White Paper

PTC®

Managing Product Variants in a Software Product Line with PTC Integrity

Software Product Line (SPL) engineering has become indispensable to many product engineering organizations. It enables those organizations to effectively manage the many product features and platform variations needed to remain competitive without stifling innovation or being affected by dramatically increasing costs of compliance.

At the same time, many product development organizations today are burdened with a wide array of disparate application lifecycle management (ALM) tools acquired over time and incrementally bundled together to automate the myriad of high-ceremony development processes. The resulting environment is one fraught with redundancy, inefficiency, error, escalating costs, compliance issues and lack of visibility into product release readiness.

In addition to the existing challenges, these organizations often find significant new challenges when implementing an SPL engineering practice in such an environment. Their legacy processes and tooling are immediately driven far beyond their original intent, stalling any SPL process implementation indefinitely.

This paper presents key aspects of successful lifecycle management processes and associated automation platforms that are critical to success when implementing SPL engineering practices. This paper also examines some proven patterns for management of product variants – a key aspect of SPL engineering – with PTC Integrity, a global software development product.

What Defines a "Good" Software Development Process?

A good software development process is lean. Every activity is purposeful and directly or indirectly contributes to tangible end results; that is, no process activity would be performed "just because the process says so." Hence, a particular process artifact should only be produced if it is used to help deliver the working software, maintain the software or help meet compliance.

For example, in a "high-ceremony" process, no change request can be implemented without creating a change package and no changes can be made to any lifecycle artifact without a corresponding change request.

In contrast, a less formal development process may not require a change package for certain types of change requests, and might allow certain artifact changes without an associated change request as long as a change package is created to track the changes.

Furthermore, a good process should lend itself to lifecycle automation. For example, meeting compliance regulations should be a natural byproduct of the automated development process, whereby artifacts for a compliance submission can be produced on-demand in an automated way – significantly reducing the time and effort needed to prepare for an audit.

While there are other important facets of a successful SPL engineering process, these two are fundamental regardless of the specific SPL method used.

Traditional ALM Solutions vs. ALM Optimized for Software-Intensive Product Development

For years, engineering organizations that develop software-intensive products have been forced to contend with one industry focused on automating traditional IT focused ALM and another, largely separate, industry focused on automating product lifecycle management (PLM). This situation has left a void for software development teams responsible for delivering ever-more complex software components that are embedded in engineered products. As the complexity and volume of software in engineered products have grown exponentially over the past few decades, the capabilities of traditional ALM tools to address the unique needs of software engineers working within a larger product engineering organization have fallen far behind. At the same time, leading PLM products have incrementally added capabilities to help manage software development activities and artifacts, but they have typically been significantly limited in scope and sophistication.

In particular, traditional ALM tools are unable to accommodate of the larger PLM process and automation platforms in which they must operate. Integration of a set of disparate ALM tools – which are themselves poorly integrated with one other – into a larger engineering automation ecosystem has been a daunting task largely left to the customer to plan, implement, and maintain. While PLM and traditional ALM vendors offer varying approaches to integrating the software engineering tool set into the system and hardware engineering environment, most attempts focus on aggregating huge numbers of software engineering artifacts into single line items in the product bill of materials (BOM) managed by the PLM tool. This approach, unfortunately, fails to recognize and manage the enormous complexity and ever-increasing number of software components in modern software-intensive products. A more effective approach is urgently needed to integrate software, hardware and system engineering artifacts and processes so as to meet the changing needs of today's engineering organizations.

PTC is addressing that need with PTC Integrity, a single product built from the ground up for global software development for engineered products. PTC Integrity was also built from the start for seamless integration into the larger PLM environment in a way that provides the required granular visibility and control of artifacts and processes between system, hardware, and software engineering.

Single Platform vs. ToolBox

PTC Integrity is the industry's only Global Software Development solution that offers out-of-the box capabilities to directly address customer challenges in product engineering. Many competing vendors offer toolboxes[1] containing collections of disparate tools to provide coverage of the development lifecycle. These tools have typically been acquired from multiple sources over time, with a few having been developed in-house. Thus, the individual tools tend to depend on widely varying implementation technologies and architectural paradigms, and have different user-experience models. The tools typically cover the lifecycle with gaps and overlaps in their collective capabilities. This toolbox approach does not allow the creation of effective lifecycle automation solutions without the organization investing in additional staff-months to bundle, customize and configure those tools to meet the organization's needs.

1 The toolbox often contains disparate tools developed by different vendors with varying point-to-point integrations. Regardless of how well-integrated these disparate tools are, the client must still invest significantly to create desired solutions from the toolbox.

With PTC Integrity, customers can begin realizing a return on their investment immediately, growing into a comprehensive solution that meets the organization's needs in an incremental fashion. PTC Integrity also allows the organization to maintain tight connections with existing practitioner and workgroup ALM tools, providing a single source of truth that gives the organization real-time visibility into product release readiness – as well as an incremental path to deployment that does not require a disruptive "rip-and-replace" implementation.

Effective Management of Product Variants

To show how PTC Integrity – as a single product – automates global software development processes, this paper will consider the problem of managing product variants, a key aspect of an SPL engineering practice. Product variants, whether differing by function or parameter values that drive functional behavior, have a significant amount of commonality that must be leveraged to lessen complexity across variants. Without effectively exploiting that commonality, duplication will lead to exponential growth in the number of artifacts that need to be managed for each variant – thereby multiplying the resources and drastically increasing the cost to develop and maintain each variant. This duplication approach, sometimes described as "clone-and-own," also leads to loss of traceability and other relationships between "cloned" artifacts across the variants. Once cloned, artifacts being reused among multiple variants take on a life of their own, leaving teams to develop and maintain each variant as they would a stand-alone product.

To be successful in SPL engineering, the commonality among product variants must be maintained and propagated in a controlled manner across conforming product families. PTC Integrity excels at enabling organizations to "tame" the complexity of managing holistic, shared artifact sets, or assets, to create deliverable product variants. At the core of PTC Integrity is the notion of a release of related functionality and management of both variant and core artifacts, as well as planned propagation and impact analysis. PTC Integrity provides support for both functional variation as well as data-driven software variation, the latter being highly desirable – particularly in control-oriented systems – to establish a common code base, with the majority of the variance in functional behavior parameterized based on input data values.

Managing Shared Assets for Product Variants with PTC Integrity

While managing the development of product variants requires supporting shared assets across the lifecycle (e.g., requirements, designs, models, code and test), this paper will illustrate the capabilities of PTC Integrity in this respect by focusing on how it tackles the problem of managing the requirements for product variants.

Requirements of Functional Variants[2]

A functional product variant is one that shares myriad common requirements with other variants of the same product, but also adds requirements that are specific to its own structure and behavior.

First, a set of base requirements are written that will be common to the functional variants of the product.[3] The set of requirements for each functional variant will then comprise some or all of the common-base requirements, plus a set of requirements that are specific to that variant.

The following scenarios show how this approach provides maximum efficiency and accuracy in creating, maintaining and certifying functional variants.[4]

2 Please note that what follows is only one set of specific scenarios for using PTC Integrity to manage functional and parameter-driven variant requirements. Such scenarios can be varied to more closely meet user needs.

3 For simplicity, the creation of the common base requirements is presented here as preceding the creation of any product variant requirements. More realistically, however, the requirements for the common base may emerge over time after creating the requirements for two or more variants. In any event, the main characteristics of the scenarios presented here would not change.

4 The scenarios in this paper do not address more complex situations such as multiple layers of commonality. These more complex scenarios can also benefit from a common base, al eit in a more involved manner.

New Variant

- Create a blank new document for the new variant – call this the "parent" document. See Figure 1;
- Enter two sets of requirements in this document. For the first set, point to all, or a subset, of the common set of requirements. This set cannot be edited in the parent document, which was created in the previous step;
- For the second set of requirements in the parent document, enter the requirements that are specific to only that variant; and
- The ability to have each product variant point to the same physical common base enables avoidance of "clone-and-own" and maintains the key relationships between the common shared requirements across multiple variants.

Certification/Recertification

- Once the common base is certified, it need not be recertified every time a new variant points to it;
- Only the requirements unique to the variant (i.e., those contained in the parent document mentioned in the "New Variant" scenario above) need to be certified;
- When changes are made to the common base (see the "Change Request" scenario below), recertification will need to be performed for all product variants pointing to the changed requirements in the common base. However, the effort needed to determine which variants are affected and how – as well as which downstream shared development assets may also need to be recertified for each variant – is straightforward and precise, since PTC Integrity maintains the appropriate relationships between shared assets across the product line;
- Ultimately, the time/cost of producing a compliance submission is dramatically reduced.

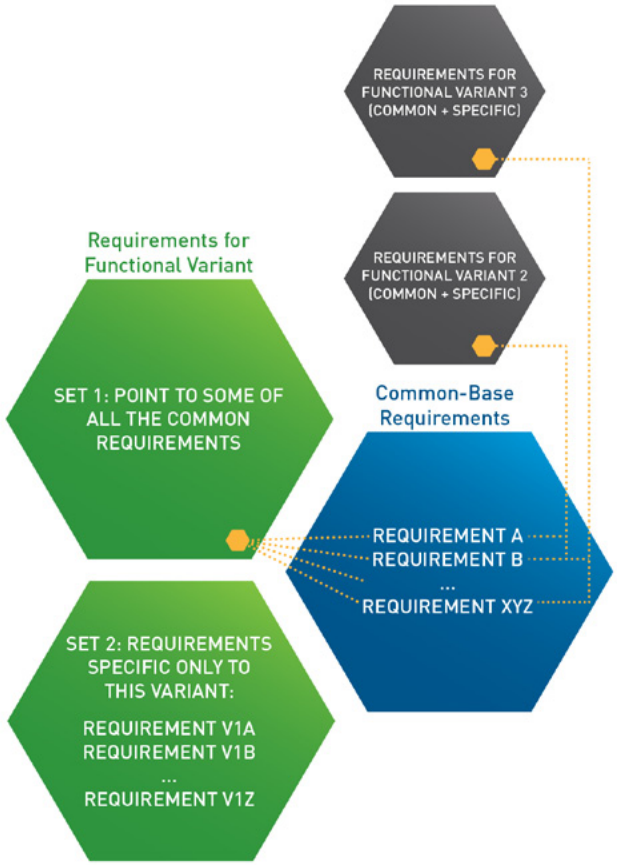


Figure 1: Several functional product variants leveraging common base requirements. In this example, Variant 1 includes its own specific requirements, plus all the requirements in the common base, whereas each of Variants 2 and 3 includes its respective specific requirements plus only a subset of the common base.

Change Request

- Significant savings in cost, time and quality are realized if the change request targets something in the common requirements;
- The change is made only once in the common base and saved in a new version;
- Future variants can point to this new version (see Figure 2 below), but current variants will not automatically change to point to the new version until it is determined that a given variant should inherit the change. This preserves the certification that was conducted for each of the current variants (i.e., original common base certified once + variant-specific requirements certified per variant).

Requirements of Parameter-Driven Variants[5]

A parameter-driven product variant represents a common template of parameterized requirements, assigning its specific values to some or all of the parameters in the common template. This common base template contains parameters whose values can be changed per product variant. Such parameterized requirements can be highly desirable in control-oriented systems to establish a common code base that is parameterized based on input data values.

5 In the example scenarios shown here, although functional variants are treated separately from parameter-driven variants, the two types of variants can be combined in any given development environment.

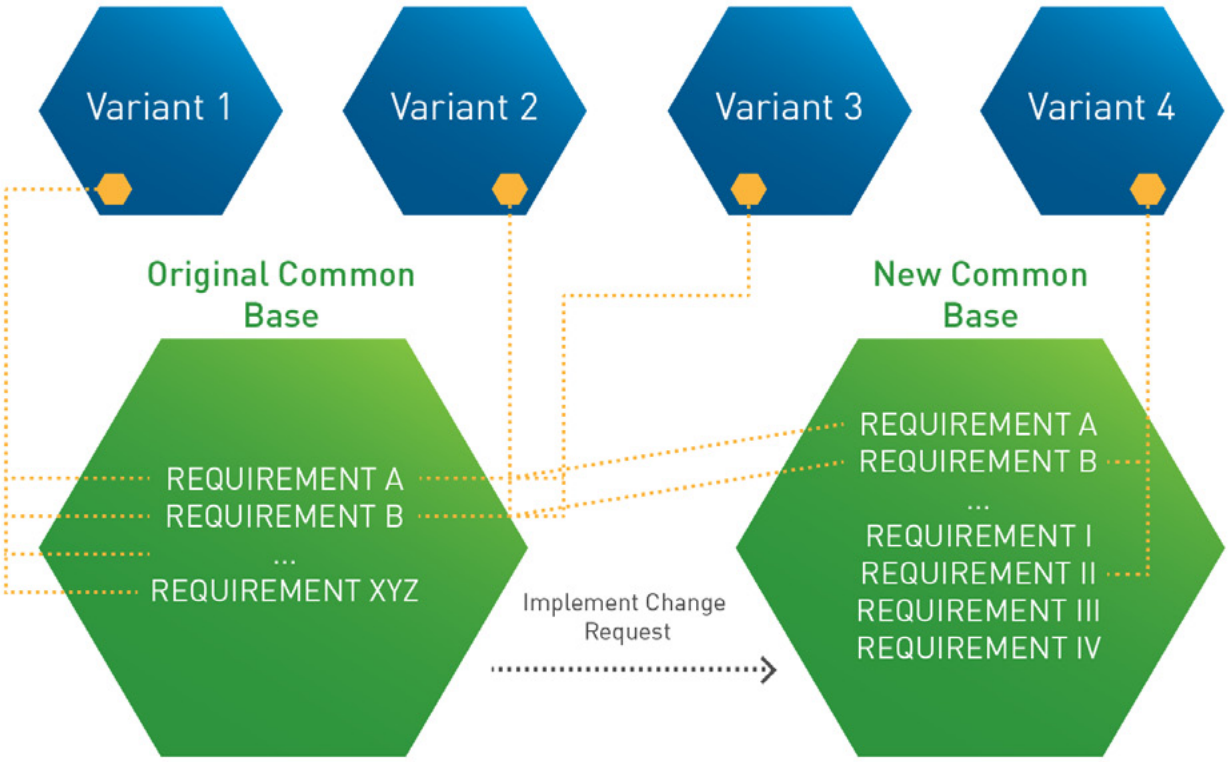


Figure 2: When a new version of the common base is produced, new variants may point to its requirements, but existing variants can keep pointing to the original version of the common base to preserve their certification until it is determined that a given variant is ready to inherit the change. As an architectural point, requirements that don't change in the new common base are just pointers to the same requirements in the original common base.

For example, a requirement in the common base may be "The maximum allowable temperature shall be {{TempMax}} degrees". (See Figure 3 below.) The variable {{TempMax}} can be set to different values per product variant. PTC Integrity allows each product variant to point to the same physical common base while modifying as many parameters as necessary for the product variant.

The following scenarios show how this approach provides maximum efficiency and accuracy in creating, maintaining and certifying parameter-driven variants.[6]

New Variant

- Create a blank new document for the new variant – call this the "parent" document;
- Point that document to the common set of requirements, which contains a base set of already certified parameter values;
- Substitute only the parameters that need to change for the new variant in the common base set. You'll immediately see the result of the new substitutions in the context of common requirements immediately. Such immediate, visual feedback allows the user to ensure accuracy in the requirements – thereby achieving requirement correctness by "construction" rather than just by "inspection" in later stages of development, when it becomes more expensive to fix requirement errors;
- Being able to have each product variant point to the same common base precludes the occurrence of "clone-and-own," which would have resulted in a new physical copy of the set of common requirements for each variant.

6 The scenarios in this paper do not address more complex situations such as multiple layers of commonality. These more complex scenarios can also benefit from a common base, albeit in a more involved manner.

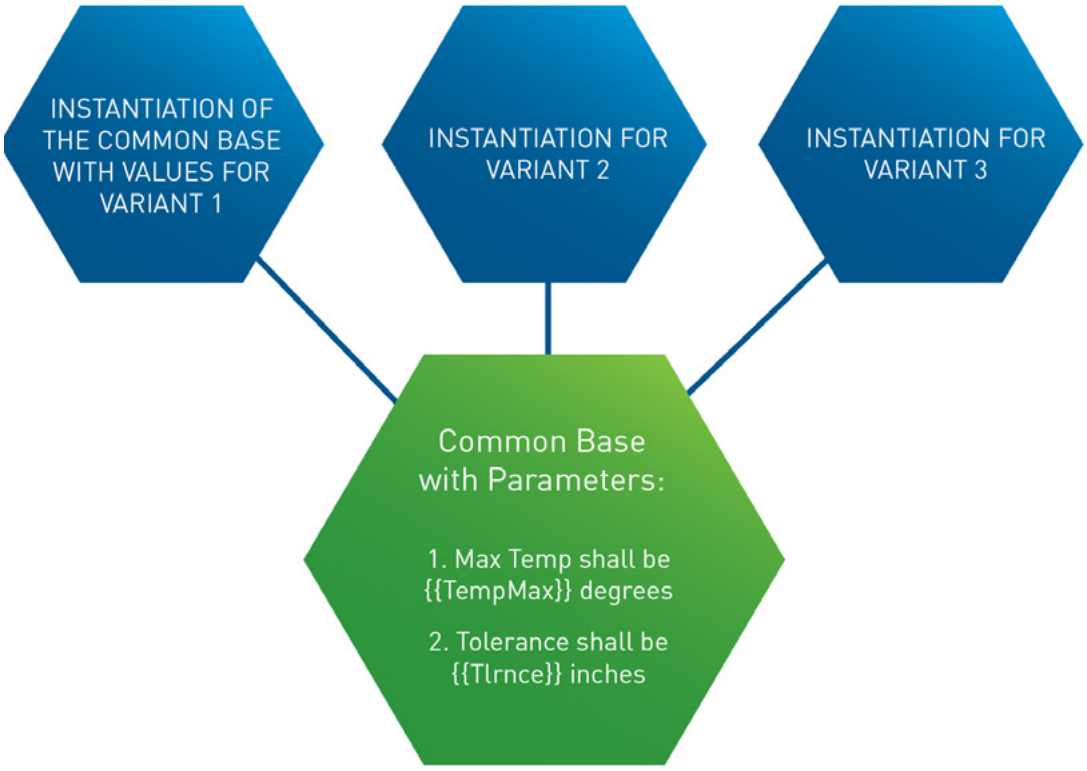


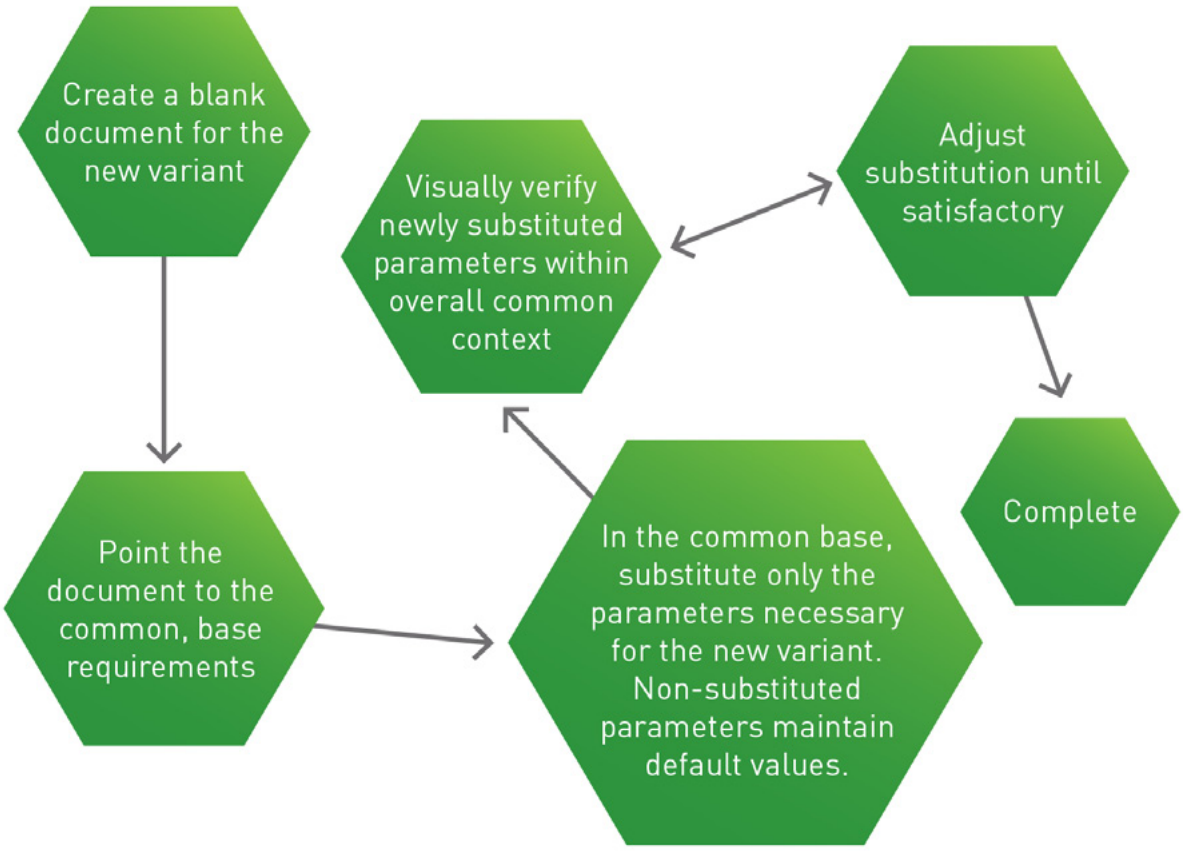
Figure 3: Several parameter-driven variants can represent the same physical parameterized common base

Certification/Recertification

- Once the common base is certified, it need not be recertified every time a new variant points to it;
- Only the newly represented values that are unique to the variant need to be certified;
- When changes are made to the common base (see the "Change Request" scenario below), recertification will need to be performed for all product variants pointing to the changed requirements in the common base. However, the effort needed to determine which variants are affected and how – as well as which downstream shared development assets may also need to be recertified for each variant – is straightforward and precise, since PTC Integrity maintains the appropriate relationships between shared assets across the product line;
- Ultimately, the time/cost of producing a compliance submission is dramatically reduced.

Change Request

- Significant savings in cost, time and quality occur if the change request targets something in the common requirements;
- The change is made only once in the common base and saved in a new version;
- Future variants can point to this new version (See Figure 2 on page 5), but current variants will not automatically change to point to the new version until it is determined that a given variant should inherit the change. This preserves the certification that was conducted for each of the current variants (i.e.,original common base certified once + variant-specific requirements certified per variant).



Conclusion

PTC Integrity allows organizations implementing SPL engineering practices to manage the complexities of product variants effectively and efficiently. PTC Integrity maintains a single physical copy of the common shared assets across a software product line. As such, PTC Integrity handles common requirements and other lifecycle assets much more efficiently than a traditional approach, which can dramatically increase the number of requirements – inviting error-prone duplication and dramatically increased cost. PTC Integrity also maintains the traceability relationships between common, shared assets and variant-specific assets, providing teams a clear understanding of the relationships between specific variants and the shared core assets. The advanced variant management capabilities of PTC Integrity enable organizations to effectively implement SPL engineering without losing control of shared and variant-specific assets across the lifecycle.

PTC Integrity Business Unit Locations

North America
1 800 613 7535

United Kingdom
+44 (0) 1252 453 400

Germany
+49 (0) 711 3517 750

Asia Pacific
+65 6830 8338

Japan
+81 3 5422 9503

For more information visit: PTC.com/product/integrity.

7511–ManagingProductVariants-WP–EN–0912