Second Generation Product Line Engineering Takes Hold in the DoD

**Paul Clements, BigLever Software,
Susan P. Gregg, Lockheed Martin,
Charles Krueger, BigLever Software,
Jeremy Lanman, U.S. Army PEO STRI,
Jorge Rivera, General Dynamics,
Rick Scharadin, Lockheed Martin,
James T. Shepherd, Lockheed Martin,
Andrew J. Winkler, Lockheed Martin**

Abstract. Product Line Engineering (PLE) is a well-established engineering discipline that provides an efficient way to build and maintain portfolios of systems that share common features and capabilities. Systems—including DoD systems—built with PLE have, for decades now, demonstrated improvements in development time, cost, quality, and engineering productivity that consistently attain integer-multiple improvements over comparable non-PLE engineering efforts. Until recently there was no unified repeatable approach available; each PLE project went its own way. But now, two high-visibility DoD examples (Navy's AEGIS and Army's Live Training Transformation) are taking advantage of a strong and well-defined automation-centered approach that some are calling Second Generation PLE, and reaping substantial benefits as a result.

Introduction

The DoD is rife with systems that share much in common. For example, over 80 companies, universities, and government organizations are actively developing one or more of some 200 unmanned aerial vehicle designs. They differ from each other in important ways, but they resemble each other in ways that are at least as important. In 2004, the General Accounting Office was able to identify 2,274 separate DoD business systems (but nobody knows the true number) that are different, but also alike. The Joint Strike Fighter is being delivered in three main variants with very different capabilities, but they are all still the F-35. Communication systems, armored vehicles, tactical fixed-wing aircraft, helicopters—the list of large-scale examples of systems that are different yet the same goes on and on.

These examples are—or in many cases should be—product lines. A product line is a set of systems that share common features, and are engineered, developed, and sustained using a common set of shared assets[1]. The systems are built and maintained in a way that respects the variations in capability and function that they each need to provide to their respective users, but also takes maximum advantage of the commonality they share. PLE is the name of the established engineering discipline that far-sighted organizations use to accomplish this. It is an efficient way of building and maintaining portfolios of systems.

This article is about two high-visibility examples in the DoD where far-sighted organizations are achieving that efficiency. The AEGIS command and control systems of Naval surface combatants differ widely, but have so much in common with each other that it is more beneficial to consider them as variants in the same family. The Army's Live Training Transformation comprises a multitude of training systems covering a spectrum from single-soldier weapons trainers to large-scale synthetic force-on-force wargaming systems. Once again, there is benefit being gained by viewing them as a family.

PLE: Feeling Its Way in the First Generation

Systems built under the discipline of PLE have, for decades now, experienced improvements in development time, cost, quality, and engineering productivity that consistently attain integer-multiple improvements over previous engineering efforts. The PLE community, eager to spread the word, has over the years published a swarm of readily available case studies and catalogs of successful PLE-engineered families of systems in industry [14][3][9][12][15]. Many of the improvements reported are jaw-dropping, such as a family of embedded engine controllers that used to take a year to develop and under PLE take less than a week [3], or a family of computer peripherals can be built with 1/4 of the staff, in 1/3 of the time, and with 1/25 the number of bugs as the organization's pre-PLE products [14].

However, each of these successes employed its own unique approach and techniques applied atop the basic concepts in varying degrees and in varying ways. These approaches, which can be characterized as first-generation, were point-case effective but lacked a systematic, repeatable, codified methodology. All made a strong distinction between domain engineering (creation of reusable parts) and its equal counterpart application engineering (creation of specific products from those parts), focused on software code as the most important reusable resource, and used the concept of a feature to compare systems in a domain.

Nevertheless, the benefits were real and attention-grabbing. In addition to the hard numbers, PLE practitioners have consistently reported a wide array of less tangible (but arguably no less important) benefits, including:

- **Ability to perform continuous portfolio-wide insertion of new technology and new functionality at low cost**
- **Uniform look and feel to products and greater interoperability**
- **Higher engineer satisfaction with resulting lower workforce turnover**

This message was not lost on the Pentagon or its contractors, both eager to lower cost and to translate (for example) reduced time to market into reduced time to deployment to support the Warfighter. Some early but notable examples of DoD-oriented product line efforts include:

- **A product line of satellite ground control systems for the National Reconnaissance Office [3]**
- **A product line of weapons test ranges at the Naval Undersea Warfare Center [4]**
- **A product line of helicopter avionics systems for the Army's Technical Applications Program Office [2]**
- **A product line of submarine combat systems for the Navy's Submarine Warfare Federated Tactical System [8]**

These efforts, too, enjoyed the same kind of eye-catching benefits: Millions of dollars saved, delivery times slashed, and increased capability for lower cost.

Meanwhile, PLE as a discipline was evolving. Languages for expressing variation became more uniform and simpler, reflecting only what was needed in practice. Automation to support product derivation from shared assets moved out of the research labs and into real-world application, gaining robustness, simplicity, and usability. PLE adopted a whole-system perspective, a powerful generalization reflecting a move away from the field's software-only roots. The trends have crystallized into an approach some are calling "Second Generation Product Line Engineering" (2GPLE) [7].

PLE: Second Generation Maturation

Building on first-generation efforts, 2GPLE embodies a more well-defined and repeatable process, centered on a strong factory paradigm. Distinguishing characteristics of 2GPLE include:

1. Features express product variation: In the factory paradigm, we need a way to describe what product we are building, so the shared assets (requirements, designs, code, test cases, user manuals, etc.) can be configured appropriately. Rather than adopt a different "language" and mechanism for each type of artifact (for example, compiler directives for code, attributes for requirements, text variables for documents, and so forth), 2GPLE uses a small and consistent set of variation mechanisms [1] for all of the artifacts. Each product is described by giving a list of its features: "A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems" [10]. Features are used to express product differences in all lifecycle phase artifacts. This streamlines the development process and lets all stakeholders speak the same language.

2. Shared assets come from all lifecycle phases, not just the software: Early approaches to PLE certainly encouraged practitioners to include all kinds of artifacts in their collection of shared assets, but the unmistakable emphasis was on software. But in large-scale product lines, automated production of whole and consistent sets of lifecycle artifacts is essential. Managing these artifacts means imbuing them with variation points [1], which are places where an artifact can change to support different products. Variation points reflect the different feature-based product contexts in which the artifacts will be used. In 2GPLE, all supporting assets are considered equally important; software plays the same role as any other, or even (in cases where the products contain no software) no role at all.

3. Industrial-strength automation is employed in the form of a configurator, which is a tool that takes a feature-based description of a product and exercises the variation points in the shared assets to produce an artifact set that supports the named features. Product development thus becomes automated, so that application engineering (so important in first-generation approaches) becomes vanishingly small. (Both product line organizations in this article chose the BigLever Software Gears PLE configurator [11] as the automation engine to power their product line.)

Figure 1 illustrates these three distinguishing aspects of 2GPLE. A feature profile is a description of a product in terms of the feature choices. The configurator (here, Gears) uses the feature profile to configure each shared asset (by exercising its variation points) to produce the set of engineering artifacts specific to that product.

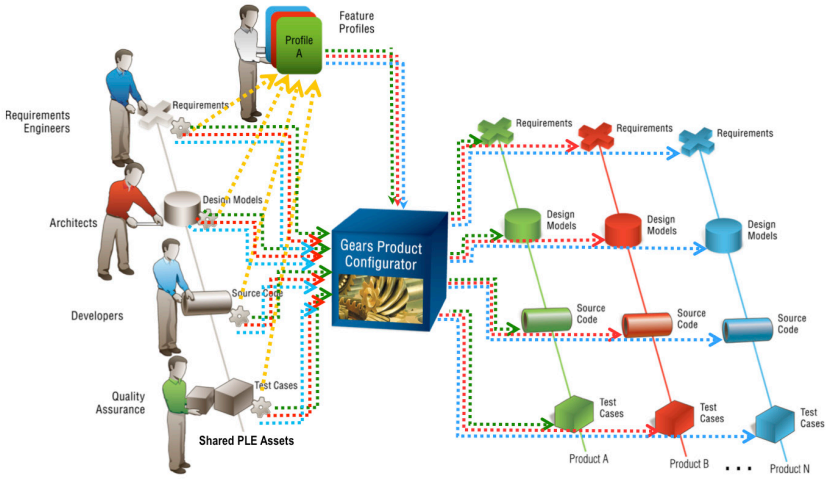

Figure 1. The 2GPLE factory paradigm. The configurator uses a feature profile for a product to exercise variation points (denoted by the gear symbols) in the shared assets, configuring them to support a product with those features.

PLE IS MUCH MORE THAN REUSE

To understand what PLE is, it is important to understand what it is not. A superficial explanation of PLE describes reuse through shared artifact repositories. Yes, there is reuse, and yes, there are repositories, but that is like explaining Project Apollo by starting with powdered orange breakfast beverage. It was there, but was hardly the point.

Many organizations claim, incorrectly, that they are employing PLE when in fact are only practicing reuse and nothing more. And they are practicing a particularly problematic form of reuse called "clone and own."

Figure 4 shows a stylized view of a production shop in which N products are developed and maintained—or, for that matter, acquired. This "shop" could turn out the systems under a PEO's purview, and be run by a single contractor, or a prime with subs, or separately administered programs. In this simplified view, each product comprises a set of artifacts; for example, requirements, design models, source code, and test cases. Each engineer in this shop works primarily on a single product. When a new product is launched, its project copies—clones—the most similar assets it can find, and starts adapting them to meet the new product's needs. Development and acquisition efforts that think reuse is the goal can chalk up impressive metrics to claim success.

But under this kind of reuse, making portfolio-wide changes becomes prohibitively expensive. And portfolio-wide changes are the norm in DoD systems: New hardware, new architectures, new standards, new mission doctrines, new rules of engagement, new systems to interoperate with, new adversaries, and new threats can easily lead to the need to change every system in a family.

To see how clone-and-own reuse can lead to intractable complexity, consider one kind of portfolio-wide change: Defect elimination. Assume that a defect is found in Product B and that the defect is traced to an ambiguous or incorrect requirement in Product B's requirements. The Product B team fixes the error, re-designs as necessary, then fixes the code and test cases before re-deploying Product B. Product B is now healthy again.

But suppose that the defect in Product B's requirements was "inherited" when the Product B team copied the requirements from Product A. Suppose further that the source code for Product N was copied from Product B's (defective) source code, and the test cases for Product N were similarly "borrowed" from Product N's (inadequate) test cases.

To really root out the defect from the entire portfolio, each of the N product teams should really confer with each of the other N-1 product teams. These communication paths are shown in red in Figure 4. This communication obligation imposes an overhead that grows as the square of the number of products. So, in a relatively modest product line of 30 products, almost 900 inter-project communication paths should be activated. This complexity will quickly overwhelm any program office, let alone any engineering staff, and the result is usually exhaustion, a climbing defect rate, out-of-control sustainment cost, and a reluctance or inability to make changes.

This complexity occurs even if reuse levels are as high as possible among the programs; the product line will still collapse under the weight of its "clone and own" reuse strategy. Copy-based reuse gives the copying program a head start, but then loses all of its value as the new program spirals off on its own evolution and sustainment trajectory. Acquisition programs that encourage reuse but not true product line engineering are setting themselves up for sustainment failure.

The automation-centered approach also enables a fourth salient characteristic of 2GPLE: A simplified model for configuration management. The shared assets are configuration-controlled, but the products need not be, since they can be quickly re-generated [11].

A fifth characteristic involves feature languages that facilitate modular and hierarchical product lines developed across organizational boundaries [7]. This allows a system-of-systems family to become a product-line-of-product-lines.

Overall, 2GPLE represents a more clearly formulated methodology that organizations can use directly. It simultaneously generalizes and simplifies concepts from its first-generation roots. Once again, industry and the DoD are paying attention. In addition to 2GPLE projects in industry at large—General Motors, for instance [7]—two multi-billion-dollar high-visibility programs in the Army and the Navy (respectively) are employing 2GPLE to help their Warfighters train and fight, and are seeing substantial benefits in reliability, sustainability, and responsiveness. The two programs are Live Training Transformation and AEGIS.

2GPLE in the Army: Live Training Transformation

In 2010 General Dynamics teamed with BigLever Software (the PLE technology provider) to create the winning proposal for the US Army's Live Training Transformation (LT2) family of training systems. (This contract was the first U.S. Army contract focused specifically on product line engineering as a required part of the solution.)

The United States Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) is in the business of training soldiers and growing leaders by providing responsive, interoperable simulation, training, and testing solutions and acquisition. Its training and testing systems portfolio includes live, virtual, and constructive training packaged in embedded and interoperable products that are fielded and used throughout the world.

LT2 has long been a true software product line, in the sense defined in [3], using first-generation approaches. In 2010 the program made the transition to 2GPLE. LT2 shared assets include the open architectures, common software components, standards, processes, policies, governance, documentation, and more, all leading to a common approach and frameworks for developing live training systems. Examples of the many types of training systems in the LT2 family include Military Operations on Urban Terrain (MOUT), Maneuver Combat Training Center (MCTC), instrumented live-fire range training, and various Joint (that is, inter-Service) training systems.

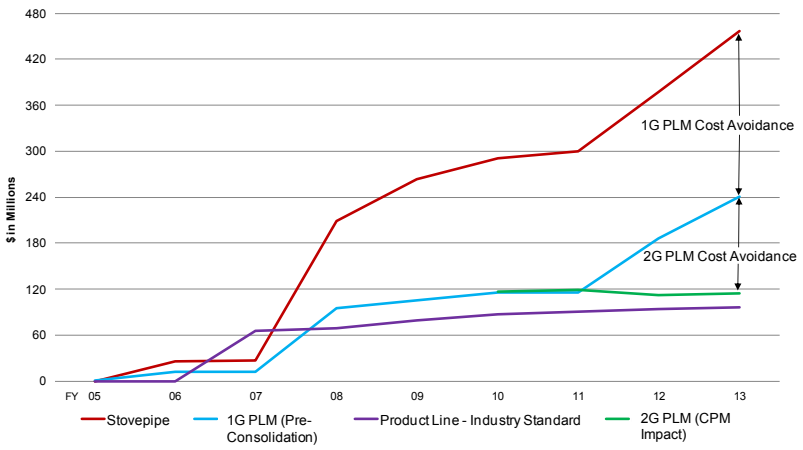

Figure 2. Cost avoidance benefits of product line engineering for LT2

The commonality behind LT2 facilitates the rapid development of new products but also ensures that products across the LT2 product line can communicate and interoperate with each other. This is important because large training exercises need to employ different kinds of training systems working together. The LT2 product line makes use of plug and play components and applications that are common between products, and permits changes, upgrades and fixes developed for 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 lifecycle costs [13].

The LT2 migration to 2GPLE is proving easier than expected. First, a product line culture and high reuse were already in place with the first generation product line. Second, 2GPLE approaches are easier to adopt because they enable non-disruptive and incremental steps to be taken rather than a large "big bang" start-over event. LT2 stakeholders have already enjoyed substantial benefits from LT2's first-generation approach and are therefore more willing to move to 2GPLE.

Maximizing asset sharing has proven to reduce fielding time and minimize programmatic costs, while enhancing training benefits afforded to the soldier. Recognized as the Army's live training standard, the LT2 product line architecture, standards, assets, and common operating environment have been used by more than 16 major Army and Department of Defense live training programs with more than 130 systems fielded.

In addition, LT2's 2GPLE approach is exhibiting the following benefits:

- **More efficient integration of the Army products by the use of common standards and products to meet training and test requirements**
- **Compatibility of objective system and products with evolving capabilities**
- **Wider interoperability before executing subsystem and device production**
- **Reduced total lifecycle costs to include acquisition, development, testing, fielding, sustainment, and maintenance.**

This continuing transformation has generated a significant return on investment to date within PM TRADE's live training system acquisition portfolio. The first generation approaches generated more than $300 million in cost avoidance across the development of live training systems to include Combat Training Centers Instrumentation Systems, Home Station Instrumentation Systems, Instrumented Ranges, and Targetry. The second generation approach, known as Consolidated Product Line Management or CPM in the Army, is projected to save another $200 million over the next two to five years[2].

2GPLE in the Navy: AEGIS Combat System

The AEGIS Combat System is an integrated warfare system deployed on some 100 naval vessels in the U.S. Navy and the navies of key allies across the globe. AEGIS is deployed on deep-water fleet ships, Littoral Combat Ships, and (more recently) U.S. Coast Guard National Security Cutters (NSCs). As the

Aegis Combat System Engineering Agent, Lockheed Martin's Maritime Systems and Sensors Division maintains the Common Product Line (CPL) requirements in a common DOORS database and source code in a Common Source Library (CSL) that is maintained for all product configurations, and they do it using the 2GPLE paradigm.

The primary objective of CPL is to develop once, and build and deploy many times from one set of common assets—principally requirements, source code, and tests. The AEGIS Baseline 9 Common Product Line comprises the requirements and source code that is maintained for all product configurations. CPL supports the US Navy's objective to more quickly field capability as well as the goal of minimizing cost and schedule for delivering computer program capability updates.

The CPL methodology is in high gear for the current AEGIS Baseline 9, which is the foundation for cruiser and destroyer

Figure 3. The Aegis destroyer USS Hopper (DDG 70) launches a missile to intercept a short-range ballistic missile. (U.S. Navy photo/Released)

platforms as well as Land Based Ballistic Missile Defense (BMD). The CPL approach enables the deployment of products from the combat system on the Littoral Combat Ships (LCS) and the US Coast Guard NSCs. It is also the basis for all future domestic and international AEGIS and LCS development efforts.

CPL enables the critical convergence of AEGIS antiaircraft warfare and BMD functionality while providing the fleet with affordable capability and timely upgrades that keep pace with evolving threats. The CPL approach encompasses all phases of the classical V-chart. In the requirements development phase, requirements are consolidated into a single database (using IBM Rational's DOORS tool) for all stakeholder programs using Gears as the variation engine. This approach avoids redundant efforts and requirements capture when managing program-unique databases. Verification of the requirements is also maintained in the DOORS database.

In the software implementation phase, a master software development repository (CSL) is utilized that contains source files, libraries and configuration files that support multiple product configurations. Products comprise common and unique capabilities such that modifications to common configurations are implemented once and feature-based variation is used to automatically include or exclude each capability from a product.

LT2 SPREADS ACROSS THE SERVICES

The hundred-plus systems deployed as members of the LT2 family include these in the Air Force and Marines, as well as other commands within the Army:

CIEDAS—Counter Improvised Explosive Device (IED) After Action Review System (USAF)

The LT2 Homestation Instrumented Training System (HITS) product was heavily leveraged in creating the Air Force's CIEDAS product for convoy counter IED training. An early version of what became the Digital Range Training System (DRTS) Integrated Player Unit (IPU) was used to instrument Air Force convoy vehicles providing multiple in-vehicle video feeds and position/location information to the mobile Exercise Controller (EXCON). Temporary mobile field cameras provided additional video coverage. The LT2 product line HITS software components and Common Training Instrumentation Architecture (CTIA) provided the basis for exercise control, player unit monitoring and control, and After Action Review (AAR) reporting. Common software components provided the video monitoring and editing, and a new rapid AAR capability was developed that allowed an on-going exercise run and an after action review presentation simultaneously with a single operator.

SMS—Soldier Monitoring System (Army—SOCOM)

The Soldier Monitoring System provides safety monitoring of special forces students conducting a land navigation exercise. CTIA and HITS provide the foundation of the exercise control and AAR capabilities of SMS. The player unit radio instrumentation takes advantage of the standard LT2 Player Unit gateway, CTIA provides the architecture and event distribution mechanism, and HITS components provide situational awareness capabilities.

I-TESS II—Instrumented - Tactical Engagement Simulation System II (USMC)

I-TESS II provides the USMC with dismounted instrumentation in support of direct force-on-force tactical training. The LT2 HITS product was used in its entirety as the exercise command and control and after action review capability. Modifications to HITS were created to provide USMC customizations to support their unique style of training. These changes were approved by the LT2 Core Asset Working Group (CAWG) Integrated Product Team (IPT) and absorbed by the LT2 product line.

MC-ITS—Marine Corps Instrumentation Training System (USMC)

MC-ITS was a predecessor to RISCon that provided force-on-force tactical training for the USMC. HITS was used in its entirety as the foundation for this program. Specific new functionality was added to HITS to mainly support USMC IED training and specialized IEDs and IED jammers. The modifications produced by this program have just recently been rolled into the LT2 product line.

RISCon—Range Instrumentation System Control (USMC)

The RISCon program's objective is to reduce sustainment, operational, and enhancement costs of the existing and future Marine Corps Range Instrumentation System Product Line. RISCon leverages the CPM construct of tools (i.e. Gears) and processes to establish and manage a framework for affordable USMC Product Line operation, improvements and deployments. The project leverages the US Army's LT2 Product Line using CTIA. CTIA establishes the framework (protocols, standards, interfaces, etc.) for developing a repository of LT2 core components.

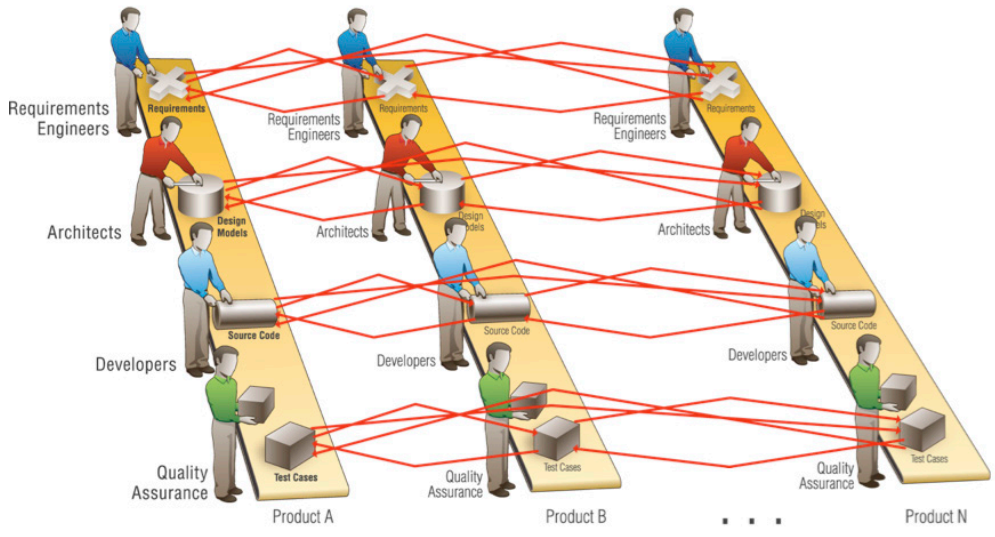

Figure 4. Product-centric development and O(N²) complexity

During the test and verification phase, CPL utilizes a consolidated testing approach to maximize efficiency of common requirements and capabilities. This results in tailored regression testing based on changed functional areas. This also utilizes an integrated test team using common test plans and procedures. Common test efforts are leveraged and consolidated problem reporting avoids duplicate reporting caused by redundant testing. These test benefits are currently being realized as AEGIS baseline 9 prepares for certification.

Organizational consolidation became possible under product line development. Overall program management was consolidated to minimize redundancy and achieve a common program structure and consolidated business rhythm, metrics, and reviews. An engineering product team was established that spans programs to maximize commonality and to drive consistency and design practices. An Engineering Review Board was established as a decision authority to ensure proper CPL behavior at the product level for each of the elements.

The benefits were highlighted when the US Coast Guard made the decision to enter the family with their new National Security Cutter. Once in the product line, they avoided the months it would have taken to implement and verify the hundreds of fixes and upgrades that set their application apart. Instead, the Coast Guard applied their unique feature-based requirements to the CPL DOORS database using Gears, and thus avoided having to apply the specification changes one by one. This resulted in a much quicker deployment of code and requirements for the Coast Guard.

Conclusion

Although this is primarily the story of an Army and a Navy program, LT2 and AEGIS have put down 2GPLE roots in every Service. Aegis has brought the Coast Guard into its product line family. And the hundred-plus LT2 family members include several developed for and in use by the Air Force and Marines.

There are organizational, management, and contracting issues that these programs have had to surmount, but their success shows that those issues are tractable. As a result, they would seem to provide strong evidence that Second Generation Product Line Engineering is an engineering discipline suitable for DoD acquisition programs, across Services and domains. Like its first-generation predecessor methods, it is showing multiple-integer improvements in quality, time to deployment, cost, and engineering productivity.

ABOUT THE AUTHORS

Dr. Paul Clements is the Vice President of Customer Success at BigLever Software, Inc., where he works to spread the adoption of systems and software product line engineering. He was previously at Carnegie Mellon's Software Engineering Institute, where for 17 years he worked in software product line engineering and software architecture documentation and analysis. Clements is co-author of three practitioner-oriented books about software architecture as well as the field's leading text on software product line engineering.

E-mail: pclements@biglever.com
Phone: 512-567-1681

Susan P. Gregg is a Principal Project Engineer for the Lockheed Martin Corporation. She holds a B.A. in Physics from Rutgers University. She has over 30 years experience is systems and software engineering. She is currently the Technical Director for the US Navy's Common Product Line.

E-mail: susan.p.gregg@lmco.com
Phone: 856-359-1636

Dr. Charles Krueger, BigLever founder and CEO, is a thought leader in the product line engineering field with 25 years of experience in software engineering practice and more than 60 articles, columns, book chapters, conference keynotes, and session presentations. Krueger has proven expertise leading product line development teams, and helping establish notable PLE practices in companies such as General Motors, Lockheed Martin, General Dynamics, Ikerlan/Alstom, and three Software Product Line Hall of Fame inductees.

E-mail: ckrueger@biglever.com
Phone: 512-426-2227

Jeremy T. Lanman, Ph.D. is the lead architect for the Common Training Instrumentation Architecture and Live Training Transformation Product Line at the U.S. Army PEO STRI. His professional experience includes 10 years of DOD acquisition and systems engineering of military simulation and training systems. Dr. Lanman received his B.S in Computer Science from Butler University, M.S. in Software Engineering from Embry-Riddle Aeronautical University, and Ph.D. in Modeling and Simulation from the University of Central Florida.

E-mail: jeremy.lanman@us.army.mil
Phone: 407-384-5307

Jorge Rivera is currently working for General Dynamics C4 Systems out of the Orlando facility leading live training efforts and supporting the 2nd Generation Product line instantiation under the CPM contract. His prior experience includes 25 years of DoD Acquisition service with over 15 years of those in the live training domain. As the Assistant Project Manager (APM) LT2, Mr. Rivera championed the LT2 product line and managed the CTIA & FASIT efforts. He earned his B.S. in Electrical Engineering (EE) from the University of Puerto Rico in 1983 and his M.S. in EE from Fairleigh Dickinson University, NJ in 1987

E-mail: jorge.rivera@gdc4s.com
Phone: 407-275-4820

Rick Scharadin has over eighteen years of Senior Program Management experience related to complex large scale system development, open architecture designs, software product line development, product integration, test and delivery for various Navy Aegis Baselines. He has accumulated over his career with Lockheed Martin 12 service awards, including manager of the year in 2001. Rick has a BS in Electrical Engineering from Penn State and a MS in System Engineering from Stevens Institute.

E-mail: richard.w.scharadin@lmco.com
Phone: 609-326-4685

James T. Shepherd works as a Lead Architect on the Aegis software common product line for Lockheed Martin MS2. He holds a B.S. in Computer Science from Montclair State University and an M.S. in Computer Science from Drexel University. He has more than 25 years experience in systems and software engineering of mission critical applications for the US Navy.

E-mail: james.t.shephard@lmco.com
Phone: 609-326-4685

Andrew J. Winkler is a Principal Engineer at Lockheed Martin and has over 15 years experience working on large scale systems, including the AEGIS Combat System and the DDG1000 C3I system. Most recently Andrew has held the role of System Architect for the US Navy's AEGIS Common Product Line. Andrew has a BS and MS in physics from the University of Vermont.

E-mail: andrew.j.winkler@lmco.com
Phone: 856-914-6318

REFERENCES

1. Bachmann, F., Clements, P. "Variability in Software Product Lines," Technical report CMU/SEI-2005-TR-01, Software Engineering Institute, 2005.
2. Clements, P. and Bergey, J. The U.S. Army's Common Avionics Architecture System (CAAS) Product Line: A Case Study, Technical Report CMU/SEI-2005-TR-019, September 2005.
3. Clements, P.; Northrop, L. Software Product Lines: Practices and Patterns, Addison-Wesley, 2002.
4. Cohen, S., Dunn, E., Soule, A. , Successful Product Line Development and Sustainment: A DoD Case Study, CMU/SEI-2002-TN-018, September 2002.
5. 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.
6. FedSmith.com, "Billions Wasted…," <http://www.fedsmith.com/article/313/billions-wasted-dod-because-duplicate-business-systems.html>
7. 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.
8. Guertin, N., and Clements, P., "Comparing Acquisition Strategies: Open Architecture vs. Product Lines," Proceedings of the 2010 Acquisition Research Symposium, Monterey, May 2010.
9. Jensen, Paul. (2009). "Experiences with Software Product Line Development." CrossTalk 22, 1 (January 2009): 11–14.
10. 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.
11. Krueger, C. "The Systems and Software Product Line Lifecycle Framework," BigLever Software Technical Report #200805071r3, 2010. http://www.biglever.com/extras/SplLifecycleFramework.pdf.
12. Linden, Frank J. van der, Schmid, Klaus, Rommes, Eelco. Software Product Lines in Action, Springer, 2007.
13. Rivera, J., Samper, W., Clinger, B. (2008). Live Training Transformation Product Line Applied Standards For Reusable Integrated And Interoperable Solutions. Paper No. 483; MILCOM 2008.
14. Software Engineering Institute, "Catalog of Software Product Lines," <http://www.sei.cmu.edu/productlines/casestudies/catalog/index.cfm>
15. SPLC Product Line Hall of Fame, <http://splc.net/fame.html>
16. UAV Forum, Librarian's Desk, <http://www.uavforum.com/library/librarian.htm>

NOTES

1. This is an adaptation of the Software Engineering Institute's definition of a software product line, which is a product line in which software plays a central role in the systems :3]
2. These figures are based on industry standard estimates of code cost, and are calculated assuming that post-deployment software support constitutes 70% of development cost and a life expectancy of 10 years. See [5] for a more detailed explanation.