Integrity を使用したソフトウェア製品ラインの製品バリエーション管理

要約

ソフトウェア製品ライン (SPL) エンジニアリングは、多くの製品エンジニアリング企業にとって不可欠な取り組みとなっています。SPL エンジニアリングを活用すれば、イノベーションを犠牲にすることやコンプライアンス対応に忙殺されることなく、多数の製品機能とプラットフォーム バリエーションを効率的に管理し、競争力を維持することができます。また現代の製品開発企業では、厳格なソフトウェア開発プロセスを自動化するために導入され、寄せ集められた多種多様なアプリケーション ライフサイクル管理 (ALM) ツールの存在も問題化しています。これらのツールは、冗長性や非効率性の増大、エラーやコストの増加、コンプライアンス対応の複雑化、製品のリリース準備状況に対する可視性の欠如といった問題の原因になっています。

このような環境に SPL エンジニアリングを導入しようとすると、既存の課題に加え、新たな問題が浮上してきます。その中でも特に挙げられることが多いのは、ライフサイクル管理のプロセスと自動化に関する問題です。SPL への移行では、従来のプロセス管理ツールや自動化ツールでは元々想定されていなかった機能が必要とされ、既存環境の大幅な再構築や調整、合理化を強いられるためです。

この技術白書では、SPL エンジニアリングの導入を成功に導くために必要となるライフサイクル管理プロセスと、それを支える自動化プラットフォームについて解説します。また、SPL エンジニアリングの鍵となる製品バリエーション管理のベスト プラクティスを、PTC のソフトウェア システム ライフサイクル管理製品である Integrity で実現する方法についても見ていきます。

「優れた」ソフトウェア開発プロセスの定義

優れたソフトウェア開発プロセスとは、無駄のないプロセスのことを言います。開発プロセスを構成する各アクティビティは、それぞれ確固たる目的を持ち、具体的な結果に直接的、間接的に貢献する必要があります。「プロセスでそのように定義されているから」という理由だけで、アクティビティが実施されることがあってはなりません。つまり、プロセスで産み出される成果物は、必ず完成後のソフトウェアの一部として使用されるか、またはソフトウェアを維持管理するため、もしくはコンプライアンス対応のために使用される必要があります。

たとえば、厳格な開発プロセスでは、変更リクエストの実装には変更パッケージを作成する必要があり、ライフサイクルの成果物の変更には対応する変更リクエストが必要とされます。対照的に、より決まり事の少ない開発プロセスでは、変更リクエストの実装に必ずしも変更パッケージは必要なく、また変更内容を追跡する変更パッケージを作成するかぎり、成果物の変更に必ずしも変更リクエストは必要ありません。

また、優れたソフトウェア開発プロセスは、ライフサイクルの自動化にも貢献します。たとえば、開発プロセスを自動化した副産物として、コンプライアンス規制への対応を自然に達成できるようになっています。自動化したプロセスの中で、コンプライアンス報告に必要な資料をオンデマンドで作成できるため、監査の準備に要する時間と労力を大幅に削減できます。

SPL エンジニアリング プロセスの成否を左右する要素はほかにもありますが、少なくとも上記の 2 点は、採用する SPL の手法を問わずに求められる最も重要なポイントとなります。

ソフトウェア システム ライフサイクル管理とアプリケーション ライフサイクル管理

ソフトウェアを駆使した製品を開発するエンジニアリング企業は長年、ALM の自動化に取り組むベンダーと、製品ライフサイクル管理 (PLM) の自動化に取り組むベンダーのどちらの製品を選択するかに悩まされてきました。今日、エンジニアリング製品に組み込まれるソフトウェア部品は複雑化の一途をたどっており、このような部品の開発を担当するソフトウェア開発部門は、ALM と PLM の狭間で適切なツールを見つけることができずに苦労しています。組み込みソフトウェアの点数とその複雑さは過去数十年間で大幅に増大していますが、ALM ツールは大規模な製品エンジニアリング企業のソフトウェア エンジニア固有のニーズにまったく応えることができていません。一方の PLM でも、主要製品はソフトウェア開発作業やその成果物を管理する機能を徐々に追加していったものの、多くの場合、その範囲や水準は十分なものではありませんでした。

特に ALM ツールでは、PLM プロセスや自動化プラットフォームにうまく組み込むことができないという点が大きな問題になっています。相互の統合も不十分な一連の ALM ツールをエンジニアリング自動化エコシステムに統合することは容易ではありません。しかもその作業の多くは、顧客企業自身が計画、実装、管理する必要があります。PLM ベンダーと ALM ベンダーは、それぞれ異なるアプローチで、ソフトウェア エンジニアリング ツールをシステムおよびハードウェア エンジニアリング環境に統合する取り組みを進めています。しかしそのほとんどは、ソフトウェア エンジニアリングに関する多数の成果物を、PLM ツールで管理される製品部品表 (BOM) の 1 つの項目へ集約することに終始しています。残念ながらこのアプローチでは、点数の増加と複雑化が進むソフトウェア部品を効果的に把握、管理することはできません。そのため今日では、ソフトウェア、ハードウェア、およびシステムのエンジニアリング成果物とプロセスを統合して、多様化するエンジニアリング企業のニーズに応えられるより効果的なアプローチが強く求められています。

PTC の Integrity は、このような課題を解決するために専用開発されたソフトウェア システム ライフサイクル管理 (SSLM) 製品です。PLM 環境にシームレスに統合できるよう配慮して設計されているため、システム、ハードウェア、ソフトウェアの各エンジニアリング領域で発生する成果物やプロセスを詳細に可視化し、きめ細かく管理できます。

単一のプラットフォームとツールボックス型の製品

Integrity は、複雑な設定や拡張機能の追加なしで、製品エンジニアリング ライフサイクル管理の課題に直接対処できる業界唯一の SSLM 製品です。多くの競合ベンダーは、開発ライフサイクル全体をカバーするために多種多様なツールで構成されたツールボックス型[1] の製品を提供しています。この種の製品に含まれるツールの多くは、企業買収を通じて取得されたものであり、そのベンダー自身が自社開発したものは一部に過ぎません。そのため、個々のツールはまったく異なる実装技術やアーキテクチャ パラダイムに基づいて開発されていることが多く、操作体系が統一されていないこともしばしばあります。また、開発ライフサイクルをカバーする各機能にもギャップや重複が見られます。この種の製品では、数人月をかけて各ツールを統合、カスタマイズ、設定をしなければ、自社のニーズを満たす効果的なライフサイクル自動化ソリューションを構築することは困難です。一方の Integrity では、複雑な作業なしですぐに運用を開始し、段階を踏んで自社のニーズを満たす包括的なソリューションに拡張していくことができます。また、個人やワークグループ向けの既存の ALM ツールとも緊密に連携できるため、唯一の正しい情報源を実現して製品のリリース準備状況をリアルタイムに把握できるほか、業務プロセスの一時停止を伴うシステムの「総入れ替え」なしで段階的に導入を行えます。

1 一般にツールボックス型の製品は、異なるベンダーによって開発され、さまざまな手法によって 1 対 1 で統合されたツール群から構成されています。そのため各ツールがうまく統合されていたとしても、自社の環境に最適なソリューションを実現するには多大な投資を強いられます。

製品バリエーションの効果的な管理

Integrity が単独の製品としてどのように SSLM プロセスを自動化するのかを示すため、本書では、SPL エンジニアリングにおける重要問題である製品バリエーション管理に着目します。製品バリエーションには、機能面での違いによるものと、機能の振る舞いを左右するパラメータ値の違いによるものとがありますが、どちらのタイプにもかなりの割合で共通部分が存在します。バリエーション管理の複雑さを軽減するには、この共通部分をうまく活用することが欠かせません。それができない場合、共通部分の重複によって、バリエーションごとに管理する必要のある成果物が爆発的に増加し、各バリエーションの開発と管理に要する人材やコストも大幅に増大することになります。「クローンと本体 (clone-and-own)」とも呼ばれるこの状態では、クローン化された成果物間のトレーサビリティやその他の関係性も失われます。クローン化され、複数のバリエーションで再利用される成果物はそれぞれ独立した存在となり、各バリエーションはスタンドアロンの製品として開発、管理されます。

SPL エンジニアリングを成功に導くには、製品バリエーションの共通部分を適切に管理し、共通性のある各製品ファミリーに確実に伝播させる必要があります。Integrity では、共有された成果物全体 (資産) の管理に伴う複雑さを軽減し、無駄なく製品バリエーションを開発することができます。Integrity の主眼は、関連機能のリリース管理、バリエーションおよび主要成果物の管理、計画的な伝播、影響分析を支援することにあります。Integrity は、機能バリエーションとデータ制御に基づくソフトウェア バリエーションの両方をサポートしていますが、バリエーション管理においてより望ましいのは後者です。共通基盤における機能の違いの大半を入力データ値に基づいて設定できるため、特に制御指向のシステムにおいて高い利便性が得られます。

Integrity を使用した製品バリエーションの共有資産管理

製品バリエーション開発の管理では、ライフサイクル全体にわたって要件や設計、モデル、コード、テストなどの共有資産をサポートする必要がありますが、本書では、製品バリエーションの要件管理に焦点を当てながら Integrity の機能を紹介していきます。

機能バリエーションの要件[2]

製品の機能バリエーションでは、同じ製品の別のバリエーションとかなりの数の要件を共有することになります。その一方、各バリエーションの構造や振る舞いに固有の要件も発生します。

機能バリエーションでまず主体となるのは、製品の各バリエーションに共通する一連の基本要件です。[3] 各機能バリエーションの要件は、この共通基盤に含まれる要件の一部またはすべてと、各バリエーションに固有の要件で構成されます。

以降の各シナリオでは、機能バリエーションを開発、管理、認証するにあたり、このアプローチで最大限の効率性と正確性が得られる理由を示します。[4]

新しいバリエーション

a. 新しいバリエーションのための空の新規ドキュメントを作成します。これを「親」ドキュメントと呼びます (図 1 を参照)。

b. このドキュメントで 2 セットの要件を定義します。1 セット目では、共通する要件のすべてまたは一部への参照を記述します。1 セット目の要件は、手順 a で作成した親ドキュメントでは編集できません。

c. 2 セット目では、このバリエーションに固有の要件を定義します。

d. 物理的な共通基盤を各製品バリエーションからの参照にしているのは、「クローンと本体」になってしまうことを回避するとともに、共通要件の間の主要な関係を複数のバリエーションにわたって保持可能にするためです。

認証と再認証

a. 一度認証した共通基盤は、新しいバリエーションから参照するたびに再認証する必要はありません。

b. 認証が必要なのは、バリエーションに固有の要件 (「新しいバリエーション」の項の親ドキュメントで定義した要件など) のみです。

c. 共通基盤に変更を加えた場合は (次の「変更リクエスト」の項を参照)、その共通基盤の変更された要件を参照するすべての製品バリエーションについて再認証が必要となります。ただし Integrity では、各共有資産の関係が製品ライン全体にわたって厳格に管理されているため、どのバリエーションがどのような影響を受けるかを把握したり、再認証が必要になる各バリエーションの下流の共有資産を確認したりすることは容易かつ正確に行えます。

d. 以上により、コンプライアンス報告書の作成にかかる時間とコストが大幅に削減されます。

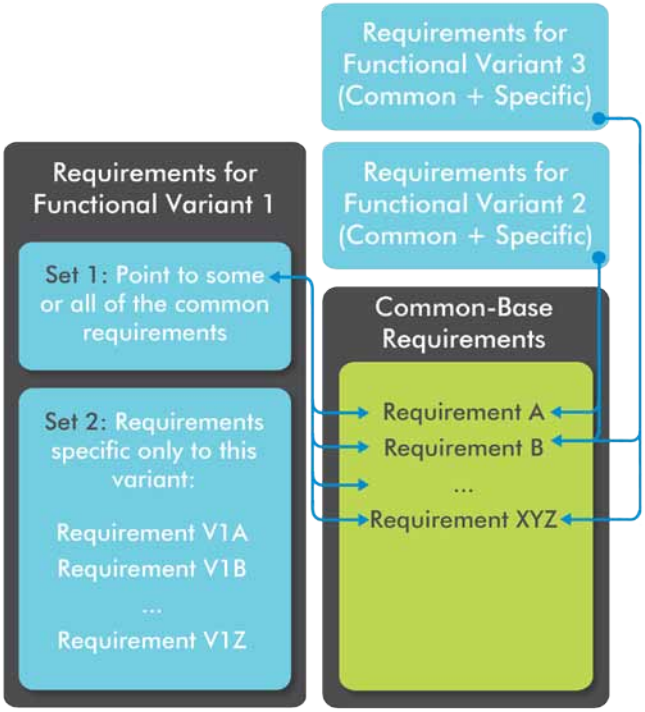


図 1：複数の製品バリエーションが共通基盤の要件を共有しています。図のバリエーション 1 の要件は、固有の要件と共通基盤のすべての要件で構成されています。一方、バリエーション 2 と 3 の要件は、それぞれに固有の要件と共通基盤の一部の要件で構成されています。

2 以降で説明するのは、Integrity を使用して機能バリエーションとパラメータ制御のバリエーションの要件を管理するシナリオ例です。実際のシナリオは、環境のニーズによって大きく異なる場合があります。
3 ここでは、わかりやすくするため、製品バリエーションの要件の前に共通基盤の要件を策定しています。しかし通常、共通基盤の要件は、複数のバリエーションの要件を策定した後、時間をかけて策定されます。この点は現実と異なりますが、シナリオの本質に影響はありません。
4 本書のシナリオでは、共通部分が階層化しているなどの複雑な問題は扱っていませんが、Integrity の共通基盤機能は、多少手間は増えるものの、より複雑なシナリオにも対応することができます。

変更リクエスト

a. コストおよび時間の削減と品質向上の面で大きな効果が期待できるのは、変更リクエストの対象が共通要件の一部である場合です。

b. 共通基盤を変更する場合、その変更を行う必要があるのは 1 回のみです。この変更は共通基盤の新しいバージョンに保存されます。

c. これ以降に作成するバリエーションはこの新しいバージョンを参照することができますが (下の図 2 を参照)、既存のバリエーションは自動的に新しいバージョンを参照するようにはなりません。新しいバージョンを参照するには、既存のバリエーションでその変更内容を継承するように指定する必要があります。これにより、既存の各バリエーションで行った認証 (元の共通基盤に対して一度行われた認証とバリエーションごとに行われたバリエーション固有の要件に対する認証) が引き継がれます。

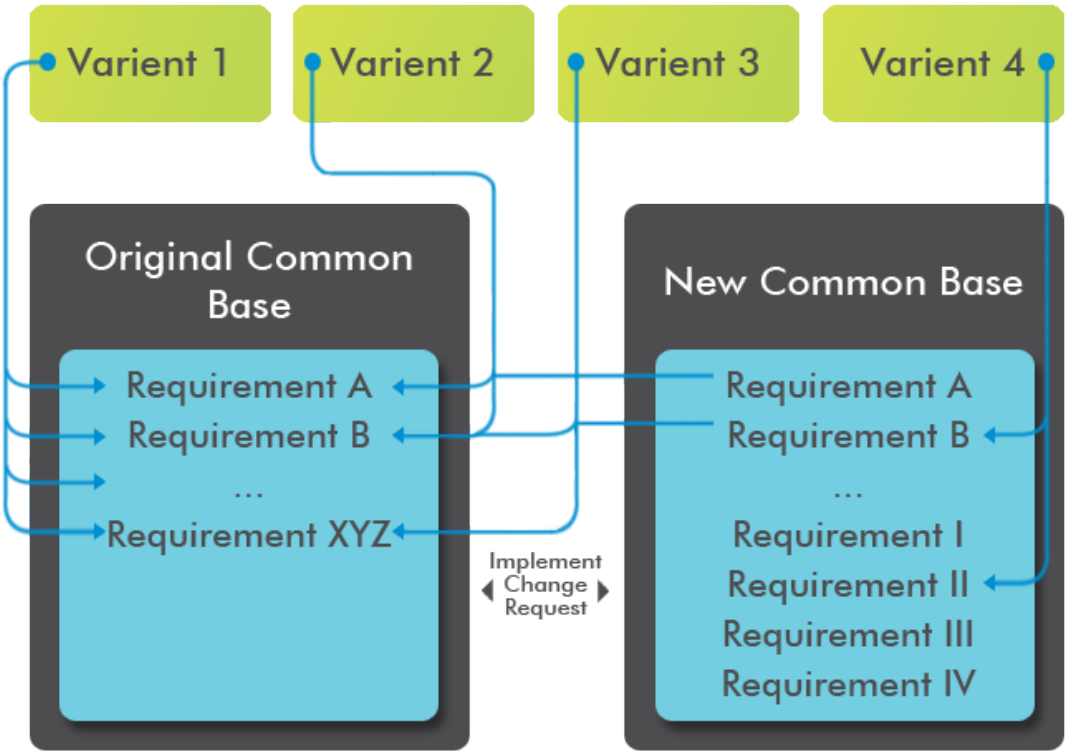


図 2：共通基盤の新しいバージョンが作成されると、新しいバリエーションはこの共通基盤の要件を参照します。ただし既存のバリエーションは、変更内容を継承するよう指定されるまで、認証を維持するために元のバージョンの共通基盤を参照することができます。新しい共通基盤で内容が変更されていない要件は、元の共通基盤の同じ要件を参照します。

パラメータ制御のバリエーションの要件[5]

パラメータ制御の製品バリエーションは、パラメータ化された要件の共通テンプレートを表すものであり、テンプレート内の一部またはすべてのパラメータに特定の値を割り当てます。パラメータには、製品バリエーションごとに異なる値を割り当てることができます。要件がパラメータ化されていると、入力データ値に基づいて共通基盤のパラメータを設定できるため、特に制御指向のシステムにおいて高い利便性が得られます。

たとえば、共通基盤に「最高温度は {{TempMax}} 度とする」という要件があるとします (図 3 を参照)。変数 {{TempMax}} には、製品バリエーションごとに異なる値を設定できます。Integrity では、各製品バリエーションで物理的な共通基盤を参照しながら、必要に応じて製品バリエーションごとに異なるパラメータ値を割り当てることができます。

以降の各シナリオでは、パラメータ制御のバリエーションを開発、管理、認証するにあたり、このアプローチで最大限の効率性と正確性が得られる理由を示します。[6]

5 本書では、機能バリエーションとパラメータ制御のバリエーションでシナリオを分けていますが、実際の開発環境では両者が混在することもあります。
6 本書のシナリオでは、共通部分が階層化しているなどの複雑な問題は扱っていませんが、Integrity の共通基盤機能は、多少手間は増えるものの、より複雑なシナリオにも対応することができます。

新しいバリエーション

a. 新しいバリエーションのための空の新規ドキュメントを作成します。これを「親」ドキュメントと呼びます

b. このドキュメントから、認証済みの一連の基本パラメータ値が設定された共通要件を参照します。

c. 共通基盤のうち、新しいバリエーション用に変更する必要のあるパラメータ値のみを差し替えます。すると、共通要件に新しいパラメータ値を適用した結果をすぐに確認できます。このように、結果をすばやく視覚的に検証できるため、要件エラーの訂正が高コストになる下流工程での「検査時」ではなく、「要件の設定時」に要件の正しさを確認できます。

d. 各製品バリエーションで共通基盤を参照できるため、共通要件の物理的なコピーがバリエーションごとに作成されてしまう「クローンと本体」の発生を回避できます。

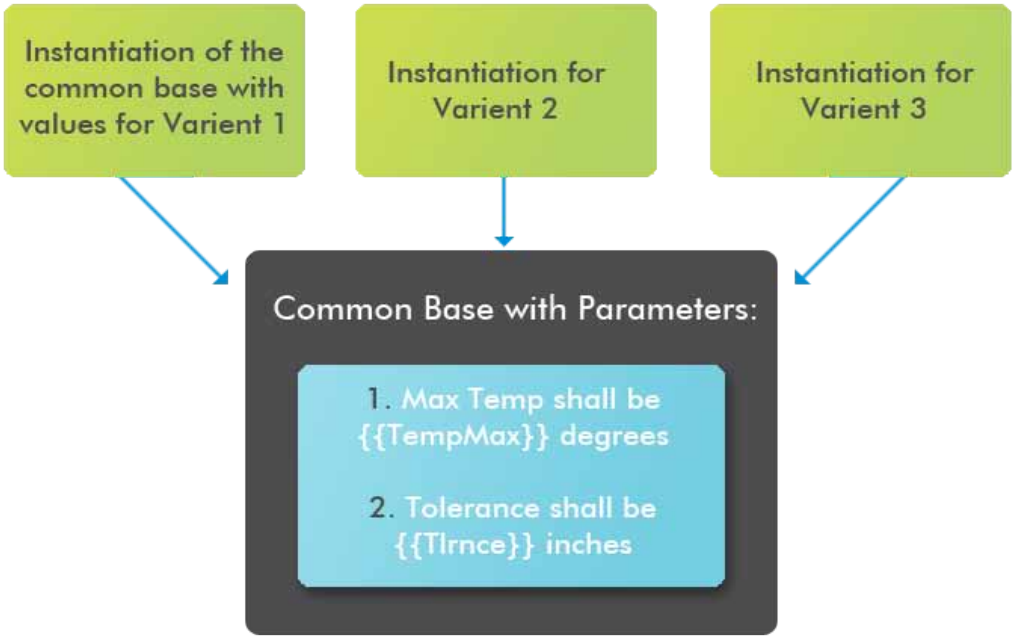


図 3 : パラメータ制御の複数のバリエーションによって、パラメータ化された物理的な共通基盤が表されます。

認証と再認証

a. 一度認証した共通基盤は、新しいバリエーションから参照するたびに再認証する必要はありません。

b. 認証が必要なのは、バリエーションに固有の新しい値のみです。

c. 共通基盤に変更を加えた場合は (次の「変更リクエスト」の項を参照)、その共通基盤の変更された要件を参照するすべての製品バリエーションについて再認証が必要となります。ただし Integrity では、各共有資産の関係が製品ライン全体にわたって厳格に管理されているため、どのバリエーションがどのような影響を受けるかを把握したり、再認証が必要になる各バリエーションの下流の共有資産を確認したりすることは容易かつ正確に行えます。

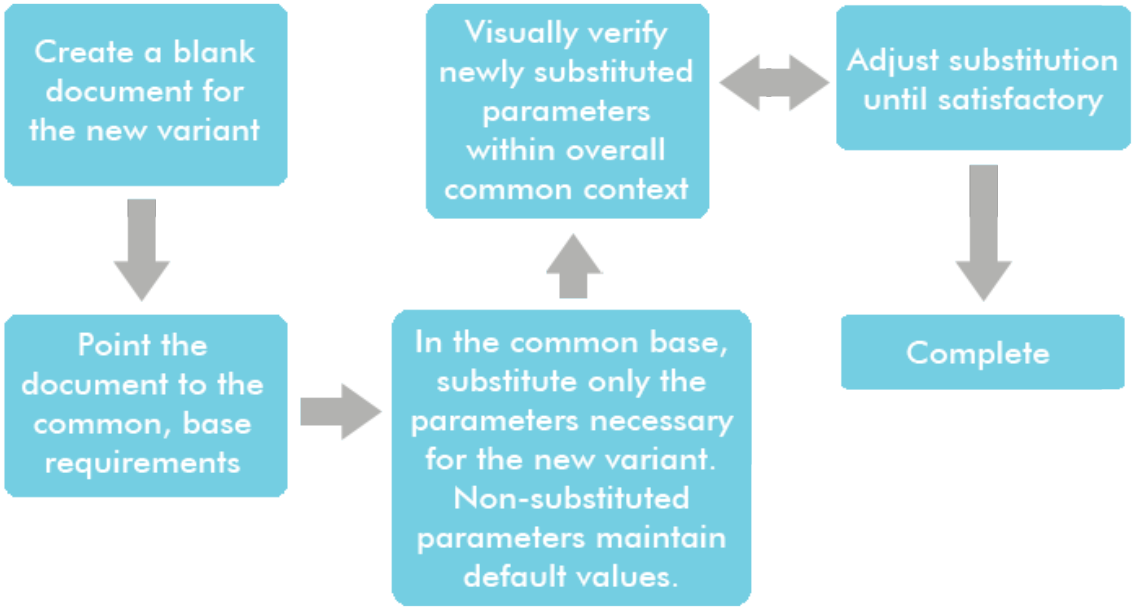
d. 以上により、コンプライアンス報告書の作成にかかる時間とコストが大幅に削減されます。

変更リクエスト

a. コストおよび時間の削減と品質向上の面で大きな効果が期待できるのは、変更リクエストの対象が共通要件の一部である場合です。

b. 共通基盤を変更する場合、その変更を行う必要があるのは 1 回のみです。この変更は共通基盤の新しいバージョンに保存されます。

c. これ以降に作成するバリエーションはこの新しいバージョンを参照することができますが (上の図 2 を参照)、既存のバリエーションは自動的に新しいバージョンを参照するようにはなりません。新しいバージョンを参照するには、既存のバリエーションでその変更内容を継承するように指定する必要があります。これにより、既存の各バリエーションで行った認証 (元の共通基盤に対して一度行われた認証とバリエーションごとに行われたバリエーション固有の要件に対する認証) が引き継がれます。



結論

SPL エンジニアリングの導入を検討している企業では、Integrity を利用することで、複雑な製品バリエーションを効果的、効率的に管理することができます。Integrity では、あるソフトウェア製品ラインの共有資産を単一の物理的なコピーとして維持管理するため、製品ラインに共通の要件やその他のライフサイクル資産を非常に効率よく扱うことができます。これに対し従来型のアプローチでは、管理すべき要件の数が大幅に増加して、重複に起因するエラーが生じたり管理コストが増大したりする可能性があります。また Integrity では、共有資産とバリエーション固有の資産の間の関係性を追跡できるトレーサビリティ機能を備えているため、両者の関係を明確に把握できます。これら先進的なバリエーション管理機能により、共有資産とバリエーション固有の資産をライフサイクル全体にわたって適切に管理しながら、SPL エンジニアリングを効果的に導入することが可能となります。

Integrity 事業所

北米 +1 800 613 7535

英国 +44 (0) 1252 453 400

ドイツ +49 (0) 711 3517 750

アジア太平洋地域 +65 6830 8338

日本 +81 3 5422 9503

integrityinfo@ptc.com

詳細については、PTC.com/products/integrity をご覧ください。

© 2012, Parametric Technology Corporation (PTC). All rights reserved. PTC 製品およびサービスに対する唯一の保証は、該当する製品およびサービスに付随する明示の保証書において定められています。本書のいかなる内容も、追加の保証と解釈されてはなりません。本文中で言及されている顧客事例は、単一のユーザー エクスペリエンスおよびかかる顧客の証言に基づいています。PTC 製品およびサービス、または PTC が関係している市場に関するアナリストなどによる将来の見通しは、そのアナリスト自身の見通しであり、PTC はその根拠または正確性について何ら言明いたしません。PTC、PTC ロゴ、およびすべての PTC の製品名およびロゴは、米国およびその他の国における PTC またはその子会社、あるいはその両方の商標または登録商標です。その他の製品名または企業名はすべて、各所有者の商標または登録商標です。新製品や新機能のリリース時期は予告なく変更されることがあります。

7511-Integrity: Managing Varients-WP-JA-0312