Produktvarianten in einer Software-Produktlinie mit Integrity verwalten

Kurzbeschreibung

Die Software-Produktlinienentwicklung (Software Product Line, SPL) ist in vielen Produktentwicklungsorganisationen inzwischen unverzichtbar. Sie ermöglicht die effiziente Verwaltung der unzähligen Produktmerkmale und Plattformvariationen, die zur Sicherung der Wettbewerbsfähigkeit erforderlich sind, ohne die Innovationskraft zu behindern oder drastisch steigende Compliance-Kosten bewältigen zu müssen. Zugleich stöhnen viele Produktentwicklungsorganisationen unter der Vielzahl von unterschiedlichen ALM-Tools (Application Lifecycle Management), die sich im Lauf der Zeit angesammelt haben und nach und nach miteinander gekoppelt wurden, um eine Vielzahl von strikten Entwicklungsprozessen zu automatisieren. Die dadurch entstandene Umgebung ist gekennzeichnet von Redundanz, Ineffizienz, Fehlern, explodierenden Kosten, Konformitätsproblemen und fehlendem Einblick in die Freigabebereitschaft von Produkten.

Neben diesen Herausforderungen entstehen oft neue Probleme, wenn diese Unternehmen in einem solchen Umfeld die Software-Produktlinienentwicklung einführen möchten. Prozesse für das Lebenszyklus-Management und seine Automatisierung spielen dabei eine wichtige Rolle, da der Übergang zur Software-Produktlinienentwicklung Legacy-Prozesse und Automatisierungstools oft an die Grenzen ihrer Leistungsfähigkeit bringt und somit umfangreiche Re-Engineering-, Neuausrichtungs- und Straffungsmaßnahmen erzwingt.

In diesem Dokument werden zentrale Aspekte erfolgreicher Prozesse für das Lebenszyklus-Management und der zugehörigen Automatisierungsplattformen dargelegt, die für die erfolgreiche Implementierung von Software-Produktlinienentwicklungspraktiken entscheidend sind. Außerdem werden einige bewährte Muster für die Verwaltung von Produktvarianten – einen wichtigen Aspekt der Software-Produktlinienentwicklung – in Integrity untersucht, einem Produkt für Software System Lifecycle Management.

Was kennzeichnet einen „guten" Softwareentwicklungsprozess?

Ein guter Softwareentwicklungsprozess ist schlank. Jede Aktivität dient einem Zweck und trägt direkt oder indirekt zu greifbaren Endergebnissen bei. Das bedeutet, dass niemals eine Prozessaktivität durchgeführt wird, nur weil der Prozess das eben so vorsieht. Daraus folgt, dass ein Prozessartefakt nur dann hergestellt werden sollte, wenn es dazu dient, die funktionierende Software bereitzustellen, sie zu warten oder Compliance-Vorgaben zu erfüllen.

Ein Beispiel: In einem strikten Prozess kann kein Änderungsantrag umgesetzt werden, ohne dass ein Änderungspaket erzeugt wird, und ohne entsprechenden Änderungsantrag können keine Änderungen an einem Lebenszyklusartefakt vorgenommen werden. Im Gegensatz dazu ist bei einem weniger formellen Entwicklungsprozess für bestimmte Änderungsanträge unter Umständen kein Änderungspaket erforderlich, und bestimmte Artefaktänderungen sind auch ohne entsprechenden Änderungsantrag zulässig, sofern ein Änderungspaket erzeugt wird, anhand dessen die Änderungen nachverfolgt werden können.

Ein guter Prozess sollte sich außerdem für die Automatisierung des Lebenszyklus eignen. Beispielsweise sollte die Einhaltung von Compliance-Auflagen ein natürliches Nebenergebnis des automatisierten Entwicklungsprozesses sein, mit dem Artefakte für eine Compliance-Einreichung bei Bedarf automatisiert erzeugt werden können, wodurch der Zeit- und Arbeitsaufwand für die Vorbereitung eines Audits signifikant abnimmt.

Ein erfolgreicher Prozess für die Software-Produktlinienentwicklung hat zwar noch weitere wichtige Facetten, doch diese beiden sind in jedem Fall grundlegend, unabhängig von der verwendeten SPL-Methode.

Software System Lifecycle Management und Application Lifecycle Management im Vergleich

Entwicklungsorganisationen, die softwarelastige Produkte entwickeln, waren über viele Jahre hinweg gezwungen, sich gegen eine Branche zu behaupten, die sich auf die ALM-Automatisierung konzentriert, und eine andere, weitestgehend unabhängige Branche, bei der die Automatisierung des Produktlebenszyklus-Management im Vordergrund steht. Durch diese Situation ist ein Vakuum für Softwareentwicklungsteams entstanden, die die immer komplexeren Softwarekomponenten liefern müssen, die in Produkte eingebettet werden. Die Komplexität und der Umfang von Software in Produkten sind in den vergangenen Jahrzehnten sprunghaft angestiegen. Die Fähigkeiten von ALM-Tools, den spezifischen Bedürfnissen von Softwareentwicklern gerecht zu werden, die in einer größeren Produktentwicklungsorganisation tätig sind, haben mit dieser Entwicklung nicht Schritt gehalten. Zur gleichen Zeit wurden die führenden PLM-Produkte zwar nach und nach um neue Funktionalitäten für die Verwaltung von Softwareentwicklungsaktivitäten und -artefakten erweitert, sind jedoch in der Regel in Umfang und Leistungsfähigkeit begrenzt.

Insbesondere sind ALM-Tools nicht in der Lage, den umfangreicheren PLM-Prozess und die Automatisierungsplattformen, in denen sie eingesetzt werden müssen, zu unterstützen. Die Integration verschiedener ALM-Tools, die schon untereinander kaum integriert sind, in ein umfangreicheres Ökosystem zur Automatisierung der technischen Entwicklung, ist eine abschreckende Aufgabe, deren Planung, Implementierung und Instandhaltung in erster Linie dem Kunden überlassen blieb. Zwar bieten PLM- und ALM-Anbieter verschiedene Ansätze für die Integration von Software-Entwicklungstools in die System- und Hardwareentwicklungsumgebung, doch bei den meisten davon steht die Aggregation unzähliger Artefakte der Softwareentwicklung zu einzelnen Posten in der vom PLM-Tool verwalteten Stückliste im Vordergrund. Bei dieser Vorgehensweise werden leider die enorme Komplexität und die steigende Anzahl von Softwarekomponenten in modernen, softwarelastigen Produkten ignoriert. Ein effizienteres Konzept für die Integration der Artefakte und Prozesse der Software-, Hardware- und Systementwicklung ist also dringend erforderlich, um den sich ändernden Bedürfnissen moderner Entwicklungsorganisationen zu genügen.

PTC erfüllt diese Notwendigkeit mit Integrity, einem Einzelprodukt, das von Grund auf für das Software System Lifecycle Management (SSLM) entwickelt wurde. Integrity wurde außerdem von Anfang an für eine nahtlose Integration in die übergeordnete PLM-Umgebung konzipiert, sodass die notwendige feine Sichtbarkeit und Kontrolle von Artefakten und Prozessen zwischen System-, Hardware- und Softwareentwicklung erreicht wird.

Einzelplattform und Werkzeugpalette im Vergleich

Integrity ist das einzige SSLM-Produkt, das Out-of-the-Box-Lösungen zur direkten Bewältigung von Kundenherausforderungen beim Lebenszyklus-Management in der Produktentwicklung ohne substanzielle Erweiterungen bietet. Viele andere Anbieter bieten Werkzeugpaletten[1] mit verschiedenen Tools an, um den gesamten Entwicklungszyklus abzudecken. Diese Tools wurden in der Regel im Laufe der Zeit aus verschiedenen Quellen erworben; nur einige wenige wurden intern entwickelt. Die einzelnen Tools erfordern daher häufig die unterschiedlichsten Implementierungstechnologien und Architekturen, und auch die Benutzerführung ist unterschiedlich. In der Regeln decken sie den Lebenszyklus nur lückenhaft habe, während es an anderen Stellen zu Überschneidungen der Funktionen kommt. Dieses Konzept der Werkzeugpalette erlaubt ohne zusätzliche Mitarbeiter keine effektiven Lösungen für die Automatisierung des Lebenszyklus.

1 Die Werkzeugpalette enthält oft unabhängige Tools, die von verschiedenen Anbietern entwickelt wurden und unterschiedliche Punkt-zu-Punkt-Integrationen aufweisen. Doch so gut diese Tools auch miteinander integriert sein mögen, es sind in jedem Fall beträchtliche Investitionen erforderlich, um die gewünschten Lösungen zu erhalten.

Die Bündelung, Anpassung, und Konfiguration dieser Tools entsprechend den Anforderungen der Organisation dauert Monate. Mit Integrity erzielen Kunden unmittelbar einen Nutzen aus ihrer Investition. Nach und nach entsteht eine umfassende Lösung, die alle Bedürfnisse der Organisation erfüllt. Integrity erlaubt auch enge Verbindungen mit vorhandenen ALM-Tools einzelner Bereiche und Arbeitsgruppen. So entsteht eine zuverlässige und allgemein gültige Datenquelle, die in Echtzeit die Freigabebereitschaft des Produkts anzeigt, – und ein inkrementeller Bereitstellungspfad, der keine störenden Eingriffe in das funktionierende System erfordert.

Effektives Management von Produktvarianten

Um zu demonstrieren, wie Integrity – ein einzelnes Produkt – SSLM-Prozesse automatisiert, gehen wir in diesem White Paper auf das Problem der Verwaltung von Produktvarianten ein, einen zentralen Aspekt der Software-Produktlinienentwicklung. Produktvarianten weisen, ob sie sich nun in ihrer Funktion oder in Parameterwerten unterscheiden, die das Funktionsverhalten steuern, beträchtliche Gemeinsamkeiten auf, die wiederverwendet werden müssen, um die Komplexität zwischen den Varianten zu reduzieren. Werden diese Gemeinsamkeiten nicht effektiv ausgenutzt, führt Duplizierung zu einem exponentiellen Anstieg der Zahl der Artefakte, die für jede Variante verwaltet werden müssen. Dadurch vervielfachen sich die Ressourcen, und die Kosten für die Entwicklung und Pflege der einzelnen Varianten steigen enorm. Dieses Konzept der Duplizierung, der manchmal auch als „Klonen und besitzen" beschrieben wird, führt zu einem Verlust der Verfolgbarkeit und anderer Beziehungen zwischen den „geklonten" Artefakten der verschiedenen Varianten. Einmal geklonte Artefakte, die in mehreren Varianten wiederverwendet werden, entwickeln ein Eigenleben, bis schließlich jede Variante wie ein eigenständiges Produkt entwickelt und gepflegt werden muss.

Für eine erfolgreiche Software-Produktlinienentwicklung müssen die Gemeinsamkeiten von Produktvarianten beibehalten und kontrolliert unter konformen Produktfamilien weitergegeben werden. Integrity ermöglicht es, die Komplexität bei der Verwaltung ganzheitlicher, gemeinsam verwalteter Artefaktsätze oder Ressourcen zu „zähmen", um lieferbereite Produktvarianten zu erzeugen. Im Zentrum von Integrity steht das Konzept der Freigabe von verwandten Funktionalitäten, der Verwaltung von Varianten- und Kernartefakten sowie der geplanten Verteilung und Auswirkungsanalyse. Integrity unterstützt sowohl funktionale Variationen als auch datengesteuerte Softwarevariationen. Gerade letztere sind gerade bei steuerungsorientierten Systemen äußerst wünschenswert, um eine gemeinsame Codebasis zu definieren. Dabei wird der Großteil der Abweichungen im funktionalen Verhalten anhand der Eingabedatenwerte parametrisiert.

Gemeinsam verwaltete Ressourcen für Produktvarianten mit Integrity verwalten

Für die Verwaltung der Produktvariantenentwicklung müssen gemeinsam verwaltete Ressourcen im gesamten Lebenszyklus (z. B. Anforderungen, Entwürfe, Modelle, Code und Tests) unterstützt werden. In diesem White Paper werden die diesbezüglichen Fähigkeiten von Integrity aber am Beispiel des Anforderungsmanagements für Produktvarianten beschrieben.

Anforderungen von funktionalen Varianten[2]

Eine funktionale Produktvariante hat unzählige allgemeine Anforderungen mit anderen Varianten desselben Produkts gemeinsam, weist jedoch zusätzliche Anforderungen auf, die für ihre Struktur und ihr Verhalten spezifisch sind.

Zunächst wird ein Satz von Basisanforderungen festgeschrieben, die allen funktionalen Varianten des Produkts gemeinsam sind.[3] Die Anforderungen für jede einzelne funktionale Variante umfassen dann einige oder alle gemeinsamen Basisvarianten sowie zusätzlich spezifische Anforderungen nur für die jeweilige Variante.

Die folgenden Szenarien verdeutlichen, dass diese Vorgehensweise beim Erstellen, Pflegen und Zertifizieren von funktionalen Varianten für ein Höchstmaß an Effizienz und Präzision sorgt.[4]

Neue Variante

a. Neues leeres Dokument für die neue Variante erstellen: „übergeordnetes Dokument" (siehe Abbildung 1).

b. Zwei Anforderungssätze in dieses Dokument eingeben. Im ersten auf alle oder eine Teilmenge des allgemeinen Anforderungssatzes verweisen. Dieser Satz kann im übergeordneten Dokument, das im vorherigen Schritt erstellt wurde, nicht geändert werden.

c. Als zweiten Anforderungssatz im übergeordneten Dokument die Anforderungen eingeben, die nur für die betreffende Variante spezifisch sind.

d. Dadurch, dass jede Produktvariante auf dieselbe gemeinsame Basis verweisen kann, ist kein Klonen und Besitzen möglich, und die wichtigen Beziehungen zwischen den allgemeinen, gemeinsam verwalteten Anforderungen unter mehreren Varianten bleiben erhalten.

2 Denken Sie daran: Die folgenden Szenarien dienen lediglich als Beispiele für die Verwaltung von funktionalen und parametergetriebenen Variantenanforderungen mithilfe von Integrity. Die Szenarien sind variabel und können noch viel enger auf die Benutzerbedürfnisse zugeschnitten werden.

3 Der Einfachheit halber erfolgt die Erstellung der gemeinsamen Basisanforderungen hier vor der Erstellung von Produktvariantenanforderungen. In der Realität ergeben sich die gemeinsamen Basisanforderungen allerdings oft erst nach und nach, wenn bereits die Anforderungen für zwei oder mehr Varianten erstellt wurden. Die Hauptmerkmale der hier dargestellten Szenarien bleiben jedoch gleich.

4 Die Szenarien in diesem White Paper berücksichtigen keine komplexeren Situationen wie mehrere Übereinstimmungsebenen. Auch in solchen komplexeren Szenarien hat eine gemeinsame Basis ihre Berechtigung, allerdings ist eine stärkere Beteiligung erforderlich.

Zertifizierung/Rezertifizierung

a. Eine einmal zertifizierte gemeinsame Basis muss nicht jedes Mal rezertifiziert werden, wenn eine neue Variante auf sie verweist.

b. Nur die spezifischen Anforderungen der Variante (also diejenigen im übergeordneten Dokument, das oben im Szenario „Neue Variante" erwähnt wurde) müssen zertifiziert werden.

c. Wenn Änderungen an der gemeinsamen Basis vorgenommen werden (siehe Szenario „Änderungsantrag"), muss eine Rezertifizierung für sämtliche Produktvarianten durchgeführt werden, die auf die geänderten Anforderungen in der gemeinsamen Basis verweisen. Die Bestimmung der betroffenen Varianten – sowie der nachgelagerten, gemeinsam verwalteten Entwicklungsressourcen, die ggf. ebenfalls für jede Variante rezertifiziert werden müssen, – ist allerdings einfach und präzise möglich, da Integrity die richtigen Beziehungen zwischen gemeinsam verwalteten Ressourcen innerhalb der Produktlinie beibehält.

d. Zeit und Kosten für die Herstellung einer Compliance-Einreichung werden drastisch reduziert.

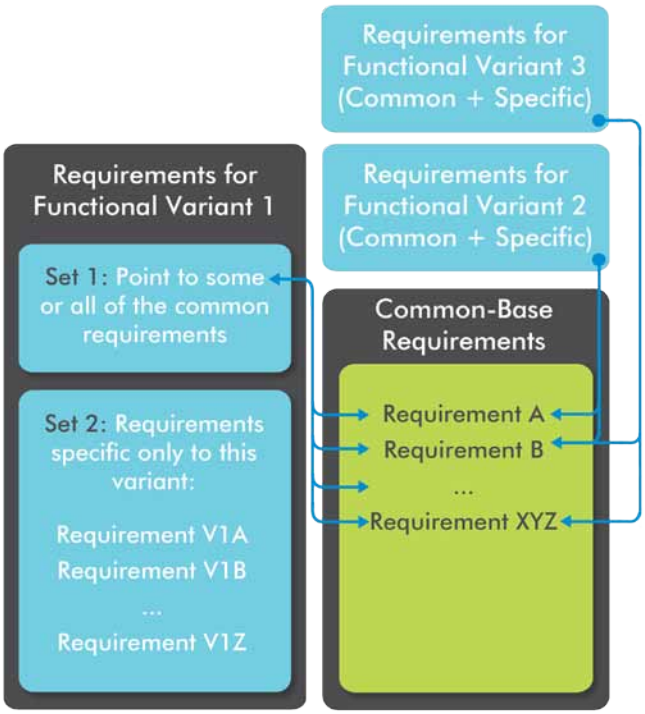


Abbildung 1: Mehrere funktionale Produktvarianten, die die Anforderungen derselben gemeinsamen Basis verwenden. In diesem Beispiel enthält Variante 1 eigene spezifische Anforderungen sowie sämtliche Anforderungen in der gemeinsamen Basis. Variante 2 und 3 enthalten hingegen ihre jeweiligen spezifischen Anforderungen und eine Teilmenge der gemeinsamen Basis.

Änderungsantrag

a. Beträchtliche Einsparungen bei Kosten, Zeit und Qualität werden realisiert, wenn der Änderungsantrag Elemente der gemeinsamen Anforderungen betrifft.

b. Die Änderung wird nur einmal an der gemeinsamen Basis vorgenommen und in einer neuen Version gespeichert.

c. Zukünftige Varianten können auf diese neue Version verweisen (siehe Abbildung 2 unten), aber aktuelle Varianten werden erst automatisch geändert, sodass sie auf die neue Version verweisen, wenn geklärt ist, dass eine Variante die Änderung erben soll. Auf diese Weise bleibt die Zertifizierung für jede der aktuellen Varianten erhalten (also ursprüngliche gemeinsame Basis einmalig zertifiziert + variantenspezifische Anforderungen zertifiziert pro Variante).

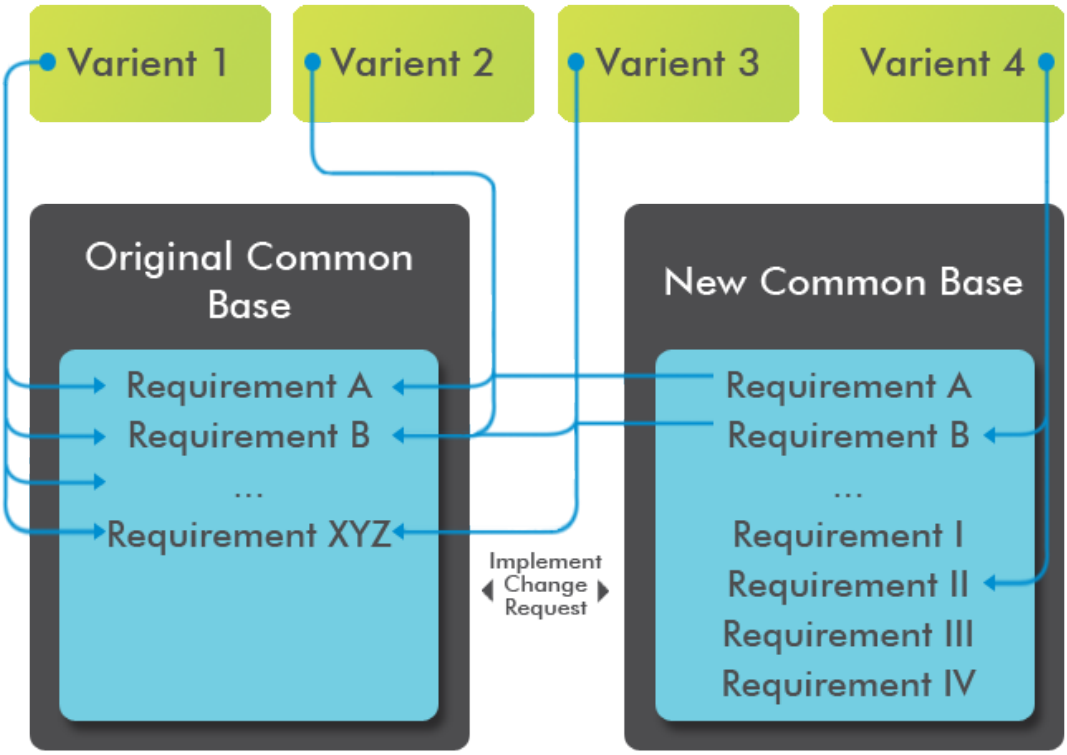


Abbildung 2: Wenn eine neue Version der gemeinsamen Basis produziert wird, verweisen neue Varianten ggf. auf deren Anforderungen. Vorhandene Varianten können hingegen weiterhin auf die Originalversion der gemeinsamen Basis verweisen, um ihre Zertifizierung zu behalten, bis sichergestellt ist, dass eine Variante bereit ist, die Änderung zu übernehmen. Vom Standpunkt der Architektur aus betrachtet sind Anforderungen, die in der neuen gemeinsamen Basis gleich bleiben, lediglich Zeiger auf dieselben Anforderungen in der ursprünglichen gemeinsamen Basis.

Anforderungen von parametergetriebenen Varianten[5]

Eine parametergetriebene Produktvariante stellt eine allgemeine Vorlage von parametrisierten Anforderungen dar und ordnet ihre spezifischen Werte einigen oder allen Parametern in der allgemeinen Vorlage zu. Diese gemeinsame Basisvorlage enthält Parameter, deren Werte pro Produktvariante geändert werden können. Solche parametrisierten Anforderungen können in kontrollorientierten Systemen äußerst wünschenswert sein, um eine allgemeine Codebasis zu schaffen, die basierend auf Eingabedatenwerten parametrisiert wird.

Beispielsweise könnte eine Anforderung in der gemeinsamen Basis lauten: „Die maximal zulässige Temperatur beträgt {{TempMax}} Grad" (siehe Abbildung 3). Die Variable {{TempMax}} kann je nach Produktvariante auf unterschiedliche Werte festgelegt werden. In Integrity kann jede Produktvariante auf dieselbe physische gemeinsame Basis verweisen, während viele Parameter bei Bedarf für die Produktvariante geändert werden.

Die folgenden Szenarien verdeutlichen, dass diese Vorgehensweise beim Erstellen, Pflegen und Zertifizieren von parametergetriebenen Varianten für ein Höchstmaß an Effizienz und Präzision sorgt.[6]

5 In den hier dargestellten Beispielszenarien werden funktionale Varianten von parametergetriebenen Varianten getrennt behandelt. Die beiden Variantenarten können jedoch in jeder Entwicklungsumgebung auch miteinander kombiniert auftreten.

6 Die Szenarien in diesem White Paper berücksichtigen keine komplexeren Situationen wie mehrere Übereinstimmungsebenen. Auch in solchen komplexeren Szenarien hat eine gemeinsame Basis ihre Berechtigung, allerdings ist eine stärkere Beteiligung erforderlich.

Neue Variante

a. Neues leeres Dokument für die neue Variante erstellen: „übergeordnetes Dokument".

b. Im Dokument auf den allgemeinen Anforderungssatz verweisen, der einen Basissatz von bereits zertifizierten Parameterwerten enthält.

c. Nur die Parameter ersetzen, die für die neue Variante im gemeinsamen Basissatz geändert werden müssen. Das Ergebnis der neuen Ersetzungen werden sofort im Kontext der allgemeinen Anforderungen angezeigt. Dank dieses unmittelbaren, visuellen Feedbacks kann der Benutzer die Genauigkeit der Anforderungen sicherstellen. Dadurch kann er die Korrektheit der Anforderungen gezielt steuern, statt sie lediglich durch Inspektionen in späteren Entwicklungsphasen zu überprüfen, in denen die Behebung von Anforderungsfehlern äußerst teuer wird.

d. Dadurch, dass jede Produktvariante auf dieselbe gemeinsame Basis verweisen kann, ist das „Klonen und besitzen" ausgeschlossen, bei dem eine neue physische Kopie des allgemeinen Anforderungssatzes für jede Variante entstehen würde.

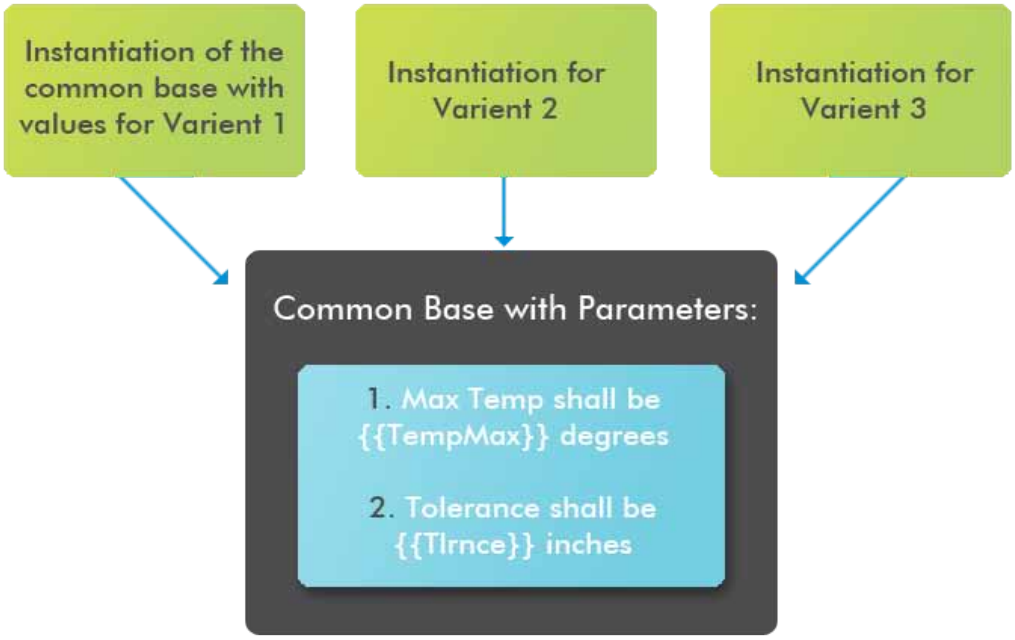


Abbildung 3: Mehrere parametergesteuerte Varianten können dieselbe physische, parametrisierte gemeinsame Basis darstellen.

Zertifizierung/Rezertifizierung

a. Eine einmal zertifizierte gemeinsame Basis muss nicht jedes Mal rezertifiziert werden, wenn eine neue Variante auf sie verweist.

b. Nur die neu dargestellten Werte, die für die Variante spezifisch sind, müssen zertifiziert werden.

c. Wenn Änderungen an der gemeinsamen Basis vorgenommen werden (siehe Szenario „Änderungsantrag"), muss eine Rezertifizierung für sämtliche Produktvarianten durchgeführt werden, die auf die geänderten Anforderungen in der gemeinsamen Basis verweisen. Die Bestimmung der betroffenen Varianten – sowie der nachgelagerten, gemeinsam verwalteten Entwicklungsressourcen, die ggf. ebenfalls für jede Variante rezertifiziert werden müssen, – ist allerdings einfach und präzise möglich, da Integrity die richtigen Beziehungen zwischen gemeinsam verwalteten Ressourcen innerhalb der Produktlinie beibehält.

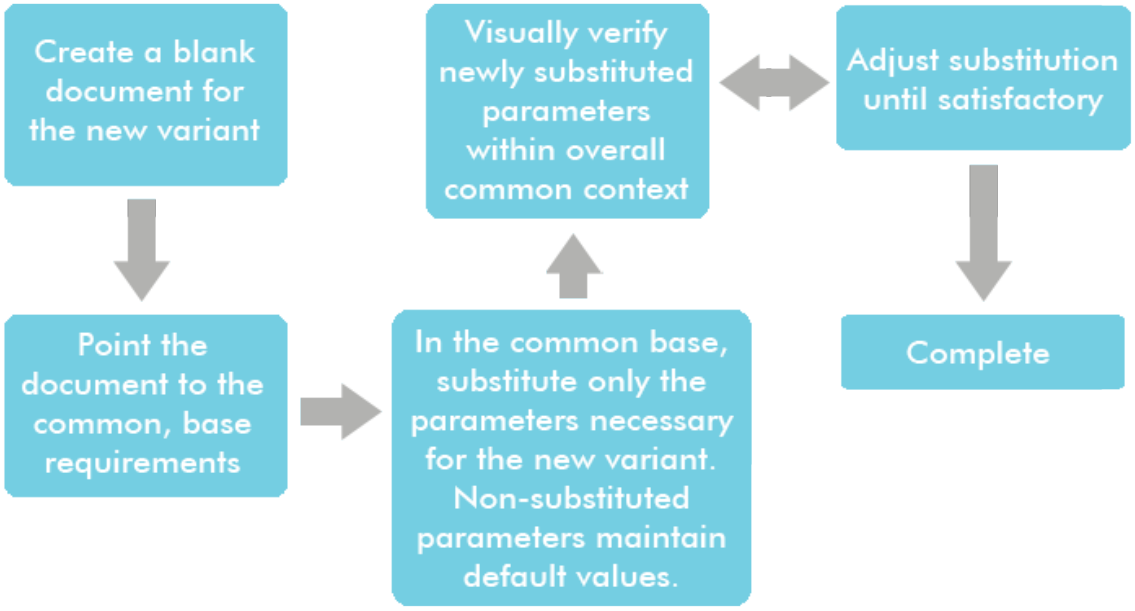
d. Zeit und Kosten für die Herstellung einer Compliance-Einreichung werden drastisch reduziert.

Änderungsantrag

a. Beträchtliche Einsparungen bei Kosten, Zeit und Qualität werden realisiert, wenn der Änderungsantrag Elemente der gemeinsamen Anforderungen betrifft.

b. Die Änderung wird nur einmal an der gemeinsamen Basis vorgenommen und in einer neuen Version gespeichert.

c. Zukünftige Varianten können auf diese neue Version verweisen (siehe Abbildung 2 oben), aber aktuelle Varianten werden erst automatisch geändert, sodass sie auf die neue Version verweisen, wenn geklärt ist, dass eine Variante die Änderung erben soll. Auf diese Weise bleibt die Zertifizierung für jede der aktuellen Varianten erhalten (also ursprüngliche gemeinsame Basis einmalig zertifiziert + variantenspezifische Anforderungen zertifiziert pro Variante).



Fazit

Integrity ermöglicht Unternehmen, die Software-Produktlinien-entwicklungspraktiken implementieren, eine effektive und effiziente Verwaltung der Komplexität von Produktvarianten. Integrity verwaltet eine einzige physische Kopie der allgemeinen, gemeinsam verwalteten Ressourcen einer Software-Produktlinie. Gemeinsame Anforderungen und andere Lebenszyklus-Ressourcen werden dadurch viel effizienter verarbeitet als bei herkömmlichen Verfahren, bei denen die Anzahl der Anforderungen drastisch ansteigen kann, was wiederum zu fehleranfälliger Duplizierung und explodierenden Kosten führt. Integrity wahrt außerdem die Verfolgbarkeitsbeziehungen zwischen allgemeinen, gemeinsam verwalteten Ressourcen und variantenspezifischen Ressourcen. Dadurch erhalten die Teams einen klaren Überblick über die Beziehungen zwischen bestimmten Varianten und den gemeinsam verwalteten Kernressourcen. Die erweiterten Funktionen von Integrity für das Variantenmanagement ermöglichen die effektive Umsetzung der Software-Produktlinienentwicklung, ohne gemeinsam verwaltete und variantenspezifische Ressourcen im Lebenszyklus außer Kontrolle zu verlieren.

Standorte der Integrity Geschäftseinheit

Nordamerika:
+1 800 613 7535

Großbritannien:
+44 (0) 1252 453 400

Deutschland:
+49 (0) 711 3517 750

Asiatisch-pazifischer Raum:
+65 6830 8338

Japan:
+81 3 5422 9503

integrityinfo@ptc.com

Weitere Informationen: PTC.com/product/integrity

© 2012, Parametric Technology Corporation (PTC). Alle Rechte vorbehalten. Die Inhalte dieser Seiten werden ausschließlich zu Informationszwecken bereitgestellt. Sie können ohne Vorankündigung geändert werden. Die Gewährleistungen für PTC Produkte und Dienstleistungen sind in den ausdrücklichen Gewährleistungsangaben der jeweiligen Produkte und Dienstleistungen festgelegt, und die hierin enthaltenen Informationen enthalten keinerlei zusätzliche Gewährleistung. Verweise auf die Erfolge von Kunden basieren auf den Erfahrungen eines Einzelbenutzers und auf Nachweisen von Kunden. Analytische oder zukunftsorientierte Aussagen über PTC Produkte und Services oder die Märkte, auf denen PTC agiert, sind Aussagen von Analysten, und PTC macht keine Zusicherungen hinsichtlich deren Genauigkeit. PTC, das PTC Logo und alle PTC Produktnamen und Logos sind Warenzeichen oder eingetragene Warenzeichen von PTC und/oder Tochterunternehmen in den USA und anderen Ländern. Alle anderen Produkt- oder Firmennamen sind Eigentum ihrer jeweiligen Besitzer. Releasetermine sowie Funktions- oder Leistungsumfang können nach Ermessen von PTC geändert werden.

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