base tier, the focus is on establishing and operating the basic infrastructure for first-class variation management in the product line. The essential base tier capabilities are provided out-of-the-box by the BigLever Software Gears PLE Lifecycle Framework.

In the base tier, the focus is on establishing and operating the basic infrastructure for first-class variation management in the product line:

- feature modeling capabilities, to capture feature models that characterize the variation available from product to product, and feature profiles that capture the feature choices reflected in each product;
- employing a uniform mechanism that supports variation points in the shared assets. Most non-PLE organizations have adopted a combination of ad hoc variation management techniques – we have seen as many as *29 different* home-grown variation management mechanisms used in combination in one portfolio. For example, a requirements engineering team might tag requirements in a requirements database with attributes that differentiate variations in requirements. A design team might adopt a UML tool and embrace inheritance as the mechanism for expressing variations. The development team might use #ifdefs, build flags and configuration management branches to manage implementation variations. Finally, the test team might adopt clone-and-own of test plan sections, stored in appropriately named file system directories to manage their test plan variations. In this scenario, how do the requirements database attributes and tagged requirements relate and trace to the subtypes and super-types in the design models? How do these attributes and super-types relate and trace to the #ifdef flags, CM branches, FODA features, and test case clone directories? Translating between the different representations and characterizations of features and variations creates dissonance at the boundaries between stages in the lifecycle. Impact analysis is almost impossible. Gears does away with all this by providing a small number of variation mechanisms that can be used across the entire collection of engineering assets that support an organization's products.
- installing and running the automated production mechanism — the Gears product configurator — to instantiate products by exercising the variation points in the shared assets according to the feature profiles provided.

These essential base tier capabilities are provided out-of-the-box by the BigLever Software Gears PLE Lifecycle Framework.

At the base tier, conventional assets that support the products (such as requirements, design models, legacy source code, test assets, and more) and conventional organizational structures (such as teams organized around products) are sufficient. Unlike early (and now outdated) approaches to PLE, there is no need for specially engineered *software product line architectures,* or a dichotomy of engineering roles for *domain engineering* and *application engineering*.

First-class variation management involves two key concepts. The first is using a small and consistent set of variation mechanisms to drive the variation points engineered into each of the assets. An example of a variation mechanism is to select one from a set of available variants to serve as the asset in a product. Another example is to insert placeholder text in an asset (such as a requirements document) to make it generic across the product portfolio, and then perform pattern-substitution to replace each placeholder occurrence with text appropriate to an individual product.

The second concept is to express the variations in terms of the feature model of the product portfolio. A feature model describes the features that each product exhibits and the features that set the products apart. Expressing variation in terms of product features (as opposed to describing asset variations in asset-specific terms) allows a single expression to drive the configuration of every asset.

First-class variation management and automated production put in place under the auspices of the base tier serve to:

- eliminate duplication, cloning, divergence, and merging
- consolidate the multiple ad hoc variation management mechanisms typically found in legacy software but also in other engineering assets such as requirements or test cases
- eliminate the manual and parallel product production efforts found in conventional approaches.

The feature-based variation management and automated production capabilities established in the base tier offer significant increases in engineering productivity.

The first-class variation points in the shared assets allow the duplication and divergence to be consolidated into shared assets with combined commonality and variation points. The feature modeling and variation point mechanism consolidate the multiple ad hoc variation techniques into one first-class approach. The feature model expresses the portfolio variation explicitly in a single consolidated representation, rather than implicitly and diffusely defined throughout the implementation of the various assets in the portfolio.

Benefits. The feature-based variation management and automated production capabilities established in the base tier offer significant increases in engineering productivity. Less time is spent dealing with duplication, divergence, and merging; less effort is expended creating and dealing with multiple, complex, ad hoc, home-grown variation techniques; and less overhead is incurred on Order-N^2 combinatoric coordination among products. Consolidation with first-class variation points means the products are easier to comprehend, structure, and maintain.

Because labor costs often dominate the cost of building products, productivity and cost benefits go hand-in-hand. Productivity increases mean lower cost per product and per feature.

New product feature profiles supported by existing feature models and shared assets can be used to immediately and automatically configure the new products. New product feature profiles supported by existing feature models but not yet fully supported by the shared assets can be easily mapped into requirements for suitably extending the shared assets (for example, by adding new variation points). New product feature profiles not supported by existing feature models and shared assets can be easily mapped into requirements for suitably extending the feature models and shared assets

The engineering effort required for any new product or feature is pure *delta engineering*, where the only new engineering required for a new product instance is precisely what is lacking in the current assets. Everything else across the portfolio engineering lifecycle is fully reused.

Compared with early PLE methodologies, the base tier of the 3-Tiered PLE Methodology provides a much simpler entry point into PLE practice. Variation management and automated production can be adopted incrementally and with little disruption to ongoing production schedules when using the Gears product configurator that provides these capabilities out of the box [4]. Legacy assets can be heavily leveraged with little or no re-engineering. Existing team structures can be maintained with little or no reorganization. The processes and day-to-day activities can remain virtually unchanged. The primary benefit gained from these capabilities in the base tier is lower engineering overhead compared to conventional approaches and therefore higher productivity and lower per-product cost.

Middle Tier: Feature-Based Asset Superset and Product Engineering

The capabilities and benefits of the base tier — putting the automation in place to achieve automated product production using the factory paradigm — enable the middle tier of the methodology, *Feature-Based Asset Superset and Product Engineering*. Here, the focus is more organizational, to align with and complement the technological changes brought about by the base tier. The focus of the middle tier is on organizing the products' engineering assets and the teams that build and maintain them.

Feature-based Asset Superset and Product Engineering Process

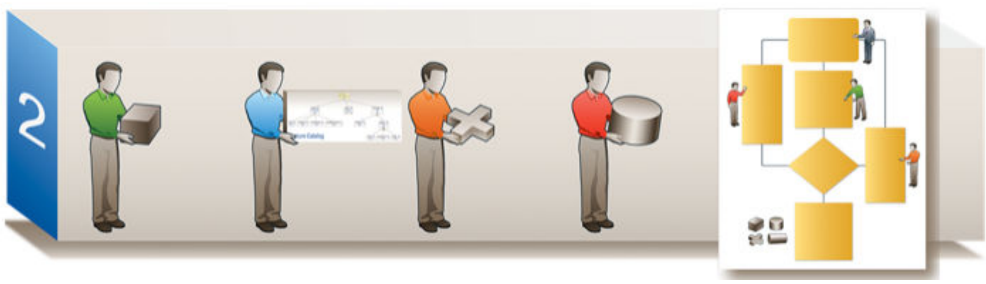

Figure 4. Middle Tier: Feature-Based Asset Superset and Product Engineering, with an emphasis on structural alignment with shared asset engineering

Organizing teams around product-focused engineering is ineffective for portfolio engineering. For engineering management, it introduces significant time, effort, cost, and overhead to staff and deploy a team for each and every product added to the portfolio. Isolated product engineering contexts degrade the potential for sharing and lead to duplication of effort across product teams. All product teams have to establish expertise for the entire product, resulting in a steep learning curve and long productivity ramp for new engineers and developers. For engineering management, there is a divergent focus on the lifecycle, project schedules, and release events for multiple products in the portfolio. This leads to high management overhead and a complex management task of resource allocation and coordination across product teams.

The middle tier of the 3-Tiered Methodology centers on establishing *engineering focused on shared assets*. Engineering management organizes teams around *shared assets* rather than products.

By contrast, the middle tier of the 3-Tiered Methodology centers on establishing *engineering focused on shared assets*. Engineering management organizes teams around *shared assets* rather than products. The shift from product-centric to shared-asset-focused approaches enables engineering management to manage the portfolio as a single system rather than managing a multitude of products in a duplicative fashion. In the middle tier, engineers learn to build and maintain shared assets that apply to the entire product line, not just individual products. Hence, the engineers' purview expand as they form shared asset engineering teams to build and evolve engineering assets whose scope is the entire portfolio.

Being able to organize teams around shared assets rather than products results in an extremely stable organizational structure. In contrast with *application engineering* approaches that have to scale the organization with each and every product added to the portfolio, and transfer engineers as individual products come and go or ramp up and ramp down, the shared asset focused capabilities created in the middle tier of the methodology mean that the organizational structure around shared assets is stable – it is very similar for 2, 20, 200, or 2000 products.

A common strategy for establishing a PLE approach in an organization is to start with existing legacy assets from products developed using conventional techniques. In the base tier of the methodology, little or no modification is required to utilize the legacy assets. However, in the middle tier, refinements may be beneficial in order to optimize the effectiveness of shared asset focused engineering.

The key to effective engineering is modularity in the shared assets. The modularity can be informal, such as simply partitioning assets along subsystem lines and storing them in their own subdirectories in the file system structure. As long as asset teams can work with relative independence, then the modularity is sufficient for the middle tier.

In cases where the legacy assets are tightly coupled in a monolithic asset base, shared asset teams can still be assigned according to subsystems within the monolith. We have seen that the "gravity" provided by the shared asset teams can be leveraged to incrementally refactor the monolithic system into an architecture that exhibits more well defined modular boundaries.

In contrast to early product line methodologies, in which engineering management created a dichotomy of engineering efforts, one for *domain engineering* and one for *application engineering*, the 3-Tiered PLE Methodology does not require the dichotomy nor any *application engineering* teams.

Temporal management in the middle tier. Of course, the shared assets evolve over time (from left to right in Figure 3), and this evolution must be managed. BigLever's innovative PLE baseline management technique handles this while substantially reducing the complexity and overhead of traditional multi-product configuration management.

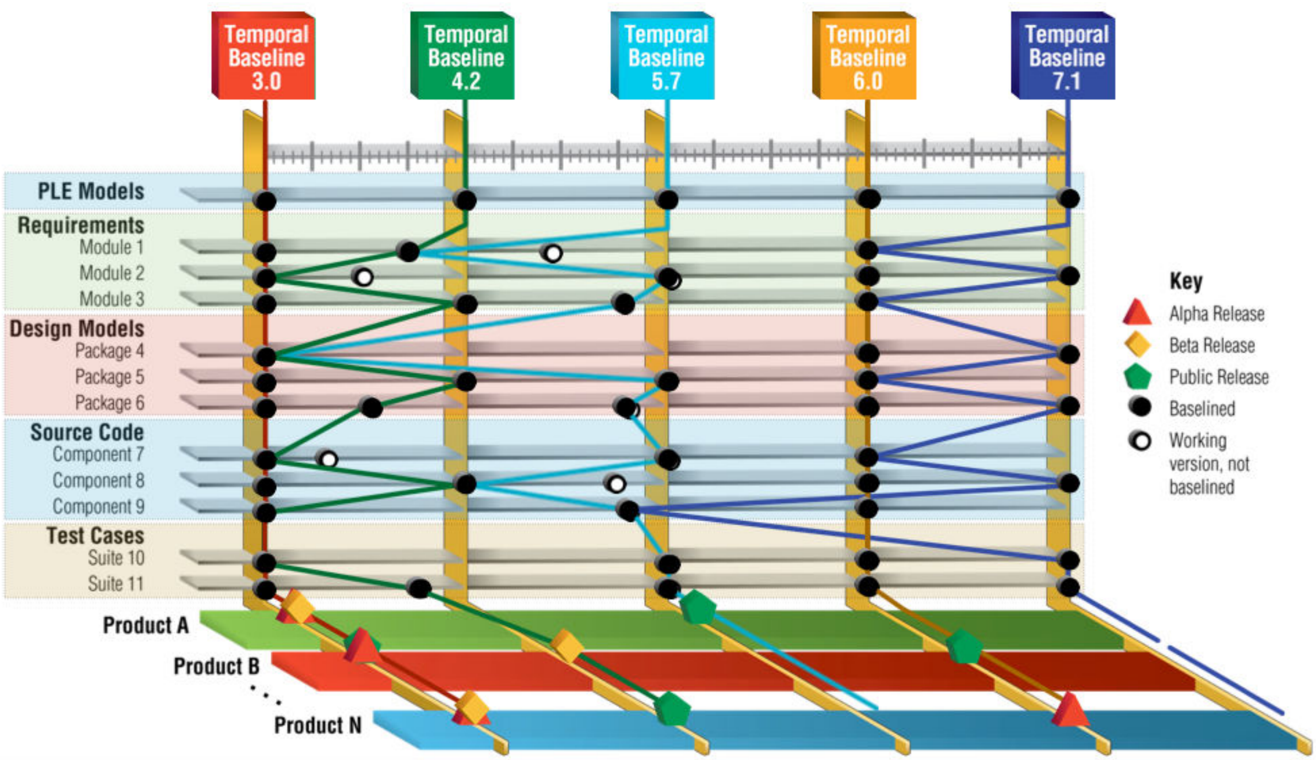

Figure 5. Temporal baselines help mange evolution of shared assets over time

BigLever's PLE Baseline Management Methodology focuses on the full set of PLE shared assets, and not the individual products. A new version of a product is not derived from a previous version of the same product, but from the shared assets themselves, using Gears. The baseline management approach is based on the concept of a *temporal baseline.* Figure 3 illustrates:

- A number of shared assets are arrayed down the left hand side, from various lifecycle phases. Each asset (shown notionally in the figure as requirements modules, design model packages, source code components, and test case suites) undergoes evolutionary change, each maintained under its normal configuration management process. Each asset's evolutionary trajectory extends to the right.
- The light-colored horizontal bands in the main part of the diagram indicate that the various shared assets *may* be maintained in physically different repositories: a DOORS database for requirements, for example, or a filesystem CM repository. Our approach is independent of repository or physical location.
- The PLE models themselves — Gears files — are also put under change control and stored in a CM repository.
- The bottom shows the products in the product line: Products A through N. Each product goes through various phases, such as Alpha, Beta, and Public releases.
- Across the top are several temporal baselines. A temporal baseline is simply a list of shared assets and the version of each that was used to build a product.

BigLever's PLE Baseline Management Methodology focuses on the full set of PLE shared assets, and not the individual products. The shared asset engineering capabilities established in the middle tier offer significant increases in quality.

This simple CM process for the product line eliminates many potential configuration errors and leads to reliable fielding of products. It also allows each of a product line's shared assets evolve at their own pace. The Gears configurator in the heart of the production line enables this simplified CM scheme and complexity reduction, because it makes it practical to re-generate any number of end products affected by a change in a shared asset.

Benefits. The shared asset engineering capabilities established in the middle tier offer significant increases in quality. Fewer defects are introduced by asset teams and as a result the entire portfolio displays higher degrees of quality. Higher quality means less time fixing defects during development, faster and more efficient test cycles, and less time and cost dealing with defects in the field. All of this leads to higher customer satisfaction.

Shared asset focus provides just what its name implies – better focus for individual engineers and teams. There is no need for all teams to understand all aspects of a product implementation. The narrow focus of a shared asset within the engineering lifecycle means that new staff members can get up to speed and become productive much quicker. The portfolio is developed as a single system rather than a multitude of products, reducing the labor-intensive combinatorics for coordination and knowledge sharing.

The organizational structure is much simpler for engineering managers to establish and maintain, compared to a multitude of product teams with shifting resource and deadline requirements. Because the shared asset structure within a product line architecture is much more stable during portfolio evolution than the number of products in the portfolio, the organizational structure tends to be much more stable as well. This organizational structure is also suitable for geographically distributed teams.

The degree of reuse in shared asset engineering is extremely high. All engineering effort on the shared assets in this methodology is candidate for reuse – 100% reusable – for any or all products in the product line. Even when shared asset engineering effort is devoted to a variation point that is specific to one product, it is possible to reuse that asset on any product at any time in the future.

Because shared asset teams are narrowly focused, they tend to gain deep expertise. That expertise can be leveraged to create a portfolio of feature rich, high quality, and highly competitive products. Asset teams gain pride of ownership which facilitates higher morale and higher quality.

Compared with early PLE methodologies, the middle tier of the 3-Tiered PLE Methodology provides a simpler and more effective approach by eliminating the need for *application engineering* (AE) teams and focusing solely on what the earlier methodologies called *domain engineering* (DE). Avoiding the dichotomy of having two types of engineering teams – AE and DE – eliminates many negative implications, including the "us-versus-them" cultural dissonance sometimes reported between AE and DE teams [1][6][7][8].

Top Tier: Feature-Based Portfolio Management and Operations

Shared asset engineering at the middle tier plus variation management and automated production at the base tier eliminate the need for manual and product-focused engineering. The capabilities and benefits in the base and middle tiers enable the top tier of the 3-Tiered PLE Methodology, *Feature-Based Portfolio Management and Operations*. The focus of the top tier is on business-wide management of the entire product line portfolio using the enhanced production line capabilities provided by the lower two tiers.

Figure 6. Top Tier: Feature-Based Portfolio Management and Operations

Here, executive leadership and portfolio managers can utilize the concept of feature-based product capabilities to:

- Exploit the time to market capabilities to reinforce existing market segments or even enter new markets;
- Build product offerings that integrate PLE capabilities from across the organization: That is, build products as product lines of product lines;
- Optimize supply chain management, as well as management of logistics and distribution to support products in the field
- Streamline product training and support

With conventional product-based portfolio management, the business side of an organization – such as the marketing team or the product support group – provides a set of requirements for each product needed in the portfolio. Marketing, Support, and Engineering will then each review and negotiate the content and schedule for the new product release roadmap. That approach has several drawbacks. Because the full requirements set is used for each product, there is a high overhead for creating and negotiating the requirements, schedules, and roadmaps for new products. This results in slow time-to-market for the products, which ultimately limits the number of products that can be effectively deployed and maintained in a portfolio. From a business perspective, this has a negative impact on strategic business operations. There is a lack of agility to quickly react to market opportunities, changing market conditions and market turbulence. The limited number of products means that the business has to pass up market opportunities.

Rather than manage the portfolio evolution on a product-by-product basis with full requirements specifications on each product, the portfolio should be managed by feature specifications, plus feature profiles for the products. Features become the primary concept and the *lingua franca* for managing the portfolio across the entire enterprise. Products are a derivative concern.

The PLE approach thus now extends outside of the engineering organization to include business areas such as marketing, support, training, and more. Using domain-level feature concepts, non-technical business roles can use a concise feature profile rather than full-scale requirements document to define a new product within the current scope and to specify where the scope needs to be extended to accommodate new feature requirements.

Benefits. The strategic business benefits of the PLE approach are fully realized in the top tier of the 3-Tiered Methodology. The benefits come both in the form of reductions in time-to-market for new products and features, as well as increases in the scalability of the portfolio.

The strategic business benefits of the PLE approach are fully realized in the top tier of the 3-Tiered Methodology. The benefits come both in the form of reductions in time-to-market for new products and features, as well as increases in the scalability of the portfolio.

Product feature profiles are conceptually simple descriptions for the Product Marketing team to create and to negotiate about with the Engineering team. New and extended feature requests can be simply expressed in terms of "deltas" with the feature model and product feature profiles.

Evolution of the portfolio is based on features. This evolution is driven by product marketing and motivated by the features have the most business value for the portfolio. Product marketing and engineering negotiate future work and product line scope according to the cost/benefit of adding features to the portfolio.

The size of portfolio can scale as large a needed to meet business demands and opportunities. The number of products in the portfolio loses relevance – this number can scale to the size needed to meet market opportunities rather than being limited by engineering cost and resources [4].

The business now has an enhanced competitive advantage. It has the agility to rapidly and precisely evolve the portfolio and to expand into new markets. Products can be brought to market very quickly. The business can survive and even capitalize on turbulent market conditions.

As the business transitions from product-based to feature-based portfolio evolution – where the entire portfolio evolves by adding or modifying feature requirements for common, optional, and varying features – the result is extremely efficient and concise communication between the business, marketing, and engineering teams, leading to optimized time-to-market and product line scalability.

Three Tiers Together

An operational view of the 3-Tiered PLE Methodology is illustrated in Figure 7 using a different rendition of the PLE factory described in Figure 1. Here we see the classes "V" model for system engineering. The "V" on the left is populated by shared asset supersets. Variation points (shown by the small gear symbols) are expressed in terms of the features in the product line's feature catalog. Feature profiles (here called Bills of Material) feed Gears, which configures the shared assets to produce the product-specific V's on the right.

In the top tier, the marketing, product support, or other business teams define the features to be added or modified for the product line portfolio to satisfy a business opportunity. These features are mapped into shared asset implementation requirements by the product line architecture team in the middle tier. Shared asset engineers then extend the shared assets as needed to implement the new feature requirements, utilizing the PLE infrastructure from the base tier to automatically configure, build, and test the products in the portfolio, and ultimately to build and deploy the final products for customer release.

This picture, empowered by the capabilities of the three tiers, shows a PLE organization in day-to-day steady state operation (augmented by the view of product line evolution shown in Figure 5).

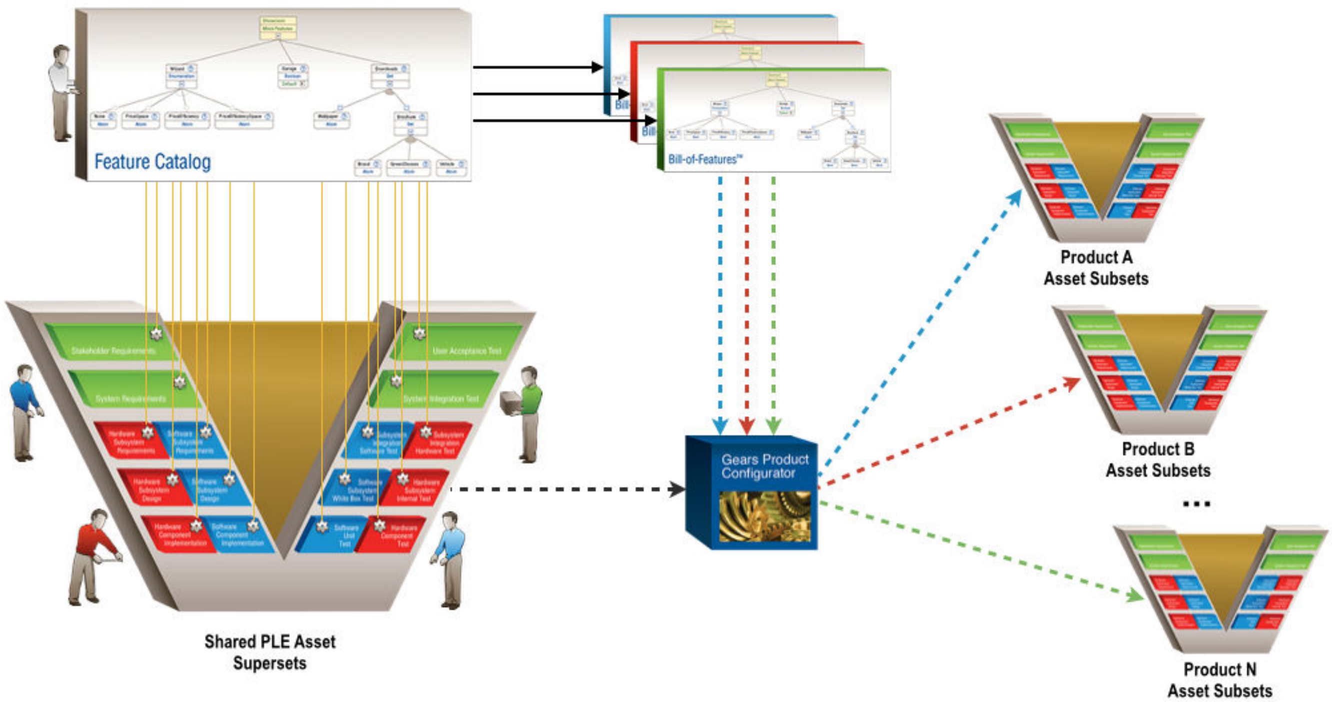

Figure 7. The three tiers in operation together

3. Organizational Transition to PLE

The obvious tendency with a transition to PLE practice using the 3-Tiered Methodology is to think bottom up. That is, begin with the base tier, and then start on the middle tier, in sequence, after the base is fully established.

While this style of transition is simple and effective, we have found that it is also effective to address the tiers incrementally and in parallel. In Figure 2, imagine the incremental transition effort growing from left-to-right rather than bottom up. Initially the base tier is incrementally addressed. Once sufficient capability is established in the base tier, then activities can begin in the middle tier of the methodology. Similarly, the top tier activities can commence once there is sufficient capability established in the middle tier.

Other transitions are possible, as well. For example, an organization may have a well defined business practice that utilizes *portfolio management* in terms of *features*. Although we have not seen this type of transition, there is nothing in the 3-Tiered Methodology to rule out starting with an established top tier and then “backfilling” the middle and base tiers.

To guide an organizational transition to PLE, BigLever has created the Spiral Model for PLE Adoption (Figure 8). Spiral models chart an organization’s trajectory through various activity areas. Each spiral takes the organization farther away from the starting point and into advanced capability. Each spiral represents an incremental improvement, the scale of which is carefully planned so as to not overtax an organizations ability to accommodate change.

As you traverse each spiral, more and more PLE capability is rolled out across more and more of your organization, based on a plan and tempo tailored to your specific needs and situation. At each step, incremental adoption brings incremental benefit.

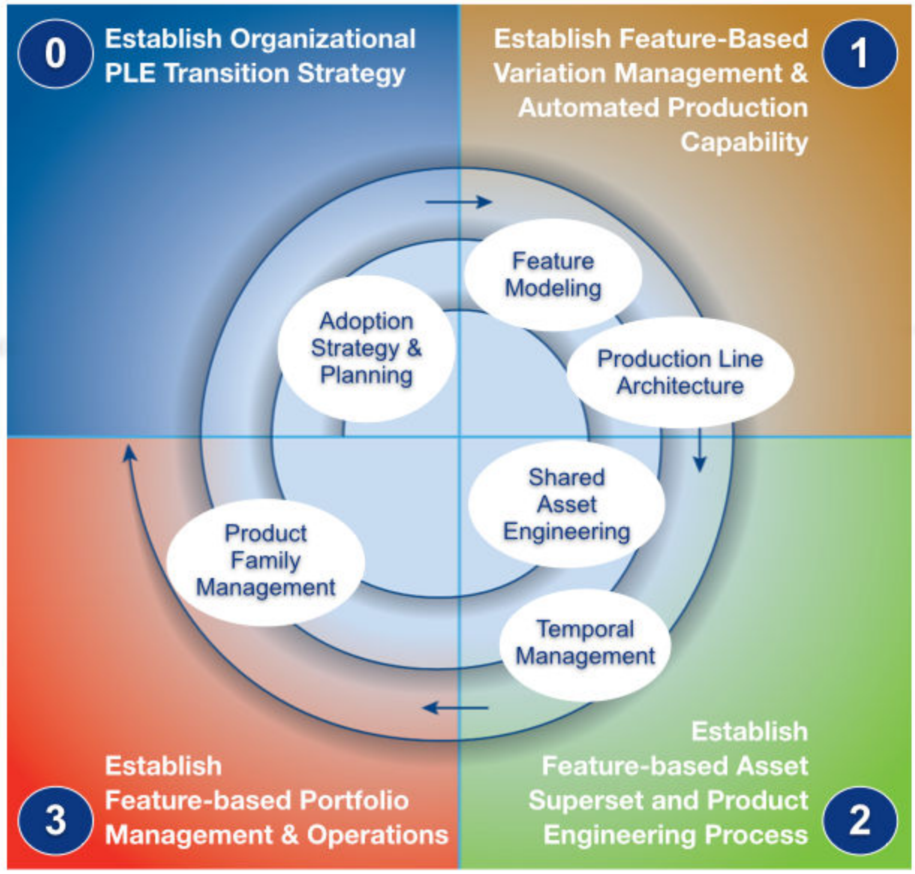

Figure 8. Spiral model for organizational PLE adoption

As you traverse each spiral, more and more PLE capability is rolled out across more and more of your organization, based on a plan and tempo tailored to your specific needs and situation. At each step, incremental adoption brings incremental benefit.

The spirals iteratively traverse four quadrants, described in detail below. Briefly, Quadrant 0 is for planning and monitoring the roll-out activities. The other three quadrants each focus on activities affiliated with one of the three tiers of the 3-Tiered PLE Methodology. A typical spiral will comprise activities from each quadrant, but some spirals (especially the early ones) may focus specifically on some but not all of the quadrants.

Quadrant 0. Establish Organizational PLE Transition Strategy

A successful transition to PLE requires thoughtful planning and commitment from the organization's technical and executive leadership. Activities in this quadrant engage those key decision-makers in the planning, guiding, and monitoring of the PLE transition and roll-out.

Focus areas in Quadrant 0 include:

- **PLE Adoption Strategy and Planning** to produce the PLE *concept of operations* and a detailed roll-out plan. A PLE ConOps is a description of the day-to-day operation of a fully functioning production line. It addresses the three dimensions of PLE (multi-product, multi-phase, and multi-baseline), to describe a holistic and pragmatic PLE solution. The ConOps, which captures the desired capability end state of the PLE organization, is then used to identify, assign, and schedule the incremental adoption steps necessary to move the organization to that state. The steps are assigned to spirals in the Spiral Model for PLE Adoption with specific tactical plans for the first few spirals that enable an incremental production deployment with minimally disruptive user adoption, big early wins, and zero risk of failure.

Quadrant 1. Establish Feature-Based Variation Management & Automated Production Capability

Activities in this quadrant focus on establishing the technical foundations for your production line: building feature models, feature profiles, and feature assertions; integrating lifecycle engineering artifacts as shared assets and designing how feature-based variation points will be implemented in shared assets; and installing and integrating any necessary tool bridges. Focus areas include:

- **PLE Feature Modeling** to build feature models that capture the important distinguishing characteristics of your products. Features provide a consistent, common language of abstraction across all phases of the engineering lifecycle. Feature profiles define products in terms of the features they possess. This activity also includes capturing feature assertions, which express the all-important constraints among the features, to preclude improper products from being built.
- **Production Line Architecture**, which is the overall partitioning and composition of a production line. The architecture defines a "product line of product lines" that mirrors the system-of-systems or subsystem structure of the existing applications' architecture. Work in this area identifies lower-level production lines that will be created and imported, and how many levels of production lines should constitute the hierarchy. Shared assets (such as requirements or code files) are attached to the appropriate production lines in the hierarchy, and an overall product family matrix is built.

The full PLE payoff for most organizations comes when shared assets are automatically configured to support any product in the product line and the organization's workflow, processes, and roles are optimally aligned with this capability.

Quadrant 2. Establish Feature-Based Asset Superset and Product Engineering Process

The full PLE payoff for most organizations comes when shared assets are automatically configured to support any product in the product line and the organization's workflow, processes, and roles are optimally aligned with this capability. Activities in this quadrant deal with transitioning engineers' roles and responsibilities to work in terms of the production line factory, and away from product-specific roles. Focus areas include:

- **PLE Shared Asset Engineering** to bring the product line's shared assets into the factory, by building in variation points defined in terms of the products' features. Asset engineering can be performed for all of the different types of shared assets you wish to incorporate into your PLE deployment – such as requirements, source code, test cases, user documentation, etc. The asset engineers (for example, requirements engineers) learn the variation point mechanisms available with Gears, configure assets with variation points to support the products in your product line, and explore techniques for transforming diverged or separate engineering artifacts for different products into a true shared asset for the product line using a consistent, cross-asset, feature-based variation point language. In this way, the asset engineers learn to shift their focus from product-specific assets to shared product line assets.
- **PLE Temporal Management**, which involves defining and adopting the practices necessary to manage the PLE factory over time. Shared assets and PLE models can evolve constantly to support new product releases, and it is essential to create and manage the temporal baselines needed for those releases. Temporal management addresses ensuring that the right versions of all of the shared assets will be available as needed to support upcoming product releases, and that past versions of any product can be re-built using the correct versions of the shared assets and PLE models (possibly across different physical asset repositories and CM systems).

Quadrant 3. Establish Feature-Based Portfolio Management Capability

As the production line is established, your organization's product line portfolio planners and managers can use it to define and deliver the products in your product line. This enables your organization to quickly bring new products or product features to market for targeting new segments, or gain competitive advantage in current market segments. Focus areas include:

- **Product Family Management**, which involves designing a product family tree based on the product offerings in your portfolio. Building a product family tree takes advantage of Gears' unique multistage configuration capability, which facilitates systematic construction of product sub-families by partially specifying feature selections at each level. Structuring your product line as a family tree is especially helpful for large portfolios with many offerings. It simplifies the task of product definition by calling out selection decisions common across an entire sub-family, and aligns the portfolio with multi-level organizations in which units at different levels are responsible for different sub-families.
- **Operations Planning,** which involves the activities necessary for the management of your products *after* they emerge from the PLE factory. These activities can include supply chain management, manufacturing, complexity management, logistics and distribution, portfolio-based sales, user training and support, service recalls and upgrades, in-field maintenance, consumer-selectable options management, and more. Treating these operations capabilities under the product line umbrella enables a consistent and robust operational capability across your entire portfolio.

The *technological* basis of modern PLE methods, such as the Gears PLE Lifecycle Framework, has come about based on years of experience in observing, capturing, and addressing the needs of product line organizations.

4. Conclusions

The *technological* basis of modern PLE methods, such as the Gears PLE Lifecycle Framework, has come about based on years of experience in observing, capturing, and addressing the needs of product line organizations. Based on many years of experience in helping organizations successfully adopt the product line engineering approach, we now follow suit with a proven *organizational* basis for PLE as well. We have identified a simple pattern for PLE methodology, referred to as the 3-Tiered PLE Methodology, and for PLE adoption, referred to as the Spiral Model for Organizational PLE Adoption.

These proven approaches are useful not only for the operation of PLE practice, but also in understanding, explaining and justifying the PLE approach, as well as for planning and making transitions into PLE practice.

5. References (TBD)

[1] Paul Clements and Linda Northrop. 2001. *Software Product Lines: Practice and Patterns*, AddisonWesley, Reading, MA.

[2] Davis Weiss and Chi Tau Robert Lai. 1999. *Software Product-line Engineering*. Addison-Wesley, Reading, MA, p 73-74.

[3] Atkinson, C., et.al. 2002. *Component-based Product Line Engineering with UML*. Pearson Education, London, UK.

[4] William A. Hetrick, Charles W. Krueger and Joseph G. Moore, *Incremental Return on Incremental Investment: Engenio's Transition to Software Product Line Practice*, October 2006, OOPSLA 2006, Portland, Oregon, ACM.

[5] Ross Buhrdorf, Dale Churchett, Charles Krueger. Salion's *Experience with a Reactive Software Product Line Approach*. 5th International Workshop on Product Family Engineering. Nov 2003. Siena, Italy. Springer-Verlag LNCS 3014, p 315.

[6] *New Methods Behind a New Generation of Software Product Line Successes*, Technical Report 200602261, BigLever Software.

[7] Charles W. Krueger. *New Methods in Software Product Line Practice*, in Communications of the ACM, December 2006, pages 37-40.

[8] Charles W. Krueger. *New Methods in Software Product Line Development*, in Proceedings of the 10th International Software Product Line Conference, Baltimore, MD. August 2006, pages 95-99.

[9] Krueger, C. *Easing the Transition to Software Mass Customization*. Proceedings of the 4th International Workshop on Product Family Engineering. October 2001. Bilbao, Spain. Springer-Verlag, New York, NY.

[10] Charles Krueger. *Variation Management for Software Production Lines*, in Proceedings of the 2nd International Software Product Line Conference, San Diego, California. August 2002, pages 37-48.

[11] Charles W. Krueger and Dale Churchett. *Eliciting Abstractions from a Software Product Line*, in Proceedings of the OOPSLA 2002 PLEES International Workshop on Product Line Engineering. Seattle, Washington. November 2002, pages 43-48.

[12] Clements, P. and Krueger, C., *Being Proactive Pays Off / Eliminating the Adoption Barrier*. IEEE Software, Special Issue of Software Product Lines. July/August 2002, pages 28-31.

[13] HomeAway: A Software Product Line Case Study, Technical Report 20070630, June 2007, BigLever Software, http://www.biglever.com/.

[14] Czarnecki, K. & Eisenecker, U. 2000. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, Boston, MA.

[15] Krueger, C. *Software Reuse*, in ACM Computing Surveys, June 1992, pages 131-183.

