Enabling Verifiable Conformance for Product Lines

Robyn Lutz
Jet Propulsion Laboratory/Caltech and Iowa State University
rlutz@cs.iastate.edu

Abstract

NASA is, with the rest of industry, turning to product-line engineering to reduce costs and improve quality by effectively managing reuse. Experience in industry has shown that it is the verifiable conformance of each system to the product-line specifications that makes or breaks the product-line practice. Verification that the software for each project satisfies its intended product-line constraints is thus essential. This paper reports early results from an effort to assemble from previous, industrial experience a set of enablers of verifiable conformance for use in the application engineering of NASA product lines. Lessons learned may be useful for developers of safety-critical, long-lived, or highly autonomous product lines, as well as for companies that integrate product line subsystems developed by multiple contractors.

1. Introduction

Product-line engineering of NASA systems offers the opportunity for significant cost savings and increased quality control. In product-line engineering, assets such as a common architecture, shared requirements, component implementations, and product-line test suites are reused to build each new system in the product line. Reuse of domain-engineered, product-line assets can reduce the cost and time to market of new systems, and can improve the quality of the developed products.

However, with this opportunity come new verification challenges. Experience in industry has shown that it is the *verifiable conformance of each new product to the product-line specifications* that makes or breaks the product-line practice. Verification that the software for each project satisfies its intended product-line constraints is thus essential. Specifically, in order to be successful in adopting product-line engineering at NASA we need to answer two questions:

1. How can we verify that delivered software conforms to the product-line requirements and architecture levied on it, and how do we document that conformance?
2. How can we enable such verification throughout the development lifecycle?

This paper reports results from experience assembling a set of enablers of verifiable conformance for use in the application engineering of NASA product lines. These techniques are intended to enable, or facilitate, both the conformance of new systems in the product line to the product-line specifications and the verification of such conformance.

Since a broad variety of product-line engineering solutions exists and has been applied successfully in industry, our first step was to extract and customize to NASA needs the problems and solutions relevant to verification reported in previous, industrial experience with product lines. The reported challenges from previous industrial experience were appropriately seen by management as risks for which their product-line plans would have to provide mitigation strategies.

We report here lessons learned from our effort to extract and formulate for NASA management previous lessons learned. The paper integrates two kinds of experience: experience reported in the literature and our experience trying to fit these reported accounts to NASA's planned missions. We found that the second-hand knowledge provided a fairly rich source of information about challenges to verifiable conformance that other organizations encountered while building product lines. However, the effort to fit previous experience to NASA needs identified some gaps in reports to date. We focus here on three of these issues that are especially significant for verifying conformance in high-integrity product lines:

- Many NASA product lines will be *safety or mission critical.* Although most consumer product lines are not safety-critical, a growing number are. Examples include embedded medical devices, avionics platforms, flight instrumentation displays, and medical imaging systems [21, 23, 28].
- Many NASA product lines will be *long-lived.* Individual systems will have lifetimes measured in decades due to the time required to reach other planets or maintain habitable bases on the Moon. While most commercial product lines are not long-lived, adopting the structured evolution and change management techniques needed for the longer NASA missions also has benefits for maintaining shorter-lived product lines.

978-0-7695-3303-2/08 $25.00 © 2008 IEEE
DOI 10.1109/SPLC.2008.12

- Extended and remote NASA missions will be *highly autonomous* [2, 8]. Autonomy allows the system to react quickly to failures and unexpected changes in the environment even when a human controller is unavailable. Many commercial, autonomous product lines are also being built. Robotic applications, for example, are among the most rapidly growing product lines (see, e.g., [18]). They also benefit from improved techniques to confirm compliance of autonomous behavior in a product with associated product-line requirements.

The work reported here differs from previous work in focusing on the need to verify conformance of each new system in the product line and to demonstrate evidence of that conformance. With the growing number of high-integrity product lines in avionics, automotive, medical, power, and robotics industries, many of the same challenges to verifiable conformance identified here for NASA product lines will be faced by other organizations. In such product lines, not verifying that the safety, dependability, performance, and/or reliability requirements are met for each new product may cause damage to life, health, or property.

The rest of this paper is organized as follows. Section 2 gives background information and describes the four product-line applications against which the reported techniques were judged. Section 3 identifies the documented challenges to verifiable conformance. Section 4 describes the application engineering techniques we recommended to NASA for enabling verifiable conformance of software product lines with descriptions of related work. Section 5 reviews positive and negative results from the effort. Section 6 provides concluding remarks and open problems.

2. Background and Applications

2.1 Background

A software product line is defined to be "a set of software-intensive systems sharing a common, managed set of features that satisfy the particular needs of a specific market segment or mission and that are developed from a common set of core assets in a prescribed way" [6]. We use the SEI definition of conformance in their open-systems glossary as "action or behavior in correspondence with current customs, rules, or styles" [29]. The product-line specifications and assets are the "customs, rules, or styles" to which the new product being built in the product line must verifiably conform.

Product-line development is typically divided into two phases: domain engineering, in which the product-line assets are specified and developed, and application engineering, in which the product line assets are reused to build each new system in the product line [32]. The first phase, domain engineering, depends on the knowledge and skill of domain experts to produce a set of optimized product-line assets. These typically include a product-line architecture and a decision model that specifies both the software requirements common to all systems in the product line, and the variations or ways in which the systems will differ. Most of the experience reported to date both inside and beyond NASA has been on effective domain engineering of product lines, both because it comes first and because correct scoping and specification of the product line and its architecture is basic to its success.

In the second phase, application engineering, the product-line assets, such as a common architecture and shared requirements, are reused to build each new system in that product line. It is this second phase that will determine the success of the product-line approach for NASA because reuse assumes the conformance of each individual project to the product-line constraints previously levied on it. This paper thus focuses on application engineering, specifically on the verification techniques that will be needed for application engineering of NASA software systems built using product-line assets. This work is a step toward the goal of developing a baseline verification process to show that the software in each system in the product line is, in fact, product-line compliant.

2.2 Applications

We draw on information from four product lines or product-line-like sets of systems, three from Jet Propulsion Laboratory (JPL) and one from NASA. Some of these systems are not yet product lines according to the SEI's scoping guidelines. This is because their assets are not centrally owned but are used in a "clone and own" fashion by multiple projects [26]. However, they do meet the level of system maturity for software product lines defined in Jaring, Krikhaar and Bosch: "functionality provided by the platform is increased to the level where functionality common to several but not all products becomes part of the shared artifacts. Product specific functionality is introduced by adapting the products after instantiating them from the product line" [16]. We use the more-inclusive definition here because it was the product-line aspects that were of interest to management in this exploratory study. The four product lines are:

- An interferometer (spaceborne telescope) software product line developed at JPL, with reused

architecture, documentation, and code. Fig. 1 shows how product-line assets were used in

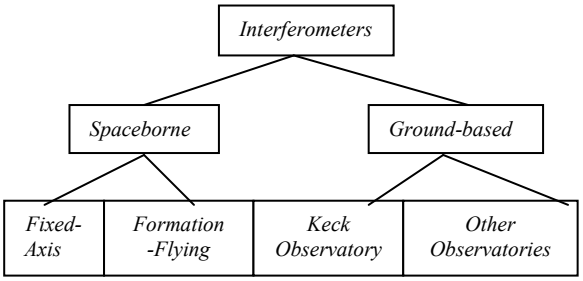

Figure 1 Interferometer Product Line

multiple interferometers with plans for additional missions. The project did not self-describe their work as product-line engineering, and we found in a previous study that because it was not originally designed as a product line, the actual architecture gave evidence of evolution away from the documented architecture [22].

- TechSat21, a proposed NASA mission originally scheduled for a 2006 launch, but later cancelled. Much of the software was reused on a subsequent mission. TechSat21 is a cluster of cooperating, context-aware, microsatellites [8].
- A multi-mission hardware and software architecture to provide JPL missions with a standard set of hardware and software components that can be adapted and customized to fit mission-specific needs. The determination as to use/adaptation of the product-line assets is currently made independently by each customer (i.e., project).
- A JPL ground data system for spacecraft that provides a set of common software services and tools to multiple missions for use during operations (e.g., generating commands to send to spacecraft, processing downlinked telemetry data, mission planning, etc.).

Interest in product lines has grown at NASA in response to the needs of the Constellation Program. Constellation is the NASA program to send human-flight missions to the Moon and Mars. David Atkinson, the program manager for JPL's Exploration Systems Engineering, has described reuse as one of Constellation's "key objectives" [2]. He stated in 2006, "The best practices for large-scale reuse have been captured and defined into a practice called software product lines." He also stated that "the challenges for reliably delivering safe and useful capabilities in Constellation are profound." The work reported in this paper investigates one small piece of these challenges, that of assuring verifiable conformance during application engineering.

3. Challenges to Verifiable Conformance

This section describes challenges documented in the open literature that NASA will be likely to face in verifying the conformance of a new system to product-line assets. It also describes the strategies proposed by experts in the application engineering of product lines to address these challenges through improved organization or process. Due to space limitations, only those challenges most relevant to high-integrity, long-lived or autonomous product lines are discussed here.

The approach used in identifying these challenges was to survey publicly available accounts of previous industrial experience with product line engineering in workshops, conferences, and reports. The survey was broad but not comprehensive due to time constraints related to when the product-line project management was scheduled to describe the product line to upper management. Relevance to NASA was determined informally and was based on the author's experience with product lines at JPL, discussions with product-line experts and advocates both inside and outside NASA, and discussions with project personnel (managers and developers) both amenable to and reluctant to consider product-line engineering.

Verification of product lines differs from verification of single systems: (1) in the greater opportunities for reuse of the product-line framework and product-line assets, including requirements specifications, models, code, and test cases, with potential attendant cost-savings and improved quality, and (2) in the added complexity of developing and supporting multiple variants, multiple configurations for different customers, and multiple component versions across differing product lifecycles.

The verification performed during application engineering depends on accurate and thorough verification during domain engineering. The better the verification of the product-line assets at the time they were developed, the easier the verification at the time they are reused. Since conformance of a new product to a defective product-line specification is clearly not desirable, use of inadequately verified assets could quickly erode the advantages of a product-line approach.

3.1 Organizational: asset ownership

A major consideration for verification of NASA product lines is how the development organization will be structured. The challenge here is how best to meet

each project's needs using the product-line assets. Weiss and Lai, in describing the FAST process, make the point that for large, complex product lines, decomposition into subdomains may be the answer [32]. They give the example of an organization that might have a device subdomain, a display subdomain, a database subdomain, and so on. Each member of the product line is built by building a member from each subdomain and then integrating the members. Each subdomain thus forms a product line, and the systems built form a product line of product lines. The organization responsible for integrating the domain members remains distinct from the organization for each domain. It seems likely that ownership of the product-line assets on NASA product lines (e.g., central domain ownership versus clone-and-own) will be determined by the structure of the development and integration organization.

3.2 Cultural: reluctance to invest for other projects

Frakes and Kang describe a challenge faced during application engineering when developers of different products in the product line embrace reuse without a centralized product-line set of assets. They note that "There is a danger that projects may be willing to use other's products, but will be reluctant to make investments for others" [11]. One such investment is to put in place a mechanism by which lessons learned by projects regarding use of the product-line assets are communicated back to the asset owner and forward to other projects. This is another way in which an organization can impede product-line development and verification.

3.3 Requirements: controlling the delta

Inevitably, specific applications will need additional or alternative features not initially provided in the product-line assets. Such adaptations may occur at the requirements, design, or implementation level. For example, at the requirements level, a new feature may be required. At the design level, an alternative design mechanism may be selected. At the implementation level, new coding standards may drive change. Each of these changes has the potential to change the product-line. Some changes may also drive up the cost of the new system.

The decision as to whether to incorporate the changes into the product-line assets and how broadly to disseminate the changes (e.g., whether to update other products similarly) will be difficult and important. The project's Change Control Board for the specific system considering such adaptations will probably be the first line of defense in maintaining the product-line assets and in judging which proposed changes merit moving away from the product-line standard set of verified components.

Some threats to verification that evolution can entail are described in [30]. Not folding evolutionary changes made to meet the requirements of a new system back into the other systems in the product line keeps the cost of evolution down, but can over time lead to degradation of the product line and the erosion of commonality. On the other hand, propagating each change to the product line can make the re-verification costs unmanageable.

3.4 Contractors: motivating conformance

Development or provision of subsystems or systems by external contractors or suppliers to NASA is common. When such components either belong to a product line or are to be integrated into a product line in-house, incentives for contractor conformance to product-line specifications are needed but sometimes difficult to implement. Customers may require the use of assets that are not optimal from a contracting developer's point of view [7]. On the other hand, project management responsible for acquiring and integrating supplied software often expresses concern regarding their limited control and visibility into details of the contractors' development processes.

4. Verifying Conformance of New Products

In this section we describe the product-line application engineering techniques we recommended to NASA for enabling verifiable conformance of software product lines with pointers to related work. The focus is on how NASA can more readily assure that a new system (developed by NASA, an external supplier, or both) conforms to the product-line requirements levied on it. The criteria used to recommend techniques were: address challenges described in Sect. 3, successful use in industry, relevance to application engineering, and tailorable to NASA high-integrity needs (safety-critical, long-lived, highly autonomous).

We distinguish *conformance tasks* at each development phase from *verification tasks*, although the techniques are similar. Conformance tasks for the new product check whether its development at each phase (requirements, architecture, design, code, testing) conforms to the product-line's constraints. On the other hand, verification tasks for the new product

check whether its development at each phase is consistent with its own previous phase.

4.1 Requirements conformance

Verifying whether the requirements for the new system conform to the product-line requirements can only be accomplished if adequate documentation exists from the domain engineering of the product line. The specification of the feature model underpins all subsequent conformance analysis. For example, the FAST process, which pays careful attention to the conformance of the final new system to its product-line specifications, has an item in the Analysis List for the Application Engineer role: "FinalProduct_ Validation_Analysis: Check whether all the decisions made in the application model exist in the final product" [32]. Both Krueger [20] and Weiss and Lai recommend an application modeling language and tool-supported environment to enable auto-generation of code and documentation for as much of the new member as can be auto-generated with manual development of any customized portions.

Because many of NASA's missions involve the use of emerging technologies or operational space environments that are only partially understood, product-line evolution will be highly likely. In some cases, the product-line assets may not satisfy all the requirements for the new system. Early assessment of the delta enables an evaluation of whether it is feasible to add the missing features on top of the product-line assets [13]. Planning for product-line evolution entails managing change such that conformance (or non-conformance) to the product-line specifications can be demonstrated.

4. 2 Safety requirements conformance

Verification that the software requirements satisfy the safety requirements for the system uses results from the preliminary hazard analysis, including software fault tree analysis (SFTA) and software failure mode, effects and criticality analysis (SFMECA) performed on the product line. SFTA can be performed at the product-line level and reused, via tool-supported pruning to exclude fault paths not relevant to the new system's choice of features [9]. Bi-directional product-line safety analysis, which combines a forward analysis (from failure modes to effects) with a backward analysis (from hazards to contributing causes) has also been useful in finding missing and incorrect software safety requirements [10].

4.3 Architectural conformance

Verification that the software architecture for the new system conforms to the product-line reference architecture traces the architectural elements in the new system back to their abstract elements in the product-line architecture. Clements and Northrop have urged that "conformance rules" be put in place as part of the domain engineering activities in order to ensure that the products in the product line conform to the architecture [6]. While their concern is to avoid the degeneration of the core assets rather than to enable verification of conformance during application engineering, such rules can facilitate checks for conformance. Muccini and van der Hoek describe the use of conformance testing of software architecture for product lines [25].

If changes have been made to the reference architecture for the new system, perhaps in response to new requirements, they must be carefully verified. Some changes are local while others have cross-cutting effects (to several components) on the architecture.

4.4 Design conformance

Verification that the detailed design for the new system conforms to the product-line components' detailed design is enabled by traceability from the product-line requirements to the product-line component design. Verification that the design of the new product satisfies the product-line requirements may be assisted by modeling of the new system. Model-driven development of product lines supports an incremental approach to the development of the new application since the common (or kernel) features can be included first, followed by the optional features already modeled in the product line, followed by any components that need to be added in response to the requirements for this new system.

For critical applications, such as human-flight or mission-critical systems, safety-related scenarios derived from the hazard analysis can exercise state-based models of product-line components in order to verify that the as-designed behavior is safe [21]. The process of verifying that the new product's design complies with the product-line design builds on the verification that the new product's requirements comply with the product line's specified requirements.

The design verification is simpler when the product-line scoping has been done effectively (i.e., the crystal ball has accurately foreseen future needs). In that case, the options already available in the product-line assets may suffice. If customer needs cause

changes to the product-line components then the verification effort is increased [31].

4.5 Implementation conformance

Verification that the implementation of the new system conforms to the product-line involves checking that the selected product-line assets (component code files, interfaces and parameters, configurations, libraries, and databases) are consistent with the current configuration of those product-line assets and are integrated correctly [6]. This helps provide assurance that the configurations possible in the new system are consistent with the configuration management of the product line.

Analysis of problem reports is important both because they may describe problems that can affect other product-line systems and because they help locate undocumented non-conformances between the new system and the product line. Similarly, Zelkowitz and Rus's work to describe defect analysis across the multiple releases of the Space Shuttle's flight control software confirms the importance of testing each new product in the product line for change [33].

4.6 Testing conformance

Conformance testing "tries to test the extent to which system behavior conforms to its specifications"[4]. It establishes a clear link between the requirements specifications, the test cases generated or derived from them, and the verification goal. Conformance testing may be mandated by a regulatory agency and/or performed by an independent testing organization.

Testing of product lines is a mature field, see, e.g., [3, 19] with its own workshop, SPLiT. Based on the previous verification steps, testing of the new system validates both that it meets its own requirements and that it conforms to the product-line requirements. McGregor provides a good introduction to the testing of product lines that emphasizes traceability to the software architecture [24]. The test plan identifies the scope for reuse of the product-line test suite and identifies new or changed features for which new tests to be developed. The test plan also identifies new or changed domain environments which may affect reuse of the product-line test suites. For independent verification of conformance, Knauber and Hettrick assign validation in product lines to an independent quality organization [19]. For acceptance testing, Geppert, Li, Rößler, and Weiss have reported success in using the decision model to generate acceptance test cases. In an application to parts of a legacy acceptance test suite of a large product line they found that using seven parameters to generalize the test cases reduced the duplication effort by about 85% [12].

4.7 Conformance of acquired product lines

Many large NASA product lines or their components, including Constellation, will be built primarily by contractors and subcontractors rather than in-house. On the other hand, most product-line experience reports describe situations in which the domain engineers and the application engineers work together. The SEI's guidebook on software product line acquisition notes that testing is performed on evolving products, not simply the final product, and that results from these tests provide an objective basis to accept the product or not [30]. The document emphasizes the importance of spelling out in the supplier agreements the content of the product-line verification process (including early testing, conformance with product-line specifications, and delivery of product-line testing assets).

5. Discussion of Results

In this section we review positive and negative results from the effort to extract and formulate for JPL and NASA lessons learned from previous, industrial experience regarding verifiable conformance of each new product to its product line.

Table 1 shows an excerpt from the results presented to management. A tabular format was preferred with supporting detail available textually. The left column lists the category and mapped to risk categories familiar to the projects. The middle column describes the challenge itself (from Sect. 3). In the reports produced for management, reference links were also given to the original experience report and to additional information. The right hand column describes the techniques by which the challenge to verifiable conformance might be resolved (see Sect. 4). Sometimes this technique was recommended in the same experience report describing the challenge; other times it was based on previous experience or other readings on product lines.

The managers felt that existing, published business cases for building product lines did not readily offer sufficient insights into the conformance be risks involved, especially when building product lines for critical missions. That initial observation was largely confirmed in this study. The good news is that the product-line community has tended to relatively forthcoming in the open literature about the difficulties of implementing product-line engineering (e.g., [7]).

Table 1 Challenges and techniques for verifiable conformance of a product line

Categories	Challenges	Techniques
Organizational	• Ownership of assets unclear • Structure doesn't match project needs	• Divide into subdomains • Consider separate integration organization • Rigorous two-way traceability
Cultural	• Reluctance to invest for other projects • Feedback from user to owner of assets fails	• Create conscious PL culture • Learn from experience and problem reports
Requirements	• Products needs new features • Time can erode product line	• Establish strong change board to judge tradeoffs • Capture requirements delta early for new projects • Invest in tool support for change management
Contractor/supplier	• Low incentive for conformance to product-line specifications • Less control than in-house	• Motivate/reward conformance to product-line specifications • Include source code in supplier agreements

The bad news is that, perhaps due to proprietary concerns, failures of product-line applications are not described in enough detail to avoid those same risks and failures. Negative accounts of product-line application engineering tend not to provide enough information for systematic mining of insights into how a breakdown of the mapping from product-line assets to product can occur and how it can be practically detected at each stage of the lifecycle. What is missing is careful analysis of near misses and accidents of product-line systems.

A second negative result is that even experience reports describing high-integrity product lines did not describe their critical aspects or described them only in relation to the domain engineering of the software architecture. For example, the CAAS case study described the product-line avionics software architecture used for the Army's helicopters. It discussed reliability in the context of the domain-engineered architecture but not in application engineering [5]. Similarly, Jaring, Krikhaar and Bosch's description of a product line of MRI scanners did not discuss the safety-related issues or requirements [16]. More studies of high-performance product lines are also needed. For example, in previous work we found that high-performance requirements on one product (e.g., picometer metrology and microarcsecond astrometry) constrained product-line decisions [22].

5.1 Key enablers for product-line conformance

Our study identified several techniques that facilitate verification that a product conforms to the product-line constraints. In this section we describe those that appeared to be most relevant to safety-critical, long-lived, and/or highly autonomous product lines. We call these "key enablers" to indicate their practical impact.

A key enabler for *establishing verifiable conformance of safety requirements* in the new product is rigorous traceability to the product-line safety requirements. Traceability from the domain-engineered, product-line hazards analysis and software safety requirements to the application-engineered product's derived hazards analysis and derived software safety requirements supports checks for product-line compliance. For example, for one product-line system we found via model-checking that outdated data from an expired source could sometimes incorrectly be used to calculate the target (for pointing the telescope) [22].

Another enabler for verifiable conformance in safety-critical product lines is to start small [20]. For example, Weiss and Lai recommend applying product-line engineering first to an isolatable section of a legacy system with frequent changes at high cost. The reason is that in this case changes can be encapsulated with a single group of software developers responsible for the software [32]. Planning to start simply with a loosely coupled subsystem in a single domain and a small, cohesive team of developers supports both the development and demonstration of verifiable conformance for a safety-critical product line. However, this recommendation may be controversial in that it

conflicts with the need to think big to gain institutional support and funding.

A key enabler for *maintaining conformance for long-lived missions* is accurately capturing the requirements delta information in the documentation of the application's requirements specification. This allows early identification of features that are not currently in the product-line assets and of previously unplanned and unstudied configurations. For example, one interferometer needed to have a new pathlength feedforward capability, while another needed a different fringe-search algorithm (for detecting planets around suns).

A key enabler for *verifying conformance of highly autonomous product lines* such as those planned at NASA is good documentation of the product-line behavior, preferably as a machine-readable model allowing automated checking. For example, in one new product we found that non-adjacent layers in a layered architecture could now communicate with each other, an unintended side effect of a new software requirement. In addition, access to source code for the product-line components may be an issue for future NASA missions which depend on supplier-provided software. Pohl, Bockle, and van der Linden point out that, since any executable evaluation copy contains bound variants, additional artifacts such as source code and compilation and linking instructions may need to be available in order to adequately verify all the possible behaviors [27].

Another enabler for verifiable conformance of autonomous product lines may be to use small configuration units. For example, Rockwell Collins found, in building a large product line, that a finer grained decomposition of functionality led to higher levels of reuse and systems that were easier to test and get through airworthiness qualifications [5]. This was despite the fact that more configuration items were thus produced to integrate and maintain. This confirmed a finding from previous work on a product line at JPL which found that the unit of reuse was small although the number of units reused was large [22].

5.2 Evidence of conformance

For safety-critical product lines it may be necessary not only to verify conformance of the new product to the product-line specification, but also to demonstrate evidence that it conforms. In such cases assurance that a new system in a product line conforms to the product-line requirements that it was intended to satisfy entails assembling evidence to indicate that conformance is achieved [15].

NASA has recently been investigating whether techniques involved in safety or dependability cases can assist in assuring that their critical systems are safe. The structure of a safety case supports the type of structured reasoning about the product-line conformance of a critical system that will be required for human-rated missions to the Moon.

A safety case typically consists of three main parts: claims, evidence, and arguments that link the claims to the evidence [1, 17]. Claims are safety requirements that the system must satisfy. Evidence is information that helps demonstrate that the safety requirements are met. Arguments provide a chain of reasoning to show how the evidence relates to the claims. The safety case documents traceability from the hazard analysis to the safety requirements to the design and implementation.

Habli and Kelly have described the construction of a safety case for a product line, the Aerospace Engine Monitoring Units (EMU) at Rolls Royce [14]. The determination in that case was that the product line software was classified as PDS (Previously Developed Software). Similarly Clements and Bergey reported for an avionics product-line for helicopters that "Airworthiness qualifications happen in a much shorter time" with product-line engineering [5]. Enabling verifiable conformance for product lines can simplify assembly of evidence for future product-line safety or dependability cases.

6. Conclusion and Future Work

The experience reported in this paper was an effort to identify from previous industrial experience the enablers for verifiable conformance to recommend for use in the application engineering of NASA product lines. Experience with four such applications with plans for ongoing, future use guided the evaluation. The work was motivated by project management's concern that existing, published business cases for building product lines did not readily offer sufficient insights into the conformance risks involved, especially when building product lines for critical missions.

Results reported here show that, although the product-line community has provided a rich source of information about techniques to address verification challenges encountered while building product lines, there is relatively little information regarding product-line conformance for high-integrity product lines. We identified three areas in which additional experience reports are needed. In all three areas product lines are currently being built but information

about the application engineering of the products is limited. These open problems are:

1. *Verification experiences on safety-critical product lines.* There is a lack of information about practical experience with verification of safety requirements during application engineering of individual products. Descriptions of high-integrity product lines tend to focus on domain-engineering appropriate architectures (which is critically important) but to ignore the challenges of verifying the safety and reliability of each new system built in the product line. Experience reports on how the conformance of safety-critical systems to the product-line specifications was verified would be very useful both to identify risks in such endeavors and to validate proposed verification techniques.
2. *Evolution management experiences on long-lived systems*. Planning for product-line evolution entails managing change such that conformance (or non-conformance) to the product-line specifications can be demonstrated. Rigorous specifications (e.g., using a domain-specific language) and automatic compilation seems to be one practical evolution management technique for long-lived product lines [19,32]. For development environments which do not include such rigorous specifications (e.g., use natural-language specifications), experience reports are needed regarding which product-line engineering practices have, during application engineering, made it easier to verify that a long-lived system continues to conform to the product-line specifications. This is important because analyses done on the product-line are only reusable (i.e., applicable) to the individual system if the assumption that the system conforms to the product-line continues to be valid.
3. *Experience with autonomous, adaptive and reconfigurable product lines.* Autonomous systems bring additional verification challenges that can complicate the handling of variabilities in product lines [8]. Many autonomous product lines are currently being built, with robotic product lines experiencing rapid growth [18]. However, "autonomy" in the product-line literature tends to refer only to organizational teams rather than to software functionality. How to verify during application engineering that all possible behaviors of an autonomous or adaptive system are within the envelope of behaviors specified for its product line is an unmet challenge. Experience-based information about how practitioners address this challenge to verifiable conformance in a product-line setting would be a useful contribution.

Verifiable conformance that the software for each new product satisfies its intended product-line constraints is essential to the success of product lines. Experience described here identified product-line practices that enable not just the conformance of new systems to the product line specifications, but the verification and demonstration of such conformance. We also found that additional empirical data is needed from the application engineering of safety-critical, long-lived, and autonomous product lines to provide a better understanding of how to achieve verifiable conformance in these systems.

7. Acknowledgements

The author thanks Tom Hoffman, Thomas Fouser and Kenneth Meyer for useful discussions about product lines. The research described in this paper was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautic and Space Administration and funded by NASA's OSMA Software Assurance Research Program. The author's research is supported in part by National Science Foundation grant 0541163.

8. References

[1] *Adelard Safety Case Development Manual*, Adelard, 2008.

[2] D. J. Atkinson, "Constellation Program Return to the Moon: Software Systems Challenges—Autonomy and Autonomicity a Solution?", *Proc. 4th IEEE In'tl Wkshp Eng Autonomic and Autonomous Systems*, 2007, 172-175.

[3] A. Bertolino and S. Gnesi, "Use Case-Based Testing of Product Lines", *Proc. 9th ESEC*, 2003, pp. 355-358.

[4] E. Brinksma, "Formal Methods for Conformance Testing: Theory Can Be Practical", *Proc. CAV*, 1999. LNCS 1633, pp. 44-45.

[5] Clements, P. and J. Bergey, *The U.S. Army's Common Architectural Avionics System (CAAS) Product Line: A Case Study*, CMU/SEI-2005-TR-019.

[6] Clements, P. and L. Northrop, *Software Product Lines*, Boston, Addison-Wesley, 2002.

[7] S. Deelstra, M. Sinnema and J. Bosch, "Experiences in Software Product Families: Problems and Issues during Product Derivation", *Proc. SPLC* , 2004, pp. 165-182.

[8] J. Dehlinger and R. Lutz, "A Product-Line Approach to Promote Asset Reuse in Multi-Agent Systems", *SELMAS IV*, LNCS 3914, pp.161-178, 2006.

[9] J. Dehlinger and R. Lutz, "PLFaultCat: A Product-Line Software Fault Tree Analysis Tool", *Automated Software Engineering,* 13(1), 2006, pp. 169-193.

[10] Q. Feng, and R. Lutz, "Bi-Directional Safety Analysis of Product Lines", *JSS,* 78(2), Nov., 2005, pp. 111-127.

[11] W. B. Frakes and K. Kang, "Software Reuse Research: Status and Future", *TSE* 31(7), July 2005, pp. 529-536.

[12] B. Geppert, J. Li, F. Rößler, and D. M. Weiss, "Towards Generating Acceptance Tests for Product Lines", *Proc. ICSR* 2004, LNCS 3107, pp. 35–48.

[13] Gomaa, H. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software* Architecture, Addison-Wesley, 2005.

[14] I. Habli and T. Kelly, "Challenges of Establishing a Software Product Line for an Aerospace Engine Monitoring System", *Proc. SPLC* 2007, pp. 193 – 202

[15] Jackson, D., M. Thomas and L. I. Millett, eds., *Software for Dependable Systems: Sufficient Evidence?* NRC, National Academies Press, 2007

[16] M. Jaring, R. L. Krikhaar and J. Bosch, "Representing Variability in a Family of MRI Scanners", *Software* Practice and Experience, 2004, pp. 69-100.

[17] T. P. Kelly and R. A. Weaver, "The Goal Structuring Notation-a Safety Argument Notation", *DSN Workshop on* Assurance Cases, 2004.

[18] M. Kim, S. Kim, S. Park, M-T. Choi, M. Kim and H. Gomaa, "UML-based Service Robot Software Development: a Case Study", *ICSE,* 2006, pp. 534-543.

[19] P. Knauber and W. Hetrick, "Product Line Testing and Product Line Development—Variations on a Common Theme", *Proc. SPLiT,* 2005.

[20] C. W. Krueger, "New Methods in Software Product Line Practice", *CACM*, 49 (12), Dec., 2006, pp. 37-40.

[21] J. Liu, J. Dehlinger, and R. Lutz, "Safety Analysis of Software Product Lines Using State-Based Modeling", *JSS* 80 (11), 2007, pp. 1879-1892.

[22] R. Lutz and G. Gannod, "Analysis of a Software Product Line Architecture: An Experience Report," *JSS* 66(3), 2003, pp. 253-267.

[23] D. McComas, S. Leake, M. Stark, M. Morisio, G. Travassos and M. White, "Addressing variability in a guidance, navigation, and control flight software product line*", Proc*. *SPLC PL Architecture Workshop*, 2000.

[24] McGregor, J. D. *Testing a Software Product Line,* CMU/SEI SEI-2001-TR-022.

[25] H. Muccini and A. van der Hoek, "Towards Testing Product Line Architectures", *TACoS'03, Elec Notes in TCS* 82(6), Sept. 2003, pp. 99-109.

[26] Northrup, L. P. Clements, et al., *A Framework for Software Product Line Practice*, v. 5.0, 2007, SEI.
.

[27] Pohl, K., G. Bockle and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*, Springer, 2005.

[28] R. Schwanke and R. Lutz, "Experience with Architectural Design of a Modest Product Family", *Software Pract and Exper*, 34(13), 2004, pp. 1273-1296.

[29] SEI, *Open Systems Glossary*, http://www.sei.cmu.edu/opensystems/glossary.html.

[30] *Software Product Line Acquisition: A Companion to a Framework for Software Product Line Practice*, 2004. http://www.sei.cmu.edu/productlines/companion.html

[31] R. van Ommering, "Software Reuse in Product Populations", *IEEE TSE*, 31(7), July 2005, pp. 537 – 550.

[32] Weiss, D. and C. Lai, *Software Product Line Engineering: A Family-Based Software Development Process,* Addison-Wesley, 1999.

[33] M. V. Zelkowitz and I. Rus, "Defect Evolution in a Product Line Environment", *JSS*, 70, 2004, pp. 143-154.