The Challenges of Applying Service Orientation to the U.S. Army's Live Training Software Product Line

Jeremy Lanman

U.S. Army PEO STRI
12350 Research Parkway
Orlando, FL 32826
+1 407 384 5307

Jeremy.Lanman@us.army.mil

Rowland Darbin
Jorge Rivera

General Dynamics
12001 Research Parkway, Suite 500,
Orlando, FL 32826
+1 407 275 4820

Rowland.Darbin@gdc4s.com
jorge.rivera@gdc4s.com

Paul Clements
Charles Krueger

BigLever Software
10500 Laurel Hill Cove
Austin, Texas 78730 USA
+1 512 426 2227

pclements@biglever.com
ckrueger@biglever.com

ABSTRACT

Live Training Transformation (LT2) is the product line strategy put in place by the United States Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI). The purpose of the LT2 product line is to provide a common set of core assets including architectures, software components, standards and processes that form the basis of all Army Live Training systems. As products consuming LT2 core assets evolve to meet the latest requirements of the military live training community, changes to the core product line architecture must also be made. Based on thorough analysis of the LT2 core capabilities and user trends toward web-enabled and mobile computing technologies, a Service Oriented Architecture (SOA) strategy was identified and adopted as the objective architecture for the evolving LT2 product line. Future success of the LT2 product line now depends on the alignment of product line engineering concepts with the business and technical benefits of SOA, and to ensure that systematic reuse continues to provide substantial return-on-investment for the Army. This paper addresses the challenges of adopting SOA into an existing software product line, the unique circumstances of the LT2 SOA environment, and present a set of analysis and design considerations for the product line engineering community.

Categories and Subject Descriptors

D.2.2 [**Design tools and techniques**]: product line engineering, software product lines, feature modeling, hierarchical product lines

General Terms

Management, Design, Economics.

Keywords

Product line engineering, software product lines, feature modeling, feature profiles, bill-of-features, hierarchical product lines, variation points, product baselines, product portfolio, product configurator, product derivation, product audit, second generation product line engineering

SPLC 2013, August 26 - 30 2013, Tokyo, Japan

Copyright 2013 ACM 978-1-4503-1968-3/13/08…$15.00.
http://dx.doi.org/10.1145/2491627.2491649

1. INTRODUCTION TO LT2

Live Training Transformation (LT2) [9] is the product line strategy put in place by the United States Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI). Through the use of LT2, the Army's office of the Project Manager (for) Training Devices (PM TRADE) builds and maintains live training systems in support of homestation training, deployed training, urban operations training, Maneuver Combat Training Center (MCTC) training and instrumented live-fire range training.

Prior to the implementation of the LT2 product line, live training systems and devices consisted largely of products developed separately by a variety of different manufacturers to comply with disparate requirement sets and were designed and implemented without a common framework. Commonality was not attempted and interoperability among systems was rare, difficult, and costly to achieve. Configuration changes to both hardware and software were often performed on-site as part of the sustainment effort, making configuration control virtually impossible.

Recognition of the commonalty of requirements between training systems and the degree of redundant work effort among contractors led to the establishment of common architecture frameworks, the Common Training Instrumentation Architecture (CTIA) and the Future Army System of Integrated Targets (FASIT), that drastically improved the reusability of developed training components. These components formed a technology shelf that enabled a high degree of reuse between products. As the number of products using the technology shelf increased, so did the corresponding complexity of managing the common software baselines and product feature sets.

The product teams using CTIA were uncoordinated in their efforts, resulting in redundant implementation that created similar features and resolved many of the same bugs. While still drastically more efficient than stovepipe development, it became evident that greater efficiencies could be gained by implementing a common governance strategy across the LT2 domain.

Reuse of core assets alone provides a substantial cost reduction in the development of new products but active participation in the *lifecycle* of the core assets assures that they remain relevant to future development and applicable to the breadth of the live training community. Controlled governance of core assets permits changes, upgrades and fixes developed for and by one product to be applied to others. This concept provides the inherent logistics support benefits that derive from commonality, standardization and interoperability including the reduction of total life cycle costs. This continuing transformation has generated a significant return-on-investment to date within PM TRADE's live training system acquisition portfolio generating an estimated $340M in cost avoidance over an eight-year period across the development and sustainment of 150 Live Training Systems deployed systems worldwide.

The LT2 vision has created a family of live training systems using a common architecture with common data, standards, processes, and components. Going forward, the LT2 vision is to combine the benefits of product line development with the benefits of SOA. By embracing these mutually beneficial technologies LT2 can ensure that systematic reuse continues to provide substantial return-on-investment for the Army.

2. LT2 AND FIRST GENERATION PRODUCT LINE ENGINEERING

Product Line Engineering (PLE) has roots that span at least four decades, going back as far as Parnas's seminal paper on product families in 1976 [20]. We characterize some of the early and long-standing approaches to Product Line Engineering as *first-generation.* First-generation PLE (1GPLE) includes:

- A strong dichotomy between domain engineering and application engineering, or core asset development and product development.
- Explicit inclusion of non-software artifacts in the collection of core assets.
- Focus on features [13] as the language to describe a product line's domain and a way to discriminate products from each other
- Acknowledgment of configuration management as an essential practice under PLE without a strong distinction between core asset CM and product CM

These approaches have yielded a rich legacy of product line success, as evidenced by numerous case studies [6][11][18][21][23]. First generation product line engineering for LT2 took the form of multiple projects reusing core assets with governance administered through a common asset repository by each PM TRADE Product Manager. The Product Managers were responsible for the configuration baseline of their systems throughout the products' total life cycle. As with any product line, the primary challenge is the management of the product line, not the technological barriers. This means that the process by which PM TRADE manages products must be deliberate, disciplined, and coordinated in order to maximize use of common assets, components, and subsystems in the development of new products. PM TRADE must synchronize the production of products to gain efficiencies, enable supporting efforts, and maintain seamless interoperability between components, products, and systems. Since the Product Managers' responsible management levels were disparate, the required coordination to ensure product line strategy was successful, albeit challenging and painful.

Adaptation of the first generation LT2 principles was highly successful and proved that greater gains could be attained by embracing the second generation product line philosophy and overcoming new obstacles that the first-generation product line highlighted in the LT2 environment.

3. LT2 AND SECOND GENERATION PRODUCT LINE ENGINEERING

As the primary focus of 1GPLE is effectively managing the core assets that compose the product line, the focus of Second-Generation Product Line Engineering (2GPLE) [5][10][14][17] is not only on the management of the core assets but on the philosophy of how the management of core assets should be governed.

In the world of manufactured hard goods, a product line refers to the variations on a common theme, where multiple similar products are combined into one line that offers different sizes, colors, features and functions, with a common goal of filling customer need for a particular kind of item. The 2GPLE paradigm strongly embraces the factory analogy. Continuing the analogy to engineering a product line of hard goods, it is much more effective to view systems and software product line engineering as creating a means of production – a single system or "factory" capable of automatically producing all of the products in a product line – rather than viewing it as creating a multitude of interrelated products. This idea is rendered in Figure 1.

Figure 2 shows the single production line perspective for producing the LT2 product line; the focus is on the means of production. Products that emerge on the right side of the diagram are automatically produced by a singular means of production:

- Feature profiles (top) that describe optional and variable features for the products in the product line where each product in the product line is uniquely defined by its own feature profile
- Shared assets (left) such as requirements, architectures, designs, models, source code components, test cases and documentation that can be configured and composed in different ways to create all instances of soft assets and products in a product line. Variation points shown within these assets are exercised to configure them for a product according to the features selected in the feature profile for that product. This results in feature-based variation management.
- Product configurator (center) that automatically composes and configures products from the shared assets, using the feature profiles to determine which shared assets to use and how to configure variation points within the assets. For LT2, the configurator is Gears [3].

In terms of Chastek, Donohoe, and McGregor's advice for building a production strategy [4] embracing the factory paradigm combines at least three of their desiderata:

- "**Be automated**. Enable the engineers to carry out repetitive functions quickly and correctly…"
- **"Be generative**. Generate low-level artifacts such as source code and detailed documents from higher level models…"
- **"Be transformative**. Allow dissimilar data formats to be aligned…" A transformative production strategy results in a production method that uses a small number of meta-models, preferably one, as the basis for models of various kinds of development information.

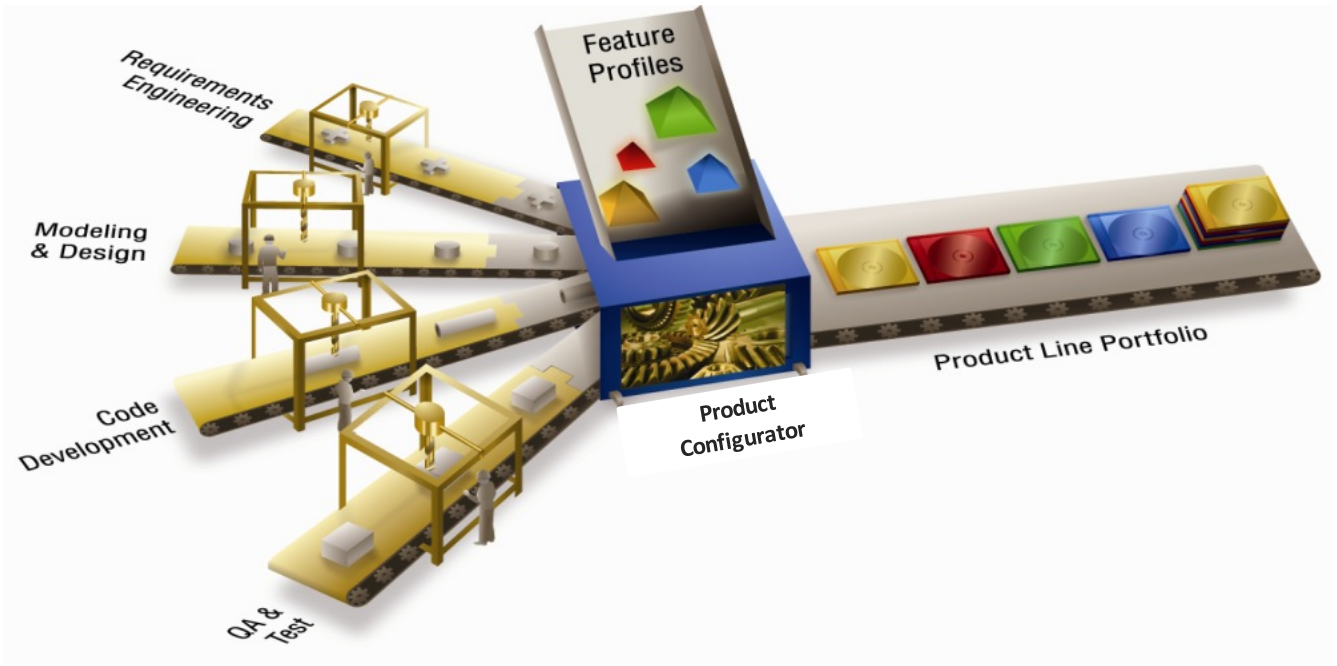

Figure 1 2GPLE's factory paradigm (© BigLever Software)

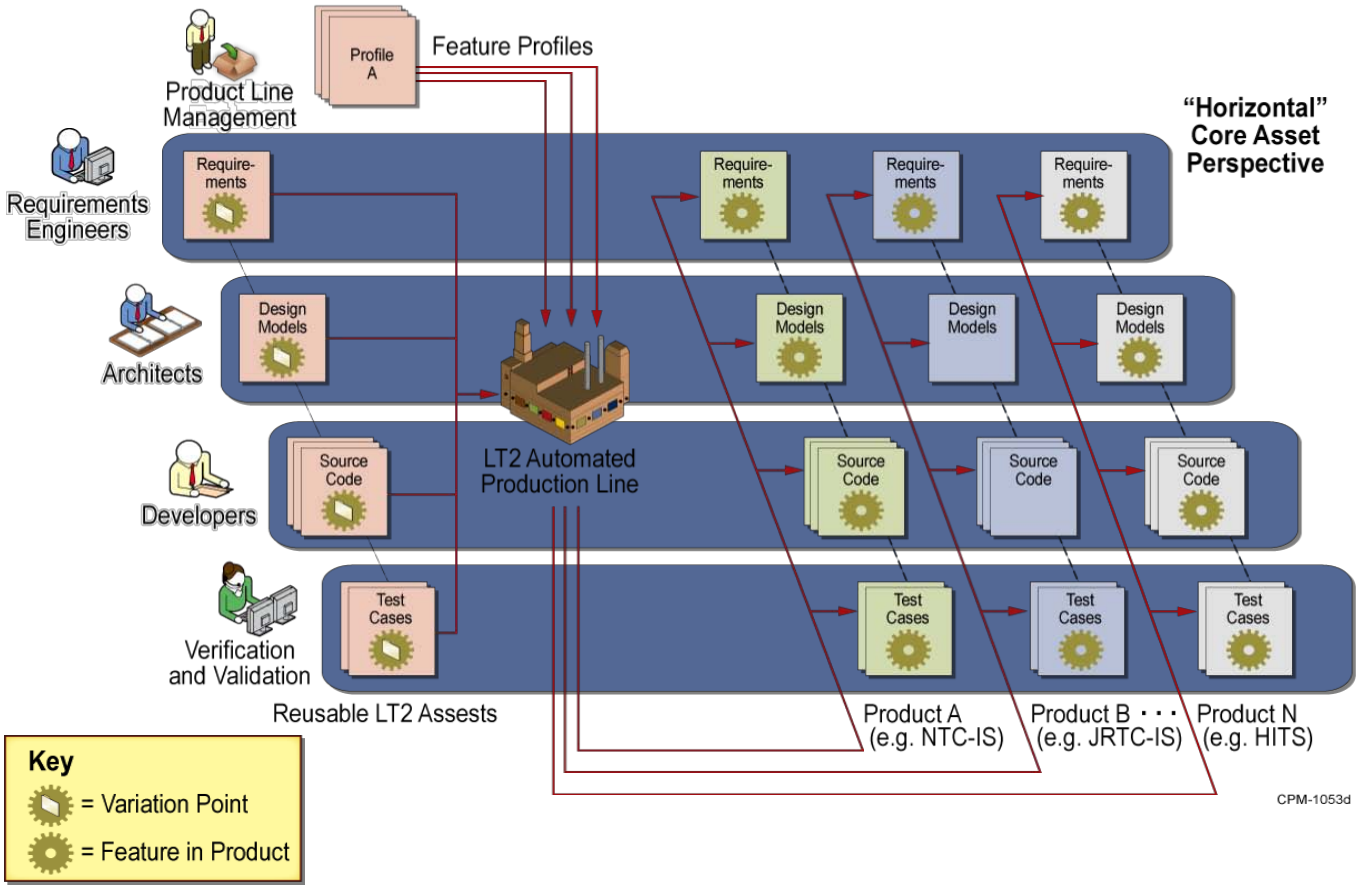

Figure 2 LT2's production line (© General Dynamics)

To effectively govern these variations, it was essential to develop the factory that would create the core assets from a single repository. Product teams shifted away from maintaining their own repositories, fixing bugs and adding features that would have to be merged later and instead, focused on defining the feature profiles that made their projects unique and distinct from the common baseline. New features and bug fixes were viewed holistically from the product line perspective in advance and immediately made available to the community instead of lagging until the beginning of the next development cycle when the previous development was merged into a baseline.

Adaptation of the 2GPLE strategies has been hugely successful for the product teams resulting in significant cost savings from elimination of configuration management, coding and testing associated with expensive baseline merges. Some of the governance problems experienced by the first-generation are still present but to a lesser degree, and not all core assets have had governance applied consistently. Though the governance of requirements and testing repositories still have not yet realized the benefits to the same degree as source code, standards, defects, metrics and configuration management. Redundant development and merging have been drastically reduced or eliminated entirely. Realizations of commonalities between products are greater than expected and variations are being used to tailor products in ways that target the core differences in needed capabilities while still retaining the benefits of industry collaboration.

4. LT2'S RETURN ON INVESTMENT

The LT2 Product Line strategy has generated significant return on investment to date within PM TRADE's live training system acquisition portfolio. Over the last eight years, some $340 million in cost avoidance has accrued across the development of Combat Training Centers Instrumentation Systems, Home Station Instrumentation Systems, Instrumented Ranges, and Targetry. The 2GPLE approach is projected to save another $200 million over the next two to five years [8] (Figure 3).

The demonstrated success of the LT2 product line has led to adoption by other military services including the Marine Corps and Air Force, and proliferation of training systems to multiple echelons within the Army [5].

5. WHY SOA?

The U.S. Department of Defense is constantly striving to improve the training of soldiers while reducing related costs. More specifically, three major areas of improvement have been identified within the military simulation and training domains:

- These systems often lack the ability to interoperate with one another unless extensive measures are taken to natively interface them.
- When users require on-demand capability, software applications, and upgrades, they must wait for fielding support and personnel to provide installation on each client.
- Massive volumes of data are being stored and processed by a variety of unmanaged clients and servers requiring excessive physical space.

The U.S. Army in particular, is considering two strategies to address these issues and recommend improvements: Service-oriented architecture (SOA) and cloud computing. SOA migration will, it is hoped, enable total system interoperability, resulting in composable, reusable, and loosely coupled services. Cloud computing will allow services, components, software applications, software updates, and upgrades to be readily available where consumers can access them as needed.

Currently, the CTIA is one of the three architectures defined by the LT2 product line. It is used by LT2 products to define interoperability standards among live training applications to support force-on-force and force-on-target training. Using an introspective approach, including honest dialog and user feedback, it was determined that CTIA must evolve to address technology obsolescence and meet the growing needs of the live training community. Therefore, in order for the architecture to meet those needs, a Service Oriented Architecture (SOA) approach was identified as the preferred strategy. The CTIA team conducted a series of workshops and utilized SOA training and Human Centered Design (HCD) techniques in order to identify and prioritize the strategic business goals and objectives for the LT2 product line.

Moreover, the CTIA team selected and prioritized service oriented design principles, which are being applied to the architecture in order to achieve the goals previously mentioned. These efforts resulted in a roadmap for the SOA migration and evolution of CTIA to Training as a Service (TaaS). The term TaaS is used by the U.S. Army internally and it refers to an "on-demand training environment" delivery model in which training software and its associated data are hosted centrally (typically in the cloud) and are accessed by users using a thin client, normally using a web browser over the Internet.

6. LT2 GOALS AND OBJECTIVES

Decisions to change a product must be driven by the goals and objectives of the customer and other key stakeholders if they are to be successful. Just as the decision to adopt a product line approach for LT2 involved recognition of avoidable duplication, the decision to migrate to a service oriented architecture involved recognition of deficiencies in meeting upcoming fielding needs for CTIA based training systems. To ensure that the adoption of SOA addressed the true needs of the LT2 community the architecture team, comprising key stakeholders based on influence and interest, defined and prioritized the strategic business goals and objectives for the Live Training Transformation architecture. CTIA provides the foundation for this architecture; however, business goals and objectives were extended to the LT2 community at large to ensure that the CTIA architecture aligns with community needs. These goals were then used to determine the priorities for the technology insertion effort.

The business goals were derived using the Goal-Question-Metric (GQM) paradigm [1]. GQM is used to define measurements such that:

- Resulting metrics are tailored to the LT2 organization and its goals
- Resulting measurement data play a constructive and instructive role within the LT2 Product Line
- Metrics and their interpretation reflect the values and the viewpoints of the different stakeholders affected

A GQM model is a hierarchical structure (Figure 4) starting with a goal (specifying purpose of measurement, object to be measured, issue to be measured, and viewpoint from which the measure is taken). Each goal is refined into several questions intended to elaborate the issue into its major components. Each question is then refined into metrics, some of them that can be answered objectively through measurement, and some of them subjective. The same metric can be used in order to answer different

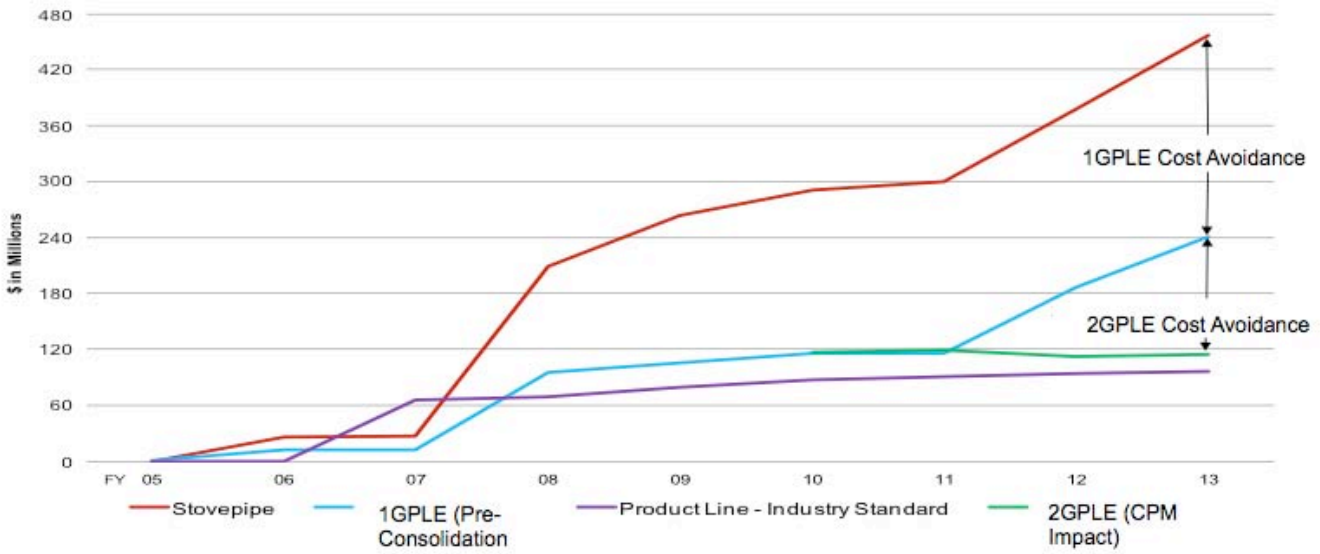

Figure 3 LT2 return on investment (© General Dynamics)

questions under the same goal. Several GQM models can also have questions and metrics in common, making sure that, when the measure is actually taken, the different viewpoints are taken into account correctly (i.e., the metric might have different values when taken from different viewpoints). For each of the measurement areas that follow, the GQM process is documented and the details of recording and reporting the resulting metrics are described.

The Architecture team created broad goals and used the GQM methodology to (a) refine each goal into questions about the goal and (b) drawing metrics that help to validate that the questions are answered and therefore, how well the goals are being met. The major goals for LT2 are summarized in **Table 1**.

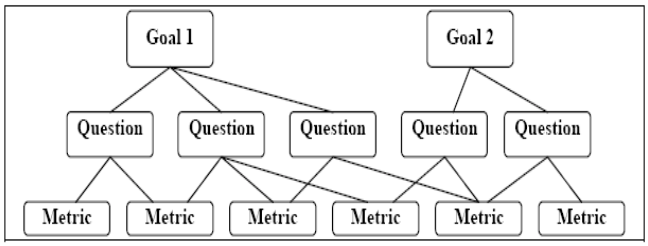

Figure 4 GQM's hierarchical model of goals, questions, and metrics

A significant investment has already been made in CTIA and LT2 components. The business goals and architectural objectives must retain at least some level of backwards compatibility to allow the product line to recoup a return on investment over approximately the next five years. Backwards compatibility may be achieved by implementing any combination of a number of different techniques, including compile-time (application programmer interface), run-time (network protocols) or design-time (data models) backwards compatibility. Although run-time backwards compatibility may provide the least impact to fielded systems, it might incur the greatest cost in implementing. In the past, the product line has strived to achieve at least some level of compile-time backwards compatibility since it provides a relatively equal balance of cost versus benefit.

Table 1 GQM-derived goals for LT2

Goal	Description
Reduce operational costs and complexity	Focused on reducing operational costs and complexity.
Align and support specific product development	Focused on using current live training development efforts to advance the product line by meeting core product needs and encouraging contributions back to the product line.
Enable enhanced soldier training effectiveness	Focused on increasing the training effectiveness of the LT2 products.
Reduce development and sustainment costs	Focused on reducing the costs associating with building and sustaining LT2 products.
Increase technology agility	Focused on maximizing the ability of the architecture to incorporate new technology.
Leverage other Army systems	Focused on providing ability for the LT2 product line to leverage systems and services that are developed by others (i.e. virtual, constructive, mission command or other live training systems).

Alignment with Army Standards. The U.S. Army's Chief Information Officer has issued directives to implement a Common Operating Environment in order to support cloud computing and virtualization on the Army's Global Network Enterprise Construct [7]. As we evolve the LT2 product line to identify and achieve business goals for the live training community, we must also keep in mind the broader business goals and technology objectives for the Army.

With these considerations it was determined that the objective architecture would be designed using SOA principles and web services with a shift from thick to thin clients [16]**.** User applications will be able to be hosted in a web browser or run as

"apps" on mobile devices and tablets. By making this switch, the architecture will become more flexible, scalable and simpler to operate. These changes also support the goal of hosting some or all of a range's services at a Regional Training Center (RTC), which could support multiple ranges from a common data center.

7. CHALLENGES TO ADOPTING SOA

CTIA is the foundation architecture of the LT2 product line. The objective of any architecture should be to combine the common parts of software development such as logging, archival, retrieval and inter process communications and build it up to a level that combines the business logic common to the live training domain such as, in the case of force on target exercises; entity addressing, entity filtering, and brokering control of instrumentation.

Ensuring Reuse. These frameworks succeed by providing a uniform and highly reusable feature rich environment that allows developers to focus on their primary objective of implementing business level use cases and not on repetitive implementation details. CTIA's success can be realized by the fact that it forms an average of 50% of the code base for all live training systems deployed since 2006. Of the two million lines of code in the CTIA framework, LT2 products typically use 57% of it. For the eight currently active fielded products listed on the LT2 Portal, this is a reuse factor of 4.5. Due to the large investment of the current architecture and the multitude of component dependencies it is unreasonable to expect that the new architecture can be developed in an isolated environment and deployed to wholly replace the existing architecture. Backwards compatibility must be maintained with legacy software components through the existing CTIA Framework interfaces.

Using the latest technology available during the design effort in 2001, CTIA was designed using the Common Object Request Broker Architecture (CORBA) interface definition language (IDL). It enabled a measurable compliance level without specifying an implementation language. The CORBA IDLs were aggregated into API-level Object Models providing methods and higher-level abstractions (e.g. proxies for remote objects). The CTIA Object Models have evolved to a point where they remain very stable and application development is almost universally tied to the OM implementation and not to the CORBA IDL. Additionally the universal adoption of the CTIA framework has negated the need for a compliance level. As a result CTIA has been able to separate itself from the IDL constraints and evolve to new technologies without affecting compatibility or external development. Because the individual CTIA services were discreetly defined by IDL interfaces applying web services at the IDL boundaries was an initial design plan but further analysis and understanding of the true intent of SOA revealed that the tight coupling between services and the state full nature of the existing services would not provide the benefits identified by the customer, and the big band deployment and migration to new versions of CTIA services would persist. The migration away from the IDL interfaces in the previous generation of CITA is a direct result of the service interfaces being too interdependent. Consumers were unable to effectively implement portions of the system that were necessary for their implementation and instead resorted to an over arching object model that reduced the services to a single monolithic architecture. This tight coupling limited innovation by consumers and was a primary driver for the move to modernization. New agnostic service boundaries must be established that encapsulate the core capabilities that can be segregated into independently fieldable capabilities dependent only on the consumers use case.

Business process. In the Live Training domain the technical problem does not directly map to the IT business process for producing goods and services which SOA is typically modeled on. The standard SOA business process is an orchestration of multiple business functions each of which rely on the results of the previous function to accomplish a discrete task. The archetypal example such a business process is booking an order -> updating inventory -> shipping -> billing. Business units in a live training environment are providers of content and context to artifacts generated within the system. The system collects artifacts and then consumers generate review content from the artifacts. The path through artifact generation, manipulation and presentation is not dictated by a predefined orchestration but ad hock, depending on the fidelity of the training environment. A combat training center for example has multiple organizations dedicated to providing context and content for discrete aspects of the training exercise such as fires, upper echelon support, or areal support, where as a home station training range instantiation is typically an individual assessing the time on target, or efficiency in meeting the training objectives for a single unit.

Deployment/cycle time. In the archetypal SOA deployment the SOA system is ubiquitous and accessed from multiple disparate organizations. The system is always available with no defined end state. As live training systems are deployed currently, installations exist as isolated standalone systems. Training exercises have specific training objectives and data that are not shared between concurrent exercises at different ranges. Service composition is a function of the training audience and range, from combat training centers with dedicated rack servers down to individual ranges consisting of a single workstation operated directly by an individual from the training unit. These systems are accessed and maintained onsite and their state is dependent on the phase of the training rotation. Though the architecture to which we're migrating (henceforth herein called the *objective architecture*) will be a ubiquitous solution overall, the SOA implementation of CTIA must account for the different phases of training where different subsets of services are available depending on the training rotation state.

Security. Any changes to the CTIA and LT2 architectures and components must consider the security and accreditation impacts in order to support the information assurance policies and DIACAP process. We must also consider that the Army evolves to implement cloud computing and virtualization that the security and information assurance requirements are likely evolve and introduce new requirements.

Technical Concerns. SOA adoption challenges typically center on bandwidth, scalability, and technical issues based on implementation details such as limitations of proprietary ESB capabilities. Historical CTIA development has resulted in a set of metrics that ensure that all development activities are meeting the required Technical Performance Measures (TPMs). Continuous integration and testing means that any time these values are exceeded the development team takes immediate action. Existing test rigs and training scenarios provide a baseline for validation testing. To date, system performance exceeds the previous generation CTIA. The first transition architectures focus on the most reused and also the most performance-sensitive elements within the system, ensuring that these issues are addressed early and often.

8. OVERCOMING CHALLENGES

Despite the challenges, SOA design principals do map very well to the Live Training Domain. The segregation of capabilities through service boundaries that can be deployed as needed and horizontally scaled has been a stumbling block of the current architecture. The enormous deployment disparity between combat training centers conducting battalion level training and small ranges for individual and squad training requires significantly different levels of user interaction with the system and model fidelity but the basic core needs of the training unit still share a common data model and have a great deal of capability overlap.

Deploy new system along old system

Though the current system has a limited ability to natively support the technological agility that is being requested from the active product teams, end users, and new acquisition efforts, it still meets the vast majority of core live training requirements and is has become a very stable and reliable architecture. The most effective way to continue to benefit from the investment already made is to continue to deploy the current system and supplement new capabilities using the new system whenever possible.

There is no expectation that the new CTIA system will be able to completely supplant the current system and all of its capabilities in one fell swoop. The new system will be deployed with the current system and synchronicity will be maintained though a system interop adaptor. New capabilities that can't be met by the current system such as mobile access to situational awareness and observation recording by combat trainers encourage immediate adoption by product teams. Product teams will build upon this new system and migration of existing capabilities will occur though the natural evolution of component lifecycle updates and by a dedicated team of developers migrating the most essential core capabilities until the old system can be completely retired. This is the same development model that built the current library of core assets. SOA migration using composable, reusable, and loosely coupled services enable greater interoperability both internally and to external systems, this is being embraced by leveraging as much of the current system as possible during the migration. These design goals enable high degrees of reuse of commercial and tactical systems with web enabled interfaces such as situational awareness displays and tactical data collection systems. Reuse of these tactical and commercial systems is enables live training capabilities to be fielded with a consistent user interface that soldiers and trainers are already familiar with.

Current LT2 components are thick applications built on the common architecture. Though there has been a large amount of re-use between components for the most part capabilities have been improved as they have been incorporated leading to subsequent components having different implementations to provide overlapping core capabilities such as unit organization trees, entity property editors and status viewers. Reducing redundant capabilities in components by defining discrete interfaces between re-usable services that are loosely coupled and are not compiled as part of a single monolithic application will encourage future development to improve the common core instead of tailored solutions. Controlling service contracts as well as their implementations though the product line will also ensure that any new implementation that obsoletes an older instantiation will be backwards compatible or at least more simple to integrate if changes to the service contract are controlled.

Alternatives to wrapping current services

CTIA provides the service hosting software infrastructure that hosts both CTIA services and product services. This includes a common message bus, application hosting, application monitoring, and web-based user interface services.

Instrumentation, targets, and cameras are real world devices that are controlled, have communications intricacies, have status, etc. these are now represented by discrete services contracts instead of being handled with a single all knowing "service". These can be deployed to sites based on their individual needs and aren't interdependent which reduces the deployed footprint and associated maintenance and operator training.

Creating services that directly map to physical capabilities is relatively straight forward for but constructs that don't have physical boundaries such as geographic location information and entity engagements between there is a need to determine the appropriate level of granularity in the service contract. Determination of granularity was performed by breaking down use cases into business processes; Each process step was analyzed to determine what capabilities were need to complete the step and the resulting list of capabilities. When groups of capabilities appeared together in multiple use cases and steps they were grouped into service candidates. Analyzing the existing product line variations and divergent implementations of existing components also influenced the grouping of capabilities in service candidates. Variations in implementations between overlapping tools such as the Fire Support Tool and Fire Support Tool Light clearly identified a domain need for variations in model fidelity which can be encapsulated in service implementations that share service contracts, separation between battle damage assessment and the fire engagements also allowed different variations in effects modeling between projects to be encapsulated and re-used within different system components instead of duplicated as they are in the current system. This provided clean, well-defined interfaces to a set of reusable services that encapsulated the variations between sites and agnostic interfaces between system components while preventing proliferation of services that would have become unmanageable in the product line. Due to the high degree of maturity, the existing data-model and requirements were reused almost in their entirety with minor improvements based on lessons learned and normalization.

Alignment with product schedules

Product teams in LT2 target specific echelons and type of live training. Within the Live Training domain the core capabilities are very similar but as training progresses up to larger echelons and more complex objectives the needs of the training platform increase. It's important to note that in almost all cases the requirements of the individual capabilities do not change, but the number of capabilities required to support the training objectives is increased. For land navigation training, participant tracking is essential, when training force on force you must know the engagements as well as participant tracking. Alignment to the defined product teams' deployment schedules enabled the development team to target the capabilities need for less complex product requirements first. Doing so encourages product teams to adopt the system early and mature the solution as other capabilities while developing new capabilities to me the more demanding use cases. Additionally as product teams are already fielding with the existing system, high value services, such as web enabled editing of organization trees were included in early transition architectures to provide incentive to adoption above and beyond the logistics and operator training benefits to the new deployed footprint.

Technical Challenges. By adhering to the core tenants of SOA design and not focusing on the technical implementations of specific ESB providers, implementation decisions have focused on ensuring that the system meets the intended use case and performance requirements. Up front testing was performed to ensure that message volume and latency were scalable to performance requirements that were already established for the previous generation of CTIA. Technologies such as message queuing, databases, and ESB constraints were evaluated before any service implementations were initiated. Performance limitations of individual services can be mitigated by technology selection such as REST interfaces and AMQP message queues instead of SOAP calls. Ensuring that services are stateless allows for vertical load balancing to meet performance requirements. History has taught us that system performance, and user perception of system performance, are key to adoption by users and product teams. Making performance concerns primary drivers to the design, and putting automated testing in place as part of the continuous integration development environment, causes performance issues to become immediately known and addressed before they are allowed to propagate though the system.

9. ALIGNMENT OF SOA AND PLE

The migration to a SOA paradigm is not a migration away from product line development (see, for example, [22]). In fact, we found that the LT2 product line process for managing components applied directly to the development of SOA services. Changes to the governance process were almost entirely additive to account for the idiosyncrasies of service development. Reuse of the existing core components feeds directly into the mentality of reusable services and development in the shared environment of a product line such that the impacts to development in a shared service oriented infrastructure with defined service contracts is not a significant change in developer ideals.

The most important part of migrating to a SOA is to understand your customers' goals. It is critical to understand what is driving the change in the system and what the final sate of the system must provide to the end user. Evolution of current systems to SOA is most likely not going to provide benefit without concurrently addressing the underlining problems of the current system. For LT2, the migration to SOA revealed that the core problems with the architecture were not in the choice of technology but instead with the separation of services and scalability of the interfaces within the system. Migration without thorough analysis to address the inherent problems would have provided a more modern interface and reduced some of the integration issues in the current architecture but the return on investment would have been very low relative to the amount effort expended.

When planning the SOA migration use your current product line as a guide. Understand your current capabilities and embrace those that are unique to your domain. Once these are clearly understood, determine the way the product is used within the domain. Scaling from extremely small to large scale environments was a large consideration for LT2 and was a primary driver in determining service granularity. Use variation in your current system as a guide to the areas was needs diverge; service boundaries should embrace these divergences. Consistent service contracts between implementations is good for embracing variations of model fidelity and breaking interdependencies for variations in fielded capabilities. Variations within service implementations will also exist and should be embraced.

10. OBJECTIVE ARCHITECTURE

The objective architecture is a cloud-based solution providing Training as a Service to the live training community. This solution is supported by the employment of several key technologies, including virtualization and wireless connectivity at the training sites, and the service infrastructure to support the applications required to record, monitor and augment live training data. The end state for the CTIA SOA effort is to provide the instrumentation system that supports this deployment.

CTIA SOA includes the service-oriented infrastructure as well as the services that provide the core capability necessary for all live training environments including entity tracking, artifact collection and manipulation. These core services are required for all categories of live training. The domain of live training spans from individual shoot houses, and weapons proficiency, up to battalion engagements that train battle staff. The objective architecture enables all levels of training to co-exist during large training exercises while not sacrificing the quality of training at any level.

Figure 5 shows the end state of the architecture deployed to a regional training center supporting exercises at home stations as well as combat training centers. These concurrent training exercises are leveraging domain expertise from trainers and analysts that may be physically distributed at various Army posts.

CTIA provides services consumed by individual combat trainers to collect training observations and artifacts where the training is taking place. Migration of this data collection and analysis is aggregated to combine the observation artifacts of all combat trainers. Combined, this aggregation of data provides inputs that TAF analysis can use to conduct higher echelon analysis of the training exercise. Combat Trainers are also able to access data from other combat trainers and data collected automatically such as unit position engagement data and battle effectiveness to conduct rapid and effective reviews with their unit. CITA is providing access to information and making the information available in a format that product teams can compose into simple to understand targeted applications for use by combat trainers, TAF analysis and soldiers.

Development of the objective architecture is decomposed into transition architectures to enable product teams to incrementally migrate to the new architecture as capabilities are matured. This also allows the CTIA development team to focus on the incremental development without a final big-bang deployment and sudden migration by product teams. The first transition architecture focused primarily on the service-oriented infrastructure and established baselines for components that compose the underlying databases, communications channels and deployment. Work on the most elemental of services such as entity and geographic location services were also begun. Subsequent transition architectures focus on services with high degrees of reuse including instrumentation which is used on all live training ranges, and fleshing out supporting services based on training use cases. Uses cases met by each of the transition architectures build on the least complex land navigation use case and work toward full-scale force-on-force engagements at the Brigade echelon. Product teams adopt the new architecture wholly as the capability is fully met and incrementally when new capabilities, such as mobile access to observations, are added and not available in the legacy architecture. During incremental adoption, the SOA system is deployed with the legacy system and interoperability is maintained though an "interop" service. Due to high degree of interdependence in the legacy system the "interop"

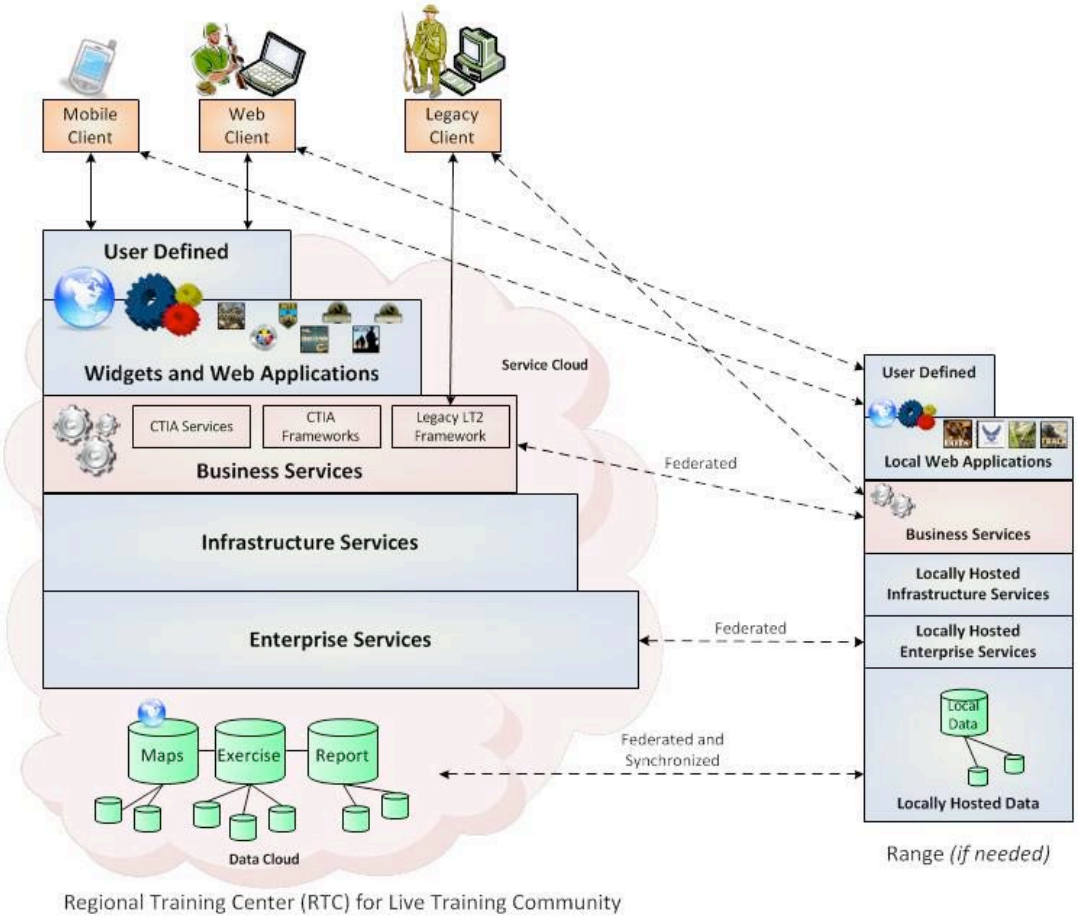

Figure 5 Architecture end state (© General Dynamics)

service is dependent on almost all new services, additionally the sheer volume of capabilities reliant on the legacy system the "interop" service will remain for an extended duration after development of the SOA system has matured to the objective architecture.

The true test of capabilities in the new system, however, is how effectively the new architecture meets the customers' stated goals. As part of the GQM process during initial planning, the metrics were decomposed into collectable and measurable elements that can be instrumented into the core infrastructure of the source code. This allows for simple calculation of usage during deployment without manual collection. Usage metrics indicate adoption in deployed systems and performance of the services within the system. Instrumenting the code in advance makes calculation inherent instead measuring only during system integration and test; this gives a better view into actual product line utilization in fielded systems, which in turn allows for better tailoring of variation and deployment footprints to meet user needs.

11. SUMMARY AND CONCLUSION

This paper has described the migration of CTIA and the LT2 Product Line to the TaaS paradigm using a specific set of modern computing technologies that will enable rapid delivery of training capabilities across servers, mobile devices, and heterogeneous platforms. Specifically, service-oriented architecture (SOA) and cloud computing were considered, which can satisfy the requirements of the TaaS and COE and meet user demands for an enhanced training experience.

Furthermore, this paper discussed the approach taken to elicit the future needs of the Army's live training community, and how cloud computing and SOA are leveraged to meet required capabilities.

Currently many training systems acquired, fielded, and sustained by the U.S. Army are unable to seamlessly comply with a continuously evolving and often complex computational environment. The current state of such training systems must advance to a Training as a Service (TaaS) future state in order to adapt to a volatile defense budget, conform to policy updates, and enhance the training capabilities afforded to the Warfighter. TaaS will transform current Army training applications into distributed web-based services, allowing them to be accessible across any location via thin client workstations and wireless mobile devices.

An important conclusion and, we hope, contribution of this paper is that we are able to confirm in practice what previous authors have asserted about the expected compatibility of SOA approaches and product line engineering (for example, [1] [11][15][19][22]). Whereas those earlier works have opined about the compatibility in abstract or theoretical terms, we have presented a real-world experience report about migrating a very large and successful product line to the service oriented architecture paradigm. Our experience affirms that "SOAs and software product line approaches to software development share a common goal. They both encourage an organization to reuse existing assets and capabilities rather than repeatedly redeveloping them for new systems to achieve desired benefits such as productivity gains, decreased development costs, improved time to market, higher reliability, and competitive

advantage… Both approaches promote reuse by developing applications/products based on a set of reusable components. Those components are developed with well-defined interfaces and processes that specify how the components are to be used, which enables applications/products to be produced in less time. Adopting either approach requires implementing similar organizational policies and practices necessary to adopt a new technology or a new way of doing business." [19]

Ultimately, SOA and cloud computing will allow product-line architectural services, components, software applications, and software updates and upgrades to be readily available in a logically centralized repository where consumers can access them as needed. Virtualization will improve organization between database servers and reduce hardware footprint. Some of these gains have already been realized and some are still in the process of development. These technology updates will allow CTIA to achieve greater ROI by providing an inherent upgrade to previously fielded systems and enabling additional reuse. ROI numbers for CTIA have been calculated based on not needing to incur the initial development costs for the features provided on each deployed product. There is also significant cost savings by preventing the duplication of software sustainment for products.

12. REFERENCES

[1] Abu-Matar, M., Gomaa, H. "Variability Modeling for Service Oriented Product Line Architectures," *Proceedings, 15th International Software Product Line Conference (SPLC),* 22-26 Aug. 2011, pp. 110 - 119

[2] Basili, V.R., et al. (2002). The Goal Question Metric Paradigm. In: Marchiniak J. J (ed.): *Encyclopedia of Software Engineering*. New York, pp. 578-583.

[3] BigLever Software, "BigLever Software Gears," http://www.biglever.com/solution/product.html

[4] Chastek, G.; Donohoe, P.; McGregor, J. *Formulation of a Production Strategy for a Software Product Line.* Technical note CMU/SEI-2009-TN-025, Software Engineering Institute: 2009.

[5] Clements, P., Gregg, S., Krueger, C., Lanman, J., Rivera, J., Scharadin, R., Shepherd, J., and Winkler, A., "Second Generation Product Line Engineering Takes Hold in the DoD," *Crosstalk, The Journal of Defense Software Engineering*, USAF Software Technology Support Center, 2013, in publication.

[6] Clements, P.; Northrop, L. *Software Product Lines: Practices and Patterns*, Addison-Wesley, 2002.

[7] Department of the Army, 2010 Army Posture Statement, "Global Network Enterprise Construct," https://secureweb2.hqda.pentagon.mil/vdas_armyposturestatement/2010/information_papers/Global_Network_Enterprise_Construct_(GNEC).asp, 2010.

[8] Dillon, M., Rivera, J., Darbin, R., Clinger, B., "Maximizing U.S. Army Return on Investment Utilizing Software Product-Line Approach," Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), 2012.

[9] Dumanoir, P., Rivera, J. (2005). *Live Training Transformation (LT2)-A Strategy for Future Army and Joint Live Training*. 2005 Interservice/ Industry Training, Simulation, and Education Conference (I/ITSEC), Orlando.

[10] Flores, R., Krueger, C., Clements, P. "Mega-Scale Product Line Engineering at General Motors," *Proceedings of the 2012 Software Product Line Conference (SPLC)*, Salvador Brazil, August 2012.

[11] Helferich, A., Herzwurm, G., Jesse, S., and Mikusz, M. Software Product Lines, Service Oriented Architecture and Framework: Worlds Apart or Ideal Partners*? Trends in Enterprise Application Architecture, Lecture Notes in Computer Science,* Vol. 4473, 2007, pp 187-201.

[12] Jensen, Paul. (2009). "Experiences with Software Product Line Development." *Crosstalk The Journal of Defense Software Engineering*, USAF Software Technology Support Center, 22, 1 (January 2009): 11–14.

[13] Kang, K.; Cohen, S.; Hess, J.; Novak, W.; & Peterson, A. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, CMU/SEI-90-TR-021, ADA235785. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.

[14] Krueger, C. and Clements, P. "Systems and Software Product Line Engineering," *Encyclopedia of Software Engineering*, Philip A. LaPlante ed., Taylor and Francis, 2013, in publication.

[15] Krut, R., "Service Oriented Architectures and Product Lines — What is the Connection?" Workshop on Service Oriented Architectures and Product Lines, 11th International Software Product Line Conference (SPLC), 10 September 2007. http://www.sei.cmu.edu/library/assets/Krut_presentation.pdf

[16] Lanman, J.T., Horvath, S.D., and Linos, P.K. (2011). *Next Generation of Distributed Training utilizing SOA, Cloud Computing, and Virtualization*. 2011 Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), Orlando, FL.

[17] Lanman, J.T.., Kemper, B.E., et al. (2011). *Employing the Second Generation Software Product-line for Live Training Transformation*. 2011 Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), Orlando.

[18] Linden, Frank J. van der, Schmid, Klaus, Rommes, Eelco. *Software Product Lines in Action*, Springer, 2007.

[19] Northrop, L., Clements, P., Bachmann, F., Bergey, J., Chastek, G., Cohen, S., Donohoe, P., Jones, L., Krut, R., Little, R., McGregor, J., and O'Brien, L. *A Framework for Software Product Line Practice, Version 5.0*, "Frequently Asked Questions." Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu/productlines/frame_report/FAQ.htm

[20] Parnas, D.L. On the design and development of program families. *IEEE Trans. Software Engineering* 1976, SE-2 (1), 1–9.

[21] Software Engineering Institute, "Catalog of Software Product Lines," http://www.sei.cmu.edu/productlines/casestudies/catalog/index.cfm

[22] Software Engineering Institute, "Workshop on Service-Oriented Architectures and Software Product Lines (SOAPL)—Enhancing Variation," http://www.sei.cmu.edu/splc2009/soapl.html

[23] SPLC Product Line Hall of Fame, http://splc.net/fame.html