Paper PP14 – PHUSE US CONNECT 2024

Software Product Lines Framework:
How to manage end-to-end metadata-driven automation at scale

Stuart Malcolm.
Head of Standards, Efficiency, and Automation.
Veramed

ABSTRACT

CDISC's ambition of developing and extending clinical standards with "the additional semantics needed to support metadata-driven automation across the end-to-end clinical research data lifecycle" is compelling, and promises benefits of "substantially improved efficiency, consistency, and re-usability across the clinical research data lifecycle." [1]

To deliver on this vision means replacing today's manual processes with new processes that support automation, not for a single trial, but for all trials across all phases, treatments, devices, therapeutic areas, etc.

The problem this paper explores is how to manage end-to-end automation such that the time and cost of maintenance across all studies does not counter the efficiency gains.

In short - how do we scale end-to-end automation?

This paper proposes using Software Product Lines (SPL) as a framework for managing automation of clinical trial projects at scale. Product lines are a well-established concept in manufacturing, and Software Product Lines are a new application of this proven concept; developed by the Software Engineering Institute (SEI) published as an open standard, currently on version 5.0 and has been successfully used in regulated industries such as Aviation, Defense or Automotive over the previous two decades. [2]

BUILDING CLINICAL TRIALS FROM COMPONENTS

A Software Product Line consists of a core set of reusable components or modules, analogous to LEGO™ bricks, which represent common features and functionalities shared across a family of related software products.

These reusable components serve as the building blocks that can be assembled and configured in various ways to meet specific requirements. Just as LEGO™ bricks allow for the creation of diverse structures by combining and arranging pieces, SPL enables the development of a range of software-based products by selectively combining and configuring modular components.

In this analogy, the "pile of bricks" represents a common asset base of reusable components which can be used along with a set of instructions (in SPL terminology, a Production Plan) that describes which bricks to select, steps to construct, and check that the product is constructed correctly (by comparing it to the pictures in the instructions).

Many different products can be built from the same set of reusable components.

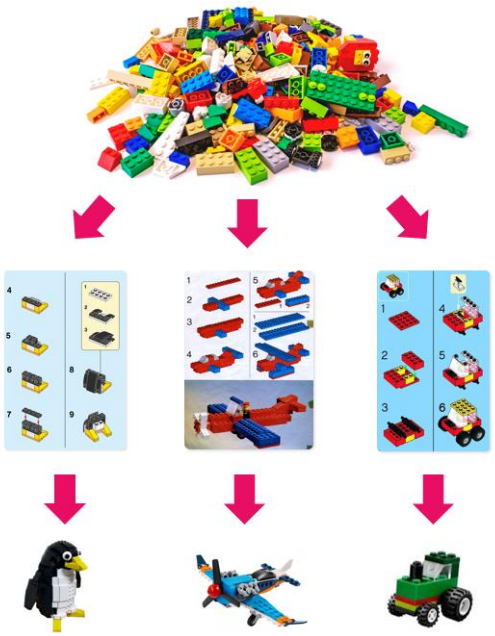


Figure 1 SPL building block analogy

INTRODUCTION TO SOFTWARE PRODUCT LINES

Software Product Lines (SPL) was created as a paradigm shift in software engineering, providing a systematic approach to developing a family of related software products. Unlike traditional software development, which often involves building individual products from scratch, SPL emphasizes the reuse of common features and components across a range of products within a specific domain. This innovative approach enables organizations to efficiently address diverse customer needs while maintaining consistency and reducing development costs. By creating a core set of assets and variability points, SPL allows for the systematic derivation of customized products, fostering both flexibility and scalability. The adoption of Software Product Lines has proven instrumental in streamlining development processes, accelerating time-to-market, and enhancing the overall quality of software products in a variety of industries.

> A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Figure 2 SPL Definition

A COMMON, MANAGED SET OF FEATURES

The term "common managed features" in the context of software product lines refers to a set of features that are shared and centrally managed across all products within the product line. These features are considered common because they are present in every product variant, providing a shared foundation for the entire family of related software products.

The concept of common managed features is essential for achieving a balance between variability and commonality in a software product line. Here's a breakdown of the key elements:

- **Common Features:** These are functionalities or characteristics that are universally present across all products within the software product line. These features represent the core functionality or characteristics that are fundamental to the entire family of products.
- **Managed Features:** The term "managed" implies that these common features are centrally controlled, maintained, and evolved. Central management helps ensure consistency, reliability, and maintainability across all products. Any updates, improvements, or changes to these common features are made in a centralized manner, reducing redundancy and enhancing the efficiency of managing the product line.

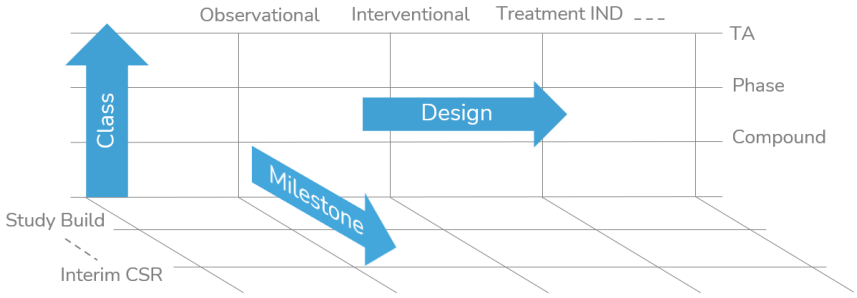


Figure 3 Features are managed by category.

By having a set of common managed features, organizations can streamline the development, testing, and maintenance processes. It allows for efficient updates and improvements since changes made to the common features automatically apply to all products within the product line. This approach promotes reuse, reduces duplication of efforts, and facilitates a more systematic and scalable development process in the context of software product lines.

When applied to Clinical Trials, common features span data collection (CRF in and EDC, Randomization, Lab data, Real-World data sources, etc.) through to data collections formats such as CDASH and SDTM domains, Safety reporting, ADaM analysis datasets, Biostatistical and regulatory reporting, etc.

DEVELOPED FROM A COMMON SET OF CORE ASSETS

A common asset base refers to a set of shared and reusable assets that serve as the foundation for developing a family of related software products. These assets typically include code, components, protocols, study designs, documentation, and other artifacts that encapsulate common features as depicted below.

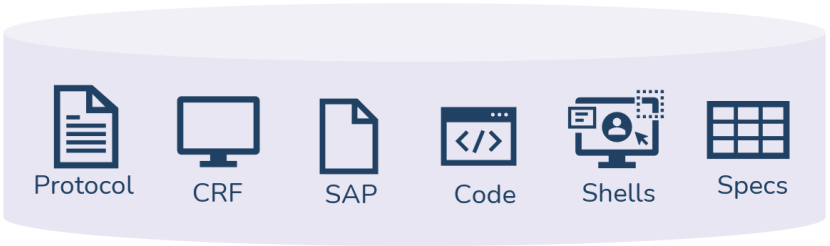


Figure 4 Typical Clinical Trial Common Asset Base

The common asset base is a key concept in SPL methodologies, emphasizing the importance of systematic reuse to achieve efficiency, consistency, and maintainability in software development. Instead of building each software product from scratch, developers leverage the common asset base to reduce redundancy and promote a more streamlined development process.

Establishing a robust common asset base in SPL enables organizations to capitalize on economies of scale, reduce development time, and improve the maintainability of their software products. It fosters a systematic approach to software development, where commonalities are identified and encapsulated, and variabilities are managed effectively to accommodate the specific needs of individual products within the product line.

WORKING IN A PRESCRIBED WAY

Production plans are used to guide and facilitate the systematic production of specific trials within the product line. A production plan outlines the configuration of features and components that should be included in a particular Trial variant. These plans are instrumental in managing the variability and customizability inherent in a product line.

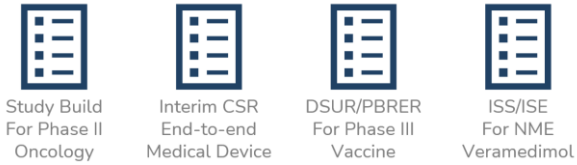


Figure 5 Production plans are used to configure specific trials.

Production plans consist of both machine-readable metadata that is used to automate the configuration of features and components – for example configuring the CRF for the study build features of a trial. The Production Plan will typically also consist of human-readable documentation that is used for the manual configuration of the trial – for example a checklist for unblinding the study, etc.

STANDARD ARCHITECTURE

A standard architecture refers to a predefined and commonly accepted architectural blueprint that serves as a foundation for designing and implementing trials within the product line.

This standard architecture encapsulates the commonalities shared among the various trials, providing a consistent and reusable structure for development.

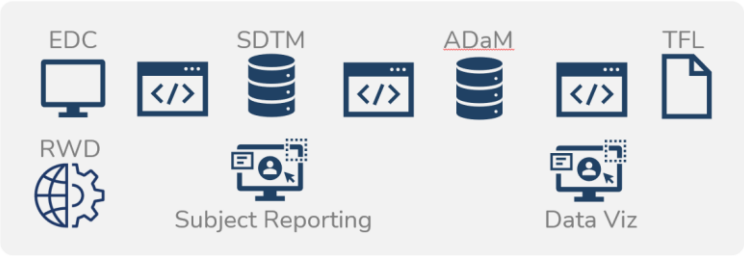
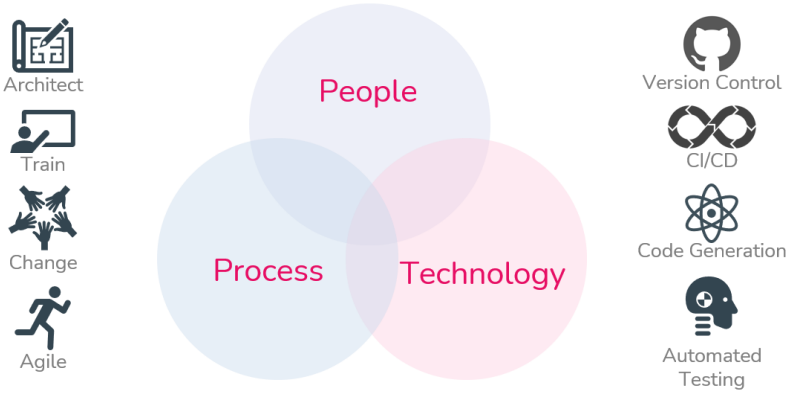


Figure 6 Simplified standard clinical trial architecture.

A standard architecture is crucial for achieving efficiency, consistency, and maintainability across a family of related products. It supports the core principles of SPL, such as variability management and feature reuse, and contributes to the overall success of the product line engineering approach.

HIGH PERFORMANCE TEAM

A high-performance team combines people with specific skills and a deep understanding of the clinical trial domain, combined with a process-driven approach to trial delivery and the tools and technology needed to rapidly assemble and manage a trial from common features and components.



Such a high-performing team is crucial for several reasons, as it directly contributes to the success of the product line approach:

- **Efficient Collaboration:** Software product lines involve the development of multiple related products with shared and varying features. A high-performance team excels in communication, collaboration, and coordination, ensuring that team members can work seamlessly together. This efficiency is essential for managing the complexity of product line development.
- **Rapid Development and Iteration:** High-performance teams are often more adept at delivering results quickly. In the context of software product lines, where multiple products may need to be developed or updated simultaneously, the ability to iterate rapidly is crucial to meet evolving market demands and stay competitive.
- **Adaptability to Change:** The software industry is dynamic, and requirements can change rapidly. A high-performance team is more adaptable to changes in project scope, shifting priorities, or evolving customer needs. This flexibility is vital in the context of software product lines, where managing variability and accommodating changes efficiently are key challenges.
- **Expertise in Domain and Technologies:** SPL development often involves dealing with complex domains and technologies. A high-performance team is well-versed in the specific domain of the product line and has expertise in relevant technologies. This knowledge enables the team to make informed decisions, design robust architectures, and implement high-quality solutions.
- **Effective Variability Management:** Variability management is a core aspect of software product lines. A high-performance team is skilled in designing and implementing features in a way that efficiently manages variability. This includes creating reusable components, defining clear interfaces, and making decisions that enable easy configuration and adaptation of products within the line.
- **Quality Assurance and Testing:** High-performance teams prioritize and excel in quality assurance and testing. In the context of software product lines, where multiple product variants need to be tested, ensuring the reliability and correctness of features is paramount. A high-performance team implements robust testing strategies to maintain high product quality.
- **Leadership and Decision-Making:** Successful product line development requires effective leadership and decision-making. A high-performance team is characterized by strong leadership, where decision-makers can guide the team through complex challenges, make informed choices about feature inclusion or exclusion, and set a strategic direction for the product line.
- **Employee Satisfaction and Retention:** High-performance teams often have satisfied and motivated team members. Employee satisfaction and retention are crucial in the long-term success of a software product line, as experienced and engaged team members contribute to the institutional knowledge and continuity of the product line.

In summary, a high-performance team is essential for navigating the challenges associated with software product lines. The ability to collaborate efficiently, adapt to change, manage variability effectively, and deliver high-quality products in a timely manner are all critical factors that contribute to the success of an SPL initiative.

REFERENCES

[1] What is CDSIC 360. https://www.cdisc.org/cdisc-360

[2] A Framework for Software Product Line Practice, Version 5.0
https://insights.sei.cmu.edu/library/a-framework-for-software-product-line-practice-version-50/

[3] Feature-Driven Development
https://en.wikipedia.org/wiki/Feature-driven_development

CONTACT INFORMATION

(In case a reader wants to get in touch with you, please put your contact information at the end of the paper.)
Your comments and questions are valued and encouraged. Contact the author at:

Stuart Malcolm
Head of Standards, Efficiency and Automation
Veramed
Stuart.Malcolm@Veramed.co.uk
http://www.veramedinc.com