SIEMENS

Ingenuity for life

Modern realities – options, variations and choices

Variety is the spice of life

Mass customization is increasingly being enabled and facilitated by software. Theoretically, software variation has no limits, and can be customized to an individual level. This paper explores the implications for software developers and the lifecycle management of software applications.

A white paper issued by: Siemens PLM Software

www.siemens.com/polarion

Contents

Introduction

We live in a fantastic world of connection and customization. In many respects these two concepts define us in modern terms. We celebrate our personal connections using social media. We are constantly connected to friends and family by instantaneous and ubiquitous means of communication. We rely daily upon a world of connected devices for both convenience and safety. To the extent that society allows, we exercise our ability to customize our world to our liking as individuals. We crave choices, and we expect them, often expecting virtually unlimited choices. We are frustrated when choices are limited.

Commercial success has forever hinged on an ability to provide variety and choices. The idea has been embraced as part of the manufacture of products in various industries, in particular quite vividly in the manufacture of automobiles. Although Henry Ford is credited with being the innovator of factory assembly-line manufacturing techniques to mass produce automobiles, he is also famously recalled as saying "Any customer can have a car painted any color that he wants so long as it is black." Automobile makers soon realized the tremendous market power of providing differences, varieties and options. Customization quickly became a cornerstone of continuing commercial success.

Fast forward one hundred years, and the level of customization, that is, the volume of choices and options made available by automobile manufacturers, seemingly defies every concept of mass production. The number of possible options that vehicle manufacturers now provide is so great that Mercedes Benz postulates that it is statistically possible that every car made of a particular model in a given year can in fact be different.

Mobile telecommunications provide another shining example of an industry steeped in variations. Smartphones offer considerable consumer choice, and are produced in such a way that a single device model must accommodate a variety of communication carriers and infrastructures, that may in turn vary widely according to the global geographies in which the devices are being used. Consumer expectations are also high, and it is a market assumption that every user's favorite downloaded apps will work equivalently and seamlessly, regardless of the types of smartphone devices that they may elect to purchase and carry with them.

Once thought of as purely conceptual, mass customization has become reality. Enabled by ubiquitous customer communication, we can now customize everything from the cars we drive to the shoes we wear. This explosion of choice and variation is accelerating, and permeating every product and service.

How is this new reality being accomplished? Increasingly it is being enabled and facilitated by software. Not just in capturing and communicating customer wants, or driving a manufacturing process, software progressively makes variety possible. Where there may be physical and practical limits to mechanical variation, software variation is considered theoretically to have no limits other than imagination. In the extreme, software can be customized to an individual level – made for a market of one (seemingly a return to an earlier era when all software was custom-made, but now only in terms of a configured-to-order delivered result, not in terms of individually engineered-to-order custom coding). Physical and virtual realms are often combined – a range of options in a general-purpose physical component are made possible entirely through its embedded software.

What does this new reality mean to software developers, and how does it affect lifecycle management of software applications? This paper explores some answers to such questions, and suggests that the art and science of software development, and the lifecycle processes employed to deliver high-quality software, require adaptations, adjustment and supplementation in order to accelerate continued effectiveness in response to a proliferation of software variants.

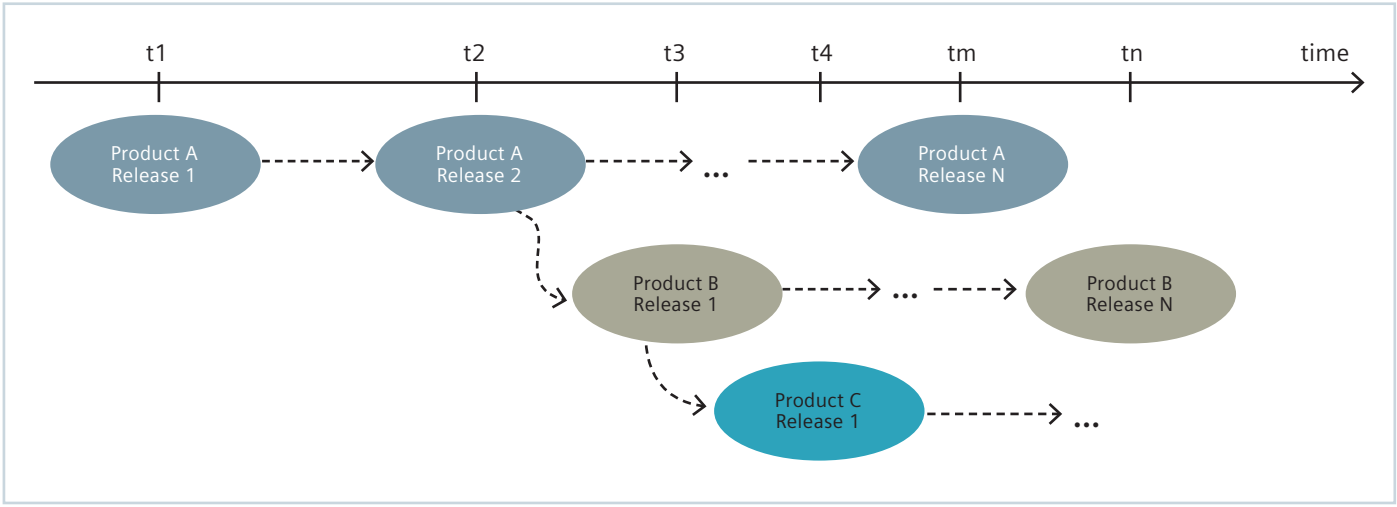
Tactical approaches

Traditional responses to software variation are insufficient
Software development organizations are forever challenged and pressured to author more, to do it faster and to release higher-quality deliverables. The introduction of software variants places even more pressure on software development teams. Traditional development responses to the ensuing disruption become insufficient as levels of software variation increase and/or accelerate.

Agile development practices
Recently, Agile development approaches have gained widespread popularity as a way to address both the need to develop faster, and to provide more successful results with respect to fulfilling software requirements. While the so-called Agile methodology certainly allows developers to more rapidly iterate releases, and ensures the ongoing validation of delivered functionality throughout the development process, Agile methodologies and practices do not specifically contemplate variants management in needed fundamental ways. Hence, the adoption of an Agile development methodology is not an effective strategy, in and of itself, to implement variant management in the development lifecycle (although the combination of Agile practices with other means of variant management may provide hugely synergistic benefits to a development effort). Increasing the speed of development, that is, release rapidity, is not scalable to accommodate increasing numbers of variants, and thus will require unrealistic resourcing levels and/or will introduce resulting organizational complexity, which even then will not provide an appropriate contextual framework to efficiently manage variants.

Adaptations of version-oriented release management
Development practice has long leveraged cloning strategies (also known as "clone and own," "branching and merging," as well as other terms) as a re-usability technique in order to speed development processes and achieve quality outcomes. Re-use is essentially basic human behavior as a response mechanism – solve thorny problems with known solutions, and/or by adapting similar solutions, in order to save time.

While leveraging the re-use of components from predecessor and parallel projects is one of the keys to variants management, adaptations of traditional version-oriented approaches are inherently limited with respect to accommodating broad levels of re-usable assets, as well as being inherently single-project focused. Mass variation (development of numerous, possibly similar components, in multiple projects) soon overwhelms the efficacy of branching techniques. Conflicts between branches may remain unidentified, and/or the ability to establish a new branch is inhibited by known conflicts. The same functionality may be developed repeatedly for slight



variations, and/or seemingly identical functions behave differently depending upon their position in a branching hierarchy. Traceability and accountability are the first casualties when adapting single-project, version-oriented techniques for variants management, and often accountability is completely lost. Testing cycles are lengthened, quality becomes compromised, and validation for regulatory and/or standards compliance becomes problematic. Maintenance efforts explode exponentially, as complexity is increasing with each branching operation, and traceability becomes more and more obfuscated. Merging changes becomes multifarious, and in the worst of cases merging becomes entirely impossible because of conflicting changes.

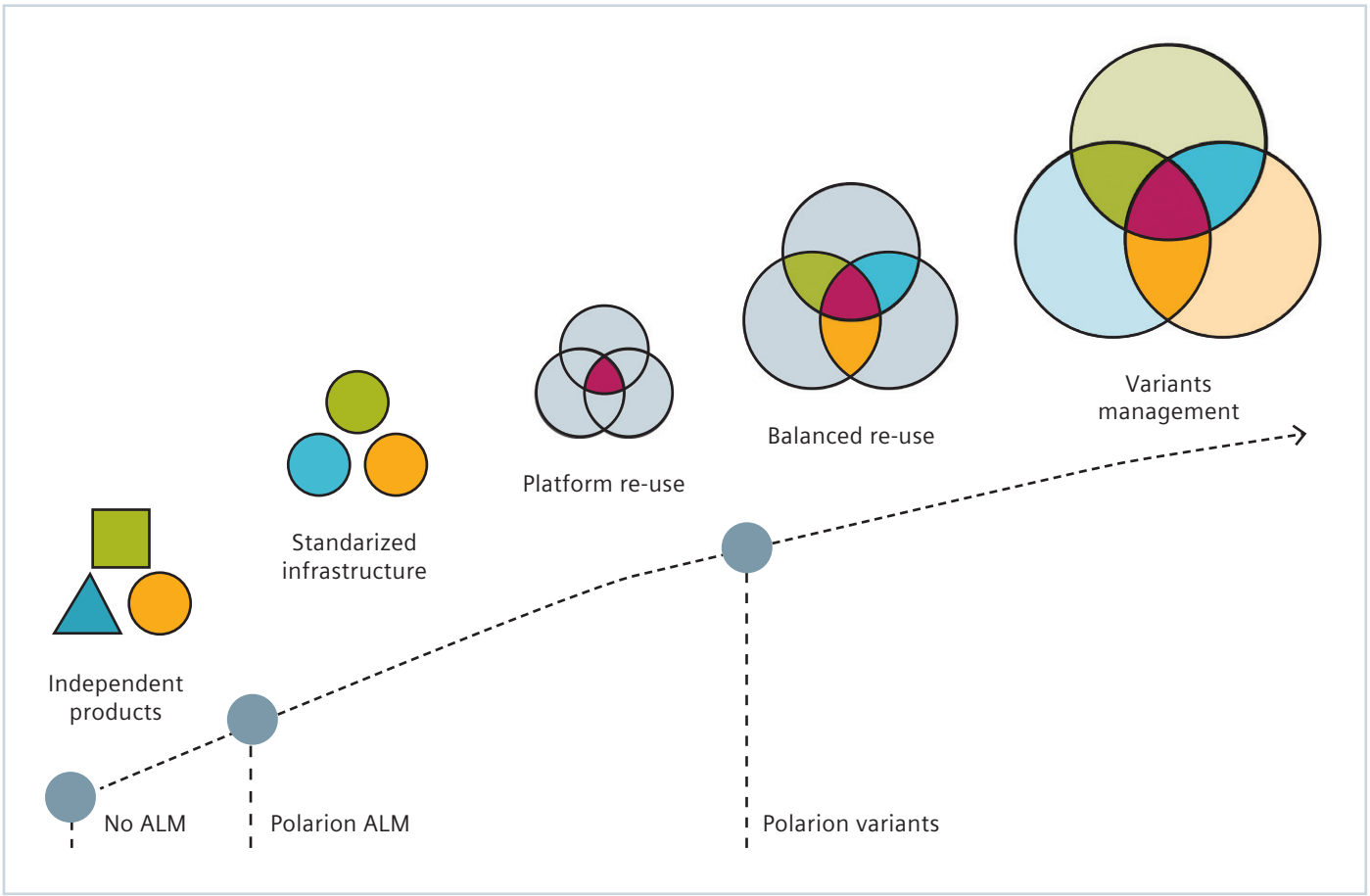
Technical re-architecture

Re-architecting an existing solution to provide more robust variation points is often viewed as a tempting alternative, the idea being that if we can essentially entrench more modularity and capability into the technical architecture, we can better accommodate variability. Stable and flexible architectures are important, but focusing too much on technical architecture will result in shallow implementations that are largely insubstantial with respect to a needed broadening of ongoing software development processes to include the capability required to support increased variation. Similarly, conditional compilation is often viewed as another related architectural technique intended to facilitate software variation. However, conditional compilation is not applicable across all programming paradigms, and whenever utilized software variation remains "buried" within source code, it is difficult to interpret and to represent due to a lack of transparency, especially for the nested conditional dependencies. Integrating re-architected solutions is an even more disruptive activity, and often requires too many changes and compromises that become unacceptable, and/or impractical either technologically or economically. Quality is, as always, a significant barrier. Market resistance may also be an overriding consideration – customers have come to expect a familiar configuration, and newly architected solutions may not be well received, threatening continued success and/or precluding market growth.

ALM – application lifecycle management

Software development has evolved significantly over a relatively short time period, enabled in large measure by an acceptance of lifecycle management approaches, using standardized artifacts, infrastructures and workflows. Customers using Polarion application lifecycle management (ALM) solutions from Siemens PLM Software experience profound benefits to development productivity, efficiency and quality by deploying Polarion's unified suite of ALM capabilities, spanning all phases – from elicitation, through software requirements, through testing/validation and ultimate release, all the while preserving comprehensive traceability, enabling collaborative agility and effectively managing change.

With this standardized ALM infrastructure, development strategies for re-use can evolve to more holistic, platform-oriented visions for application development. The platform viewpoint contains the development infrastructure as well as the functionality that is common to all applications. This brings commonalities into focus, enabling a minimalistic accommodation of variability, since core functionality can now be shared across several projects and products. Projects and products sharing such core functionality are usually of a similar type or intended for a similar problem domain. Cloning (branching and modifying) remains a necessary tactic to introduce new variation.



Product line engineering – foundational concepts
Moving beyond minimalistic accommodations for variation to richer development environment capabilities to support growing mass customization expectations compels new innovation to augment existing ALM domains.

Developers have long studied and debated, largely on an academic basis, the relative merits of adopting many of the proven concepts used in manufacturing for the purposes of software development. One borrowed idea is the concept of product lines, in which a portfolio of related products is engineered to take advantage of similarities while still respecting differences. These fundamental concepts are the foundation that makes it possible for manufacturing processes to effectively support widespread variations. For example, the foundational engineering practices employed by vehicle manufacturers make it possible to offer the vast array of options and choices prevalent in the automotive marketplace.

However, product line concepts are disruptive to entrenched software development practices, and they touch all aspects of software development, so their adoption for software engineering purposes is resisted, typically driven only out of absolute necessity, and implementation of the practices often progresses at a glacial pace.

Polarion Variants™ software is firmly rooted in these concepts, distilling the benefits of product line engineering approaches into tangible capabilities that are readily exploitable to augment and accelerate ALM capacity to successfully manage software variation. Polarion has made it possible for ALM practitioners to immediately realize the associated benefits in logically incremental ways to embrace growing levels of variation, and to avoid or mitigate many of the impediments associated with the introduction of a seemingly disruptive new paradigm. Organizations that are already steeped in product line engineering principles and practices will immediately have a familiar and intuitive toolset at their disposal.

New era of capabilities

Polarion Variants, paired with Polarion ALM, accommodates greater levels of variation and re-usability by using platform approaches in a more balanced way to identify and manage both common and shared elements as a core asset. The amount of functionality in the resulting platform is increased to the point where functionalities shared by several projects, the common core plus the overlapping shared elements, are made precisely visible, and a baseline of management controls which shared assets should be applied to which selected variants.

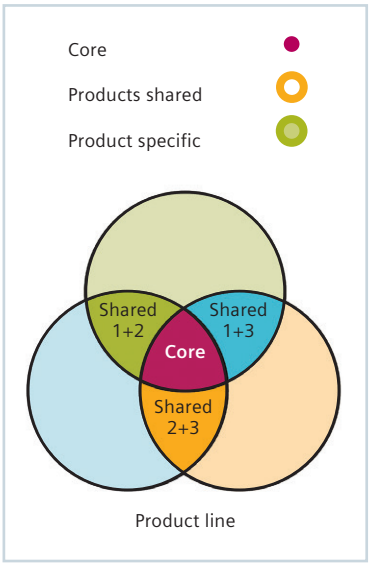
Balanced re-use – 90 percent approach to variant management

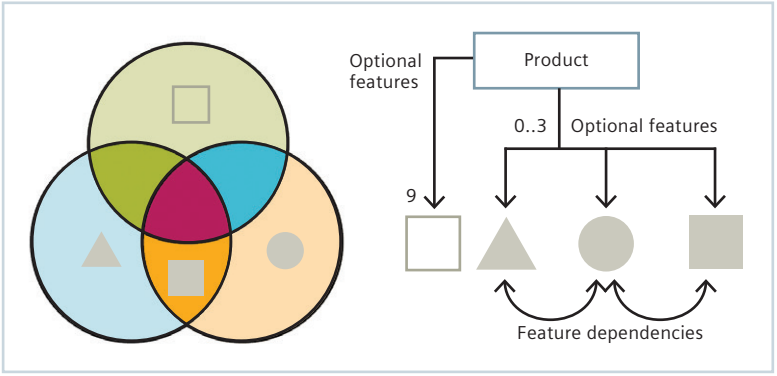
Polarion's utilization of a repository to facilitate a versioning history, and thus to provide an audit trail for all development artifacts, is further expanded by Polarion Variants to effectively provide an asset base of the shareable and re-usable building blocks that are used to establish variation diversity. ALM's collaborative facilities and powerful workflow management capabilities are leveraged by Polarion Variants for the systematic scoping, requirements prioritization and approval processes needed for the lifecycle management of the common core and shared elements. Development efficiency is increased not only with regard to differing variants, but is also maintained across multiple teams working on multiple projects, thus minimizing duplicated efforts and revealing opportunities where previously implemented variation can be integrated into the common platform(s).

The 150 percent solution to variant management

Polarion Variants can further implement a truly maximalist approach to variant management, allowing for the management of both common, shared and varying elements, as well as managing their interdependencies within the context of a software lifecycle. Re-usability is maximized to encompass all development artifacts, from requirements to testing, for all common core, shared segment, and unique elements. Functionality can be expressed in terms of features, while remaining within the familiar framework of requirements.

The real magic is that requirements-driven software development (release deliverables, artifacts, workflows, test cases, etc.) can be shaped according to feature mapping, the combination of a specific mix of features in any variant. In doing so, variability becomes a characteristic that is empowered to differ, no matter if referring to variations in functionality, supported environments, quality attributes, business constraints, etc. The set of requirements for each variant are dynamically shaped according to the specified feature map. As features are added to or subtracted from the feature map, the requirements for the variant, including any/all related components and/or artifacts, are automatically adjusted accordingly. The resulting development activities associated with introducing a new variant are radically streamlined. Specification of mutually exclusive variations and/or options, features which must not or cannot be combined, are automatically curtailed early in the development lifecycle. The complexities associated with making changes to variants are effectively bridged such that the impact of change is fully exposed and understood, and the outcomes of change become more successful.





Traceability preserved across all variations

With Polarion Variants, the comprehensive bidirectional traceability inherent in Polarion ALM (available at a granular work-item level) is expanded to include all elements and artifacts pertaining to the introduction of substantial levels of variation, making it possible to assess the impact of change across a spectrum of variants, including the impacts to all development artifacts, workflows, and process attributes. Such robust traceability across all variants renders risk assessment an efficient and optimized activity. Backwards traceability across variants minimizes risk exposure. Historical forensic-level traceability provides accountability and defensibility, making it possible to trace the genesis of any/all variants, and introduces the notion of a "pedigree" that makes clear the precise lineage of each variant. Verification for compliance purposes can be organized across the totality of common elements, shared elements and resulting variants. Verification and validation (V&V) processes can be synchronized and automated within the lifecycle of development. Exploitation of previous and/or concurrent verification workflows and outcomes provides labor-saving and cost-effective benefits to difficult compliance regimens necessitated by product/process variation.

QA for variants

Feature-shaped requirements development, along with the comprehensive traceability enabled by Polarion Variants, optimizes the complex quality assurance processes needed for testing widespread variation. Testing variants becomes a composite of re-usable artifacts, combined and automated by testing workflows, according to the relevant feature mapping. Quality can be managed and maintained across all variants emanating from different teams spread across widespread locations. Combinatorial testing can be articulated in a context of feature combinations, traced to the feature-shaped requirements, thus pushing defect identification in variants to occur earlier in the development lifecycle. Integration with third-party testing tools, through Polarion's Open Application Programming Interface (API), remains fully exploitable and amplified when combined with Polarion Variants. The outcomes of testing processes are accumulated and aggregated across all variants, providing an organization's managers with complete and comprehensive views into the state of quality among all software and/or system variants.

Summary

The desires for variety and choice arise from basic human emotions. Once experienced in a particular domain, the expectations of variety, options and customization are intensified. An ability to deliver variation is both a differentiator as well as a market imperative for success and survival. Increasing variation will be progressively made possible by software, and this will in turn have profound impact on all aspects of software development lifecycles. Already an imperative in many industries, software product variation is escalating both in terms of volume and criticality.

The range of variation implemented by software, even in safety-critical applications, continues to become more expansive, but still must be implemented without sacrificing quality or safety integrity levels.

Re-usability must move beyond tactical settings to more strategic re-use scenarios involving all development assets in order for software to contribute more in providing mass variation as a key differentiator driving competitive advantage. Managing change, the most labor-intensive aspect in a development lifecycle, becomes a particularly challenging and problematic dilemma when software variants enter the equation.

Additionally, time-to-market expectations are continuing to be shortened to better respond to the challenges of dynamic problem domains, and/or for purposes of competitive advantage.

Different approaches in managing software development are needed. By borrowing from lessons learned in manufacturing, software developers can do more than simply cope. They can embrace and exploit software variation in the development lifecycle.

Polarion Variants makes incremental exploitation of product line concepts possible, and can significantly strengthen existing implementations of product line engineering practices. Polarion Variants supplements application lifecycle management (ALM) to usher in new modes of achieving requirements-driven development processes which can encompass and deliver mass customization.

Polarion Variants broadens ALM capabilities in vital and essential ways to make the effective management of software variants a reality.

Siemens PLM Software

Headquarters
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific
Suites 4301-4302, 43/F
AIA Kowloon Tower,
Landmark East
100 How Ming Street
Kwun Tong, Kowloon
Hong Kong
+852 2230 3308

About Siemens PLM Software

Siemens PLM Software, a business unit of the Siemens Digital Factory Division, is a leading global provider of product lifecycle management (PLM) and manufacturing operations management (MOM) software, systems and services with over 15 million licensed seats and more than 140,000 customers worldwide. Headquartered in Plano, Texas, Siemens PLM Software works collaboratively with its customers to provide industry software solutions that help companies everywhere achieve a sustainable competitive advantage by making real the innovations that matter. For more information on Siemens PLM Software products and services, visit www.siemens.com/plm.

www.siemens.com/plm

© 2017 Siemens Product Lifecycle Management Software Inc. Siemens, the Siemens logo and SIMATIC IT are registered trademarks of Siemens AG. Camstar, D-Cubed, Femap, Fibersim, Geolus, I-deas, JT, NX, Omneo, Parasolid, Solid Edge, Syncrofit, Teamcenter and Tecnomatix are trademarks or registered trademarks of Siemens Product Lifecycle Management Software Inc. or its subsidiaries in the United States and in other countries. All other trademarks, registered trademarks or service marks belong to their respective holders.

55654-A10 4/17 F