Lessons from AEGIS: Organizational and Governance Aspects of a Major Product Line in a Multi-Program Environment

Susan P. Gregg
Rick Scharadin
Lockheed Martin
199 Borton Landing Road
Moorestown, New Jersey 08057 USA
+1 609 326 4685
susan.p.gregg@lmco.com
richard.w.scharadin@lmco.com

Eric LeGore
U.S. Navy/PEO IWS
1333 Isaac Hull Ave SE
Washington Navy Yard, DC 20376
+1 202 781 2251
eric.legore@navy.mil

Paul Clements
BigLever Software
10500 Laurel Hill Cove
Austin, Texas 78730 USA
+1 512 426 2227
pclements@biglever.com

ABSTRACT

This paper tells the story of the AEGIS Weapon System product line and how it evolved from a series of standalone software programs with no sharing into a true systems and software product line. The paper focuses on the strong internal and external *governance* of the product line. The need for strong governance is brought about by the strong role that the AEGIS customer community plays in oversight of design, development, and procurement. The paper recounts the product line's beginnings, and describes how the product line is operated today. Organizational issues, measurement issues, and governance issues are covered, along with a summary of important lessons learned about operating a product line in an environment of strong competing interests.

Categories and Subject Descriptors

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

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, product line governance, AEGIS, Navy, command and control, combat systems

1. Introduction

This paper tells the story of the AEGIS Weapon System and how it evolved from a series of standalone programs with no sharing into a true systems and software product line that today ranks as one of the most important and successful examples of product line engineering in the U.S. Department of Defense [3].

SPLC '14, September 15 - 19 2014, Florence, Italy

Copyright 2014 ACM 978-1-4503-2740-4/14/09…$15.00.
http://dx.doi.org/10.1145/2648511.2648541

US Navy Aegis Cruisers & Destroyers
US Navy Littoral Combat Ships
US Coast Guard Nat' l Security Cutter
Aegis Ships for International Navies

Figure 1 AEGIS sea platforms include cruisers and destroyers in the U.S. and allied navies, as well as U.S. Littoral Combat Ships and U.S. Coast Guard National Security Cutters.

What sets this narrative apart from other product line studies is its focus on internal and external *governance*. By governance, we mean the practices by which the product line is managed and controlled. With AEGIS, management and control come not just from the development organization, as might be expected, but from stakeholders external to that organization. Because each ship class is a major undertaking with national and international visibility, it has a strong constituency of its own. Customer pressures for development specific to their individual needs are extremely strong. And yet both the customer organizations and the development organization understand the overriding advantages of the product line approach and together have worked very hard to put in place a strong product line governance structure. While we provide enough background information to put the governance story in context, it is the governance that is the focus of this paper.

The paper begins with an overview of AEGIS in Section 2. Section 3 narrates the history of the product line from its beginnings as a set of independent software programs through the development and maturation of the product line. Section 4 gives a snapshot of the product line processes in play today. Section 5 describes the organizational structure put in place by the development organization to support the product line's goals. Section 6 describes the strong and elaborate governance mechanisms that are necessary to ensure the continued vitality of the product line. Section 7 describes some of the metrics that are collected specifically having to do with the product line approach. The AEGIS story would not be interesting and its governance structure not worth understanding if the program were not successful; Section 8 demonstrates why it is. Those heavily involved with AEGIS look back at a number of important lessons learned; the best of these are shared in Section 9. Finally, Section 11 closes the paper by looking to the future of AEGIS.

2. What is AEGIS?

The AEGIS Combat System, named after the mythical shield of Zeus, is a highly integrated total ship combat system. AEGIS cruisers and destroyers constitute the majority of the U.S. surface Navy and will continue to form the core of the surface fleet for the next several decades. The AEGIS Combat System is capable of simultaneous warfare on many fronts: anti-air, anti-surface, anti-submarine, and strike warfare [8]. AEGIS is deployed on some 100 naval vessels in the U.S. Navy, navies of key U.S. allies across the globe, vessels of the U.S. Coast Guard[1], and even land-based ballistic missile defense installations (Figure 1). AEGIS is a system that protects assets from airborne attack from aircraft or missiles. It detects airborne threats, plans how to engage them, and launches missiles to intercept and neutralize them.

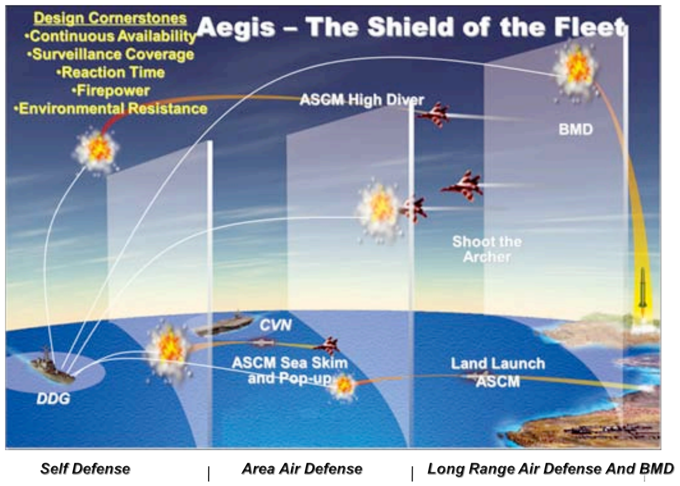


Figure 2 This viewgraph from the AEGIS program highlights the missions of AEGIS. "ASCM" stands for anti-ship cruise missile. "DDG" and "CVN" signify destroyer and aircraft carrier, respectively.

The mission of AEGIS, summarized in Figure 2, includes

- self-defense (protecting the host platform from attack),
- area air defense (for example, protecting a naval task force that includes the host platform), and
- long-range air defense and ballistic missile defense (for example, protecting a geographical area from long-range ballistic missiles).

At the heart of the AEGIS Combat System is the AEGIS Weapon System (AWS), which is a centralized, automated, command-and-control and weapons control system that was designed as a total weapon system, from detection to kill. The prime contractor for the AEGIS Weapon System is Lockheed Martin's Mission Systems and Training Division. There, some 1500 people work on the AEGIS program where, among other things, they maintain the over one hundred thousand AWS requirements and over ten million lines of source code used by AEGIS (some 1.8 million SLOC in the last major upgrade alone). Lockheed Martin employs 116,000 people worldwide and is one of the world's largest defense contractors.

Several different U.S. Government agencies make up the customer side of the AWS picture.

- The U.S. Navy is represented by a department called Program Executive Office (for) Integrated Warfare Systems (PEO-IWS). PEO-IWS oversees development and delivery of dozens and dozens of Navy surface combat systems (for foreign navies as well as the U.S. Navy), from guns and radars up to entire integrated combat systems such as AEGIS.
- The Coast Guard oversees procurement of its AEGIS-based software.
- The U.S. Missile Defense Agency, whose mission is to develop and deploy a layered ballistic missile defense capability for the United States, counts AEGIS platforms among its missile defense assets and so also plays an oversight and customer role.

Each member of the AEGIS product line – the integrated weapon system for a guided missile cruiser, for example – is a large, complex, and expensive system in its own right. The combat system for a naval vessel is in effect the whole reason for the very existence of the ship. Each ship occupies a place on the national and international stage, giving its program office the motivation to push for ship-specific concessions that subjugate the good of the overall product line.

This sets AEGIS apart from, say, a product line of mass-produced and mass-consumed products in which individual customers do not play a large role or may even be anonymous. With AEGIS, the customer agencies are heavily involved in many development-related decisions that affect cost, capability, requirements, and even high-level design.

This, plus the sometimes-conflicting priorities of the different customer-side agencies, makes a strong governance regime essential to avoid stalemate and chaos.

Architecturally, AEGIS is a system of systems [9] comprising elements that include the AEGIS Display System, Command and Decision, SPY Radar, Weapon Control System, AEGIS Training, Mission Planner, and Operational Readiness. These elements constitute the fire control loop that enables AEGIS to fulfill its mission.

The AEGIS software architecture, shared by all variants in the product line, is layered; application components populate the top layer. Components communicate by message passing (Figure 3). In addition, the architecture is *open*, meaning that the layers and the components have published interfaces that conform to organizational and/or industry published standards.

[1] The Coast Guard vessels employ portions of AEGIS, as we will see in Section 3.

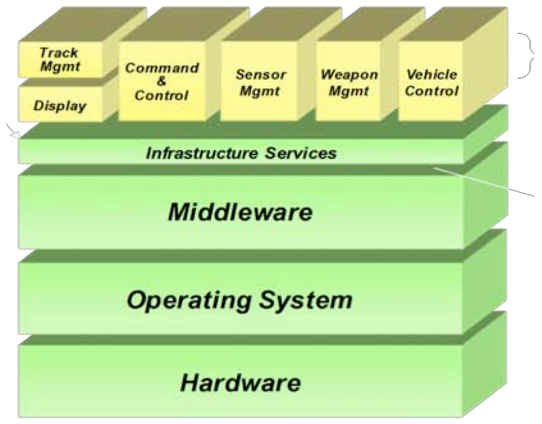


Figure 3 Layered view of the AEGIS product line software architecture

3. Beginnings of the Product Line

AEGIS began as a family of separate programs. In the early 2000s, there were over 1500 people working on nine major programs for the U.S. Navy, each of which was concerned with one or more ships in the AEGIS family. During this time, each program operated in an isolated manner. There were independent management structures, multiple review teams, varying processes and tools, redundant program plans, different architectures, and multiple independent requirements and source libraries.

To create a new program (or baseline), all specifications and source code from a previous program (sometimes still in development) would be copied with new development and maintenance then conducted independently and in parallel. Programmatic and technical decisions could be made for one baseline independent of other baselines, which was in fact seen as an advantage in terms of schedule and technical flexibility.

Of course, a large disadvantage stemmed from redundant work efforts, particularly in the area of requirements and code maintenance.

In the mid-2000s, a number of business and technical forces began to bear on AEGIS that encouraged and enabled its migration to a true product line.

First, the Navy let it be known that paying to fix the same defect (or make the same enhancement) multiple times (once for each program) was problematic and would soon become cost-prohibitive. Lockheed Martin responded by creating a common defect approach for requirements and software that leveraged defect repair effort across all of the programs. “Fix it once!” which would become a major theme of AEGIS, was now in play. From this early effort at sharing came the realization that forceful driving was required – the separate Navy customers driving the separate AEGIS programs needed to coordinate and consolidate the prioritization of their defect repair needs. This was an embryonic form of the cross-program governance that would prove to be so critical to this product line’s success.

Also about this time, the Navy was strongly pushing its contractors to follow technical approaches that encouraged reuse, opened up competition, and employed commercial off-the-shelf (instead of purpose-built) hardware and software. This “Open Architecture” initiative touted the benefits of an architecture designed as a set of modular components with published interfaces. In theory, any contractor could bid to provide any of the components, thus driving up competitiveness and driving down development and maintenance cost to the Navy.

As an example of the Open Architecture push, the Deputy Chief of Naval Operations for Warfare Requirements and Programs established the requirement to implement Open Architecture principles across *all* surface Navy combat systems in December 2005 [11]. This directive followed many other OA initiatives, Navy instructions, and acquisition regulations ([10], for example).

Meanwhile, the product line movement was gaining steam. Clements and Northrop [5] have written definitively that reuse and a componentized architecture (two pillars of the Open Architecture approach) do not constitute a product line. However, there are synergies between the two philosophies (as outlined in [6]) that, together, constitute a strong acquisition approach that some in the Navy were touting.

For example, a 2008 article in the influential *Journal of the American Society of Naval Engineers* stated categorically that Open Architecture was not sufficient to meet Navy goals, and that a true product line approach was needed [11].

In any case, the Open Architecture movement provided a strong foundation for the AEGIS product line by

- strongly encouraging a modular architecture
- strongly discouraging one-at-a-time development of system components that could and should be shared across programs
- strongly signaling an intent to increase competition, and
- making it clear that clone-and-own systems, even successful ones, were becoming unaffordable.

Lockheed Martin, clearly understanding the competitiveness implications, made the commitment to become the most competitive of all of the potential contractors, so as to maintain their role in AEGIS. To do this, they adopted product line engineering as their development paradigm. By 2009 they had adopted Gears [2] as the tool to configure their shared assets and were employing the factory-based product line approach shown in Figure 4. By 2009 they were building the largest requirements and code baseline in AEGIS history. They had, by this time, merged separate anti-aircraft warfare and missile defense software components from the various separate software programs into a common integrated air and missile defense system that could be configured to support any of the ships in the family.

AEGIS AWS was now a fully mature product line. It enjoys common oversight and governance, unified review teams, merged development teams, common processes and tools, common management plans, a shared architecture, and a single specification and code base feeding into an automated configurator.

In 2011 Lockheed Martin was awarded the contract for the third member of the Littoral Combat System (LCS) ship class. In previous years, they would have spun this program off on its own development and maintenance trajectory after copying all of the relevant assets: requirements, code, and tests. Now, however, it was incorporated as a new member of the product line.

In 2011 the U.S. Coast Guard (with Navy encouragement) made the decision to enter the AEGIS product line family by deploying the AEGIS Display System and Command and Decision components on 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 the product line had already implemented. Instead, the Coast Guard applied their unique feature-based requirements to the product line's requirements database and, using Gears, derived the requirements for this new platform. This resulted in a much quicker deployment of code and requirements for the Coast Guard – weeks instead of months – and sent a strong message that AEGIS was on the right track with its product line approach.

Since 2011, the Navy has provided maintenance funding to fuel the paradigm of "Fix it once, pay for it once," as opposed to funding separate ship programs to each fix the same defects over and over again, signaling that they understand the power of the product line approach. Section 8 will discuss the payback to the Navy for their support.

4. Lockheed Martin's Product Line Approach

Lockheed Martin views its primary objective as developing once, and building and deploying many times from one set of common assets – principally requirements, source code, and tests. Feature-based variation in requirements and code enables building a member of the product line with or without a specific capability.

First, some terminology: Lockheed Martin calls a member of its product line a *configuration*. A configuration might be, for example, a weapon system to be deployed on specific destroyer. The product line approach, then, exists to produce configurations for customers.

Figure 4 illustrates the basic concept. Shared assets on the left (only a few examples of which are shown) are imbued with variation points. A variation point is a place where a shared asset needs to differ based on whether a feature has been selected or not for a configuration; variation points are defined in terms of features. A feature profile, describing a configuration in terms of the features it exhibits, is fed to the configurator, which configures the shared assets by exercising their variation points to produce a suite of asset instances specific to the needs of that configuration.

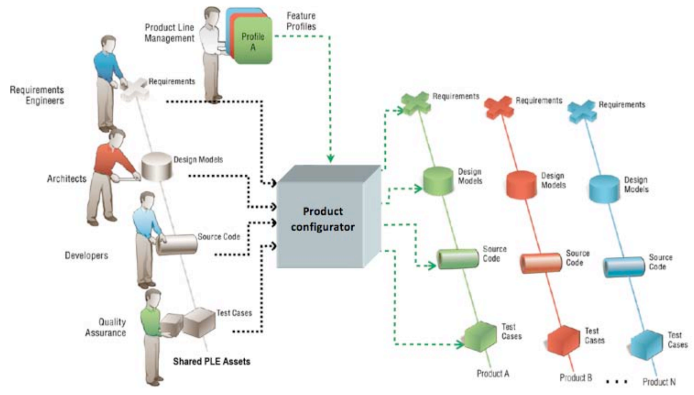


Figure 4 Basic concepts of the feature-based product line approach: A configurator configures shared assets (such as requirements, code, and tests, shown on the left) to configuration-specific instances according to the feature profile of the product line member being built.

Each configuration has a profile that identifies which capabilities (modeled as features) are included. This method facilitates profiles being updated as capabilities are matured and ready to be deployed in any given configuration.

This configurator paradigm works in concert with other variation mechanisms [1], such as frameworks, plug-in components, configurable build scripts, site-specific config files, platform adaptation data, and (at run-time) dynamic registration of services.

Figure 5 shows how this works for software components. The coarsest-grained variation point is to include or exclude an entire software component depending on whether or not the feature(s) provided by that component are included in a configuration or not. If a component *is* included, it can be further varied by exercising variation points inside, again based on feature choices.

Lockheed Martin calls its factory the "Common Source Library," or CSL. CSL is broadly defined as the set of tools and processes required to develop, store, and maintain both requirements and source code to support product line development.

Beyond this paradigm for achieving variation, Lockheed Martin considers the following as "pillars" of their product line process:

- **Common shared assets:** For requirements, CSL employs a common specification repository (a DOORS database) that contains all requirements for all programs/baselines, with varying requirements captured in feature-based variation points. This model allows for multiple baselines to share requirements while having the flexibility for each baseline to have unique requirements as well.

  For code, a master software development repository is utilized that contains source files, libraries, and configuration files that support multiple configurations. Configurations 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 configuration.

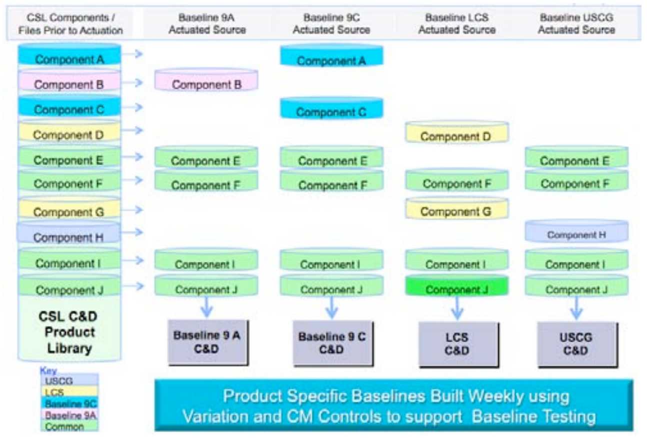

CSL Components / Files Prior to Actuation
Baseline 9A Actuated Source
Baseline 9C Actuated Source
Baseline LCS Actuated Source
Baseline USCG Actuated Source
Component A
Component B
Component C
Component D
Component E
Component F
Component G
Component H
Component I
Component J
CSL C&D Product Library
Baseline 9 A C&D
Baseline 9 C C&D
LCS C&D
USCG C&D
Key
USCG
LCS
Baseline 9C
Baseline 9A
Common
Product Specific Baselines Built Weekly using Variation and CM Controls to support Baseline Testing

Figure 5 The software view of the factory, showing inclusion/exclusion of various components for different configurations. An included component can occur in different forms depending on how its variation points are exercised.

  During the test and verification phase, CSL utilizes a consolidated testing approach to maximize efficiency of common requirements and capabilities. This results in tailored regression testing based on changed functional areas. CSL 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.

- **Regular, predictable build rhythm.** CSL releases three builds a year, one in each of January, May, and September. This so-called "1-5-9" rhythm brings great stability to the program. Everyone, inside CSL or out, can expect and plan for the next release. Inside Lockheed Martin, build meetings are held weekly, to make sure the next release is on track. Not every customer configuration in the product line is required to accept every release; each makes its own decision according to operational needs and the content of the particular build.

- **Requirements review cycle.** As in all product lines, changes to requirements (whether to add new capabilities or address defects) that are made for one configuration may have intended and unintended impacts to other configurations, and must be reviewed across the product line with that in mind. A rigorous Requirements Review Cycle for programs is held in March, July, and November (a "3-7-11" rhythm) and is a joint Lockheed Martin/Government exercise.
- **CSL governance (**beyond the regular build rhythm and the requirements review cycle mentioned above, which are part of the governance regime in their own right). This is discussed in Section 6.

5. CSL Organizational Structure

Organizational consolidation within Lockheed Martin became possible (and, as we will see under the "Lessons Learned" section, essential) under product line development.

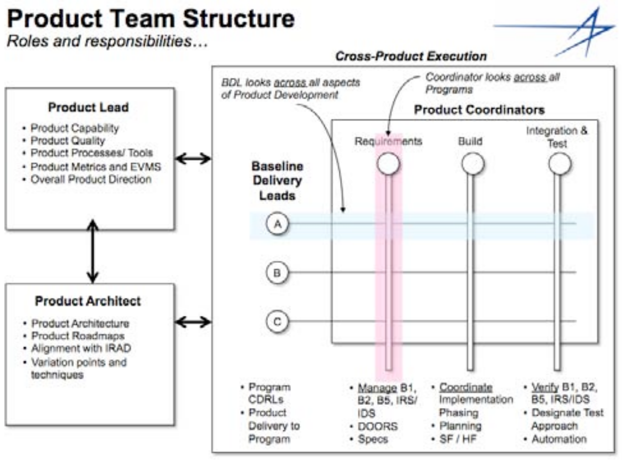
In the narrative that follows, a *product* refers to one of the basic elements of the AWS, mentioned in Section 2: The AEGIS Display System, the SPY radar, the Mission Planner, and so on. Each of these is developed as a product line in its own right, using the approach outlined in the previous section. AWS is a system of systems, built as a product line of product lines.

Lockheed Martin transitioned from traditional Integrated Product Teams (IPTs) for a baseline to a number of product-oriented teams that support all programs in the AEGIS family. The overall goal was to consolidate program management to minimize redundancy and achieve a common program structure and consolidated business rhythm, metrics, and reviews.

Specific aspects of the product-line-focused team approach include:

- **Product Leadership Team (PLT):** Full accountability for products is assigned to the Product Leadership Team (PLT). The PLT is responsible for delivery of the product to multiple stakeholder programs.
- **Baseline Delivery Lead (BDL):** In place of the IPT lead is a Baseline Delivery Lead (BDL) who is an integral part of each product team but has more of a baseline/program focus. Leads have been identified for requirements, software build coordination, and Integration and Test (I&T) activities. These leads ensure a product focus throughout each stage of the product life cycle. Sub-product component leads are also established.

  BDLs maintain cost account status and financial reporting, and oversee monthly in-depth product reviews. All programs participate in the product reviews instead of each program conducting a monthly in-depth review. BDLs also oversee product metrics, focusing on product health, affordability, and productivity. These include variation metrics to ensure there was no capability leak in the requirements or software.
- **Product architect:** There is a product architect for each major element of AEGIS who has technical responsibility for the element or product. The product architect has cognizance of new configurations coming into the CSL, and will provide design considerations to facilitate bringing new capability into the product portfolio while preserving product core to maximize reuse.
- **MB-SEIT:** A collaborative Multi-Baseline Systems Engineering Integration and Test (MB-SEIT) team ensures key CSL aspects of system and software architecture. This SEIT has decision authority to ensure proper CSL behavior at the product level. This board drives consistent methodologies to ensure each product can be built for each program configuration. The SEIT controls the single repository of requirements as well as the software, with their built-in variation points that the configurator exercises.
- **Product manager:** A single product manager is the single cost account manager for all control accounts. Sub-contract management has also been moved to the product manager to streamline communications with the teams.



Product Team Structure
Roles and responsibilities...
Cross-Product Execution
BDL looks across all aspects of Product Development
Coordinator looks across all Programs
Product Coordinators
Product Lead
• Product Capability
• Product Quality
• Product Processes/ Tools
• Product Metrics and EVMS
• Overall Product Direction
Requirements
Build
Integration & Test
Baseline Delivery Leads
A
B
C
Product Architect
• Product Architecture
• Product Roadmaps
• Alignment with IRAD
• Variation points and techniques
• Program CDRLs
• Product Delivery to Program
• Manage B1, B2, B5, IRS/IDS
• DOORS
• Specs
• Coordinate Implementation Phasing
• Planning
• SF / HF
• Verify B1, B2, B5, IRS/IDS
• Designate Test Approach
• Automation

Figure 6 Overview of CSL product team structure

Figure 6 summarizes. After the product teams were established, the team makeup and operations were assessed and modified. Roles and responsibilities were simplified. Component leads were consolidated and the requirements and test lead focus shifted based on the phase of development. The team consolidated weekly meetings, increased training, and improved metrics collection.

6. AEGIS Product Line Governance

This section discusses the all-important governance aspect of the AEGIS product line, the focus of this paper. We distinguish between internal governance, which is largely carried out by Lockheed Martin as part of its normal development activities, and external governance, in which agencies representing the consumers of the individual combat systems exercise oversight and influence.

6.1 Internal Governance

Programs in CSL are in all stages of development, including programs brand new to the AEGIS family, which is a major reason why internal governance is so important.

The "pillars" of the product line approach described in Section 4 and the organizational structure described in Section 5 certainly make up important aspects of internal governance. In addition to those, internal governance includes weekly meetings to monitor and manage product line execution.

The advent of CSL and, with it, the product line approach led to a consolidation of meetings and reviews. Since there is one build schedule, there is one weekly software build meeting for all products and programs. There is also one weekly coordination meeting with program managers and technical leads from all the programs, and a weekly cross-program MB-SEIT meeting where technical topics are discussed. The requirements review cycles and the consolidated Program Management Team (PMT) have led to a consolidation of meetings on the customer side as well. There is also a monthly product review.

Internal governance can be seen as comprising:

- **Program planning.** The program planning level provides approval and direction on the configurations' desired capabilities, which will be documented in a capability fielding plan. The plan will provide approval to build new capability on a development branch, and direction to merge that capability into the CSL mainline, as well as direction on which builds will be used to support the road to certification.
- **Program execution.** The program execution aspect involves approving product work packages and making a decision as to the maturity of the capability for deployment to ships. The program execution level will produce the artifacts to support approval and decision points by the strategic and program planning levels of governance. The program execution level will make a recommendation, including test results and other supporting information, on which CSL branch or mainline load should be used to support the road to certification.

6.2 External Governance

Early in 2011 confusion erupted regarding product line development in the CSL. The confusion centered on who in the government was giving direction to Lockheed Martin, by what authority, and what direction was being provided. There was, in the words of one participant, "a lot of angst." With multiple contracts, different lines of funding, and competing resource needs, it became quickly apparent that the needs and desires of each program office needed to be coordinated and communicated to the developer with one voice. In other words, governance was required to coordinate between the various government program offices and the developer. A major role of external governance is to resolve or at least mediate the tension inherent in PLE but especially inherent in a product line this large and complex and given the importance of the capabilities of each of the configurations. That tension is the specialized interests of the individual programs versus the overall good of the product line at large.

While Lockheed Martin worked to establish processes to manage a coordinated planning approach and day-to-day development activities to serve multiple masters, the government needed to put in place a structure and associated processes to ensure clear and consistent direction was being provided to maximize the probability of success for all programs. This structure required support for decision making at three levels: Strategic, Programmatic, and Technical (see Figure 9). To this end, three decision-making bodies were created along with a set of governing artifacts.

- **Technical**: From a technical perspective, early expectations were that all code changes, whether new development or maintenance, i.e. defect fixes, would be conducted on a single set of source, i.e. the mainline. It became quickly apparent that not all program offices were satisfied with the risks associated with this approach. Programs nearing the end of their development had little appetite for allowing new development efforts, sometimes widespread and complex, to put their mature code at risk. What was required was the ability to assess the risks of each development effort and provide mechanisms for those high risk development efforts to proceed without putting undue risk to other programs using the same products and source files. The solution was to provide for three avenues for code development: Mainline Development, Development Branches, and Event Branches. Additionally, a process for assessing risks and deciding where to conduct each new development was required. To this end a Joint Engineering Review Team (JERT) was established and co-chaired by two lead system engineers from the government, with technical representation by all product users. The charter of the JERT is to (1) provide direction on the conduct of development efforts; and (2) provide guidance to the developer in all technical matters impacting more than one program. New development efforts are brought before the JERT with supporting data in order to conduct a risk assessment. Those developments deemed to be of low risk are allowed to develop in the Mainline while those assessed as posing a significant risk to one or more programs are designated for execution in a Development Branch. (For the sake of completeness we note that Event Branches are a special type of development path and are not typically subjected to cross-program control. Event Branches are a risk mitigation mechanism to allow maturing configurations, i.e. all development is complete, to exit the Mainline and continue their final grooming to support an upcoming milestone event, such as Combat System Certification.) Additionally, a date is set for the developer to come back to the JERT to assess the readiness of the development effort to terminate the development branch and continue development in the Mainline. This decision point is referred to as a Merge Decision and is supported by a fixed set of criteria and associated data. Issues that cannot be resolved at the JERT level are escalated to the programmatic decision-making body. To date, no issues have required escalation.
- **Programmatic**: To address the programmatic perspective, a Joint Program Management Team (JPMT) was established with four co-chairs (GS-15[2] level program managers) and representation by all product users. In practice, the JPMT serves as the tactical decision maker for all cross-program issues. The charter of the JPMT is to (1) render schedule and funding decisions; (2) apply programmatic considerations to JERT recommendations and turn those recommendations into decisions; and (3) provide a means of escalating unresolved technical issues from the JERT. Issues that cannot be resolved at the JPMT level are escalated to the strategic decision-making body. To date no issues have required escalation.
- **Strategic:** Strategic decision making is the charter of the Major Program Manager (MPM) Board. This board is chaired by three O-6[3] program managers with representation by all product users. Future baseline content and strategic direction is set by the MPM Board. The board reports directly to its members' respective flag officers (generals or admirals).

To facilitate documentation and communication of product

[2] GS-15 is a U.S. civil service rank of considerable management-level seniority.

[3] O-6 is a service-independent designation of rank. It is equivalent to a Navy Captain.

development decisions, a set of governing artifacts was defined. These artifacts serve to combine the strategic, programmatic, and technical decisions and document those as official planning documents:

- **New Development Fielding Plan.** This controls planning for new and current functionality by explicitly mapping those capabilities to baselines/configurations. This provides the developer with up-front information to make intelligent and efficient design decisions. This plan documents what new capabilities the government wants available to all baselines as opposed to any one specific baseline. As it is widely accepted that universal capabilities are less complex and therefore less costly to design and code than variable capabilities, this plan allows government dollars to be spent most efficiently.
- **Branch and Merge Plan.** This provides government control over execution of development. The plan documents the development strategy and assigns it to a Development Branch or the Mainline, as well as planning the merge point for Development Branches. Figure 7 shows a sample.

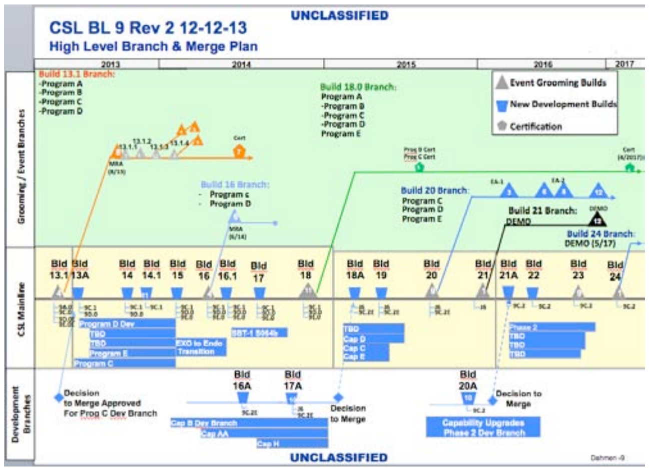


Figure 7 Sample Branch and Merge Plan

- **Build Plan.** This spans all baselines/configuration and controls what functionality is in development and when it will be delivered. It time-phases all requirements allocated to software into the master build rhythm. Signature approval ensures changes, associated rationale, and the impacts are understood and approved by product users. See Figure 8.

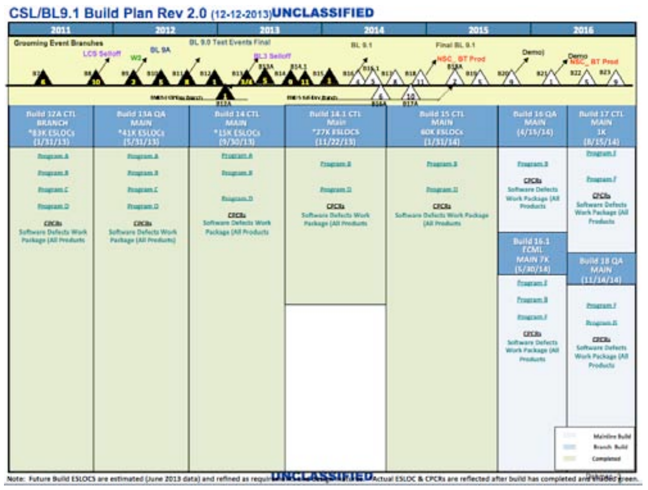


Figure 8 Sample Build Plan

These three artifacts, taken together, broadly define the vast majority of all of the work in the CSL and represent the collaboration, cooperation, and compromise among all of the cognizant stakeholders.

The government, representing all of the "consumers" of AEGIS, has also instituted a cost-sharing approach to equitably allocate the cost of fixing a defect under the "Fix it once!" paradigm. If a program introduces an upgrade or new capability, it pays for it. Other programs are free to pick it up, or not, as they wish, but they pay for any testing that is required and unique to their context. After a development is complete and time has elapsed, newly found defects become difficult to associate with any one program. In these cases all programs pitch in to pay to have the defect corrected. Lockheed Martin gets a special funding account to fix all defects, across the entire product line, that are not related to unique capability content in development. Any program receiving special development funding pays for defects in that development, up to its demonstration milestone, at which point the cost-sharing approach kicks in.

The external governance scheme outlined in this section was crafted and put into place over the period of an entire year. It has since been codified in an Instruction (the Navy equivalent of a policy directive) signed by the Program Executive Office Admiral.

The Navy feels that the underlying product line approach has served the Navy and its partners well to this point. To date, this governance structure has proven sufficiently flexible to adapt and address any circumstances and scenarios arising that were not explicitly foreseen.

7. Metrics

Metrics and measurements are as important for product line efforts at least as much, if not more so, than for conventional development projects [12]. AEGIS, like any large-scale software and systems development effort, collects a broad set of measurements and then monitors trends to identify hot spots of concern.

A particular focus is to track and monitor defects of any kind, especially requirements defects and code defects. They also track whether defects tend to stay confined in one life cycle phase or "leak" across and proliferate into other phases.

Carefully tracking defects and their resolution supports an important product line goal of the AEGIS program, which they characterize with the anthem of "Fix it once!". Thorough understanding of a defect allows the determination of how much testing the fix must undergo. If every program uses the fix in a common way, then testing the fix once time may suffice. If only some of the programs use the fix in a common way, then those programs can share the burden of testing it, even if the other programs must test it for themselves. As we shall see in Section 8,"Fix it once!" accounts for some of the AEGIS program's remarkable cost savings results.

n 2013, a new kind of defect was added to the list of defects tracked. A *variation defect* is unique to the product line engineering paradigm. Under the approach that Lockheed Martin employs for AEGIS [7], a variation defect can be

- an error in a feature model; for example, omitting a feature or a flavor of a feature needed to capture a specific variation among configurations.

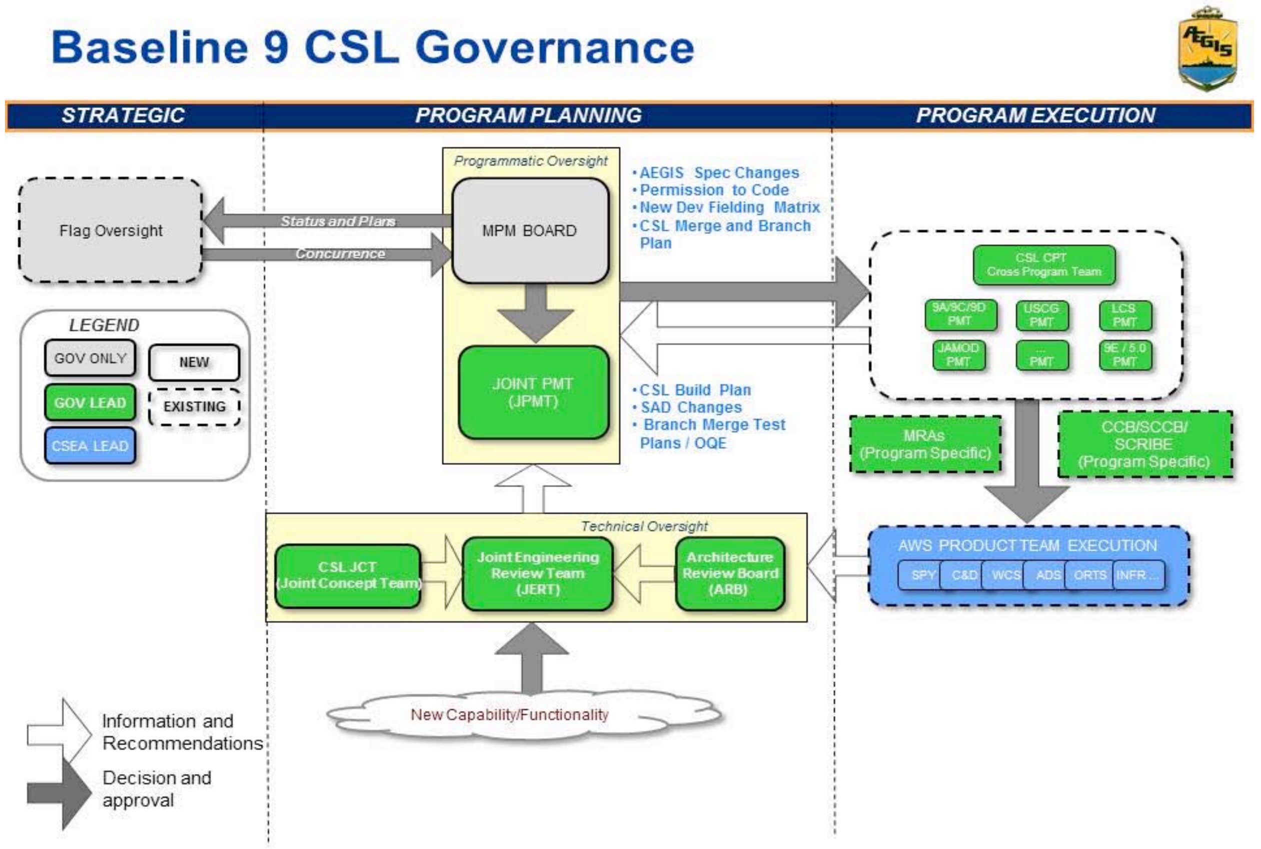


Figure 9 CSL governance structure. "Baseline 9" refers to the current version of the CSL.

- an error in a feature profile; that is, incorrectly making feature choices that define a particular configuration. This kind of error can erroneously place unwanted capability into a configuration, put the wrong flavor of a particular capability into the configuration, or incorrectly omit needed capability from a configuration.
- an error in a shared asset's variation point logic. This kind of error incorrectly exercises a variation point in a shared asset such as requirements or code, and causes an incorrect instantiation of that asset to be produced for the product undergoing a build.

Tracking and understanding variation defects has become very important as AEGIS has joined navies of U.S. allies around the world. Many AEGIS capabilities are under extremely strict export control restrictions, and inadvertently putting the wrong capability on a foreign ship comes with severe consequences. Towards this end, Lockheed Martin and the Navy have instituted a formal auditing procedure, underpinned by a strong variation management discipline, that takes into account the product line engineering paradigm in use and the possible nature of variation defects to avoid [4].

Metrics that focus on defects are one of a class called "process health" metrics. These metrics help Lockheed Martin understand how well they are applying their chosen development processes in the product line context. Defects are only one kind of process health metric; another example is conformance to the specified systems and software architectures.

By following measurement trends over time, the AEGIS program is able to identify systemic problem areas. For example, a component that consistently is involved with a high number of defects may be an excellent candidate for re-design and re-engineering. The AEGIS Engagement Manager (a component that prioritizes threats and plans the tactical response) was one such component. Showing up as a defect hot spot, the component, upon examination, revealed unacceptably high complexity measures. It was re-written, and is now a well-behaved plug-and-play component.

8. Indicators of Success

Like all product lines, AEGIS has indicators of success to point to the efficacy of its approach and, like all product lines, some are quantitative and some are more experience-based.

Quantitatively:

- Prior to implementing product lines using CSL, every code defect fixed in one program that had implications for other programs had to be fixed multiple times. (Clone and own never copies from a "perfect" system but always one in development – hence, the defects are copied too!) Eliminating the need to fix a defect in multiple libraries provided substantial savings to the various government program offices (AEGIS, LCS, FMS, MDA, Coast Guard). The combined savings of product line versus clone and own has totaled in excess of $80 million over the past 3 years.
- Requirements defects follow the same story, and bring commensurate savings. Here the combined savings for all government agencies totaled $39 million over the past 3 years.
- For testing, additional integration testing across multiple programs (instead of one) added 40% in cost to the initial fix. So that cost is gone as well.

All of these measurements come from three years of actual cost data that have remained relatively constant from year to year.

In addition to defect-fix savings, the approach brings savings in new development as well. Developing an element upgrade for the entire family seems to add, at most, about 10% to the cost of development. This is much less than the cost of cloning and adapting the upgrade (and then testing it) in each of several other programs. This "design for releasability" as it's called, is now instilled in each developer's mindset as part of an overall culture change. People ask themselves "How am I going to design this so it can be cleanly eliminated for a foreign ship?" or "How can I 'vary out' that capability?" Early data collection is trending towards a 40-60% potential reduction in test cases required for new development.

If a new program (or component or subsystem) is simply a new combination of existing features then development cost goes to zero – only testing is needed. The feature-based product line approach in use requires only a new feature profile to be written to describe the new member.

Program overhead is also reduced through elimination of duplication – merged system engineering and test meetings, and program management meetings, for example.

Finally, Lockheed Martin has been able to use the product line approach and AEGIS success to win other proposals, a very important indicator of success to a defense contractor in a time of shrinking defense budgets.

9. Lessons Learned

AEGIS came to its current success by learning lessons, some of them hard, and adjusting course to adapt. Below, we roughly categorize the lessons as technical and organizational/social.

Technical lessons include the following.

- As emphasized in many product line case studies, the systems and software architectures have played a large role in the success of the product line. The architectures provide the structure and set the scope and granularity of many of the shared assets. The Navy's Open Architecture push served as an impetus to change the business model for AEGIS and helped instill a culture to avoid configuration-unique development.
- Establishing a product-line-wide thrice-yearly build rhythm has been invaluable as an aid to planning and execution. Engineers and Program Managers for both Lockheed Martin and its customers know far in advance when builds will be available and the content of those build enabling them to conduct planning meetings to adjust content for near-term builds and define new content for future builds.
- While every customer does not have to take every build, it is important for customers not to get too far behind by waiting too long between upgrades. Skipping too many builds tends to result in large numbers of changes that can lead to regression issues. Every customer is encouraged to "be an active part of the family," and stay engaged by periodically building configurations to mitigate regression risk.
- Alignment of artifacts, especially requirements and software, is extremely powerful. Under the product line approach of Figure 4, shared assets are configured all at once using the feature profile for the configuration being built, so feeding in a ship's feature profile results in a fully-aligned set of requirements, code, and (coming soon) tests. These artifacts are automatically consistent with each other because they are derived from the same feature profile, so there is never a concern about artifact mismatch. This consistency turns out to be extremely useful in the process of auditing the contents of a build to ensure that no proscribed content is included in any of the artifacts.

Organizational and social lessons include the following.

- Lockheed Martin first tried to instill the product line approach throughout the AEGIS program by senior management fiat. Despite sincere management intent, including a number of intense meetings in which the technical leaders were asked one by one to say how they were going to support the product line approach, the paradigm shift was never completely fulfilled. People doing the day-to-day work were allowed to drift back into configuration-centric activities and mindsets. It was only after re-organizations occurred that re-structured the customer-specific teams (replacing them with smaller, leaner product delivery teams) and moved the resources into product-line-wide shared asset groups did the transition finally find traction. In the language of [12], Lockheed Martin did a good job *launching* the product line but the *institutionalizing* was not fully successful until after reorganization. This manifested itself during a delivery cycle for one of the ships in which work was done under the new approach but under the old organizational structure. The delivery was eventually successful, but not without an alarming amount of re-work.
- From the Navy side, the three-tiered governance scheme (Technical, Programmatic, Strategic) has been successful, but challenging. Products in this product line are typically destined for ships or ballistic missile defense land-based facilities. All are enormous and expensive, highly visible, with a place on the national and international stage, and with a very strong constituency. Pressures for customer-specific development and scheduling are probably stronger in this product line than in any other. Thus, external governance is an ongoing exercise in hard-fought compromise.
- The difficulty of obtaining compromise is compounded by the fact that the participants in the governance structure are physically distributed, work for different organizations and therefore have different priorities and loyalties, and tend to advocate each for his or her own world view. Just to have a meeting with everyone in physical attendance is extraordinarily difficult, and so personal relationships of the kind that lead to compromise are often hard to come by. This stands in stark contrast to, for instance, a case study in [5] in which the business unit managers had strong personal commitments to one another, based on working for the same company and living in the same close-knit town.
- "Build it once and fix it once!" has been a powerful central theme of this product line, and has been an intuitive and helpful aid to bring people on board with the idea quickly.
- The measures of success of the product line approach led to centralized customer funding that reinforced the approach.

Perhaps the largest lesson to be taken from AEGIS is that the product line approach works in this high-stakes high-visibility environment. Successful interaction between the various customer agencies and the development organization has brought about careful (if challenging) and successful governance procedures to support the product line approach. Governance is hard, but all parties agree that the payoff of the product line approach is worth meeting the challenge.

10. Summary

This paper has presented a comprehensive look at a large and complex product line with national and international visibility. We have focused on the governance structures and policies needed to operate the product line successfully in an environment of multiple and diverse customers, each with their own specific functional and schedule needs that must be met while still achieving overall affordability. To achieve these goals, Lockheed Martin and the Navy have, in cooperation, put in place development-side and customer-side governance policies.

On the development side, the salient aspects of governance include

- an **organizational structure**, with roles and responsibilities of the various players, based on the product line approach and the system-of-systems (product line of product lines) architecture of the AEGIS Weapon System;
- careful attention to **metrics and measurements**, to detect trouble spots and ensure that the "factory" is working as expected;
- a **regular, predictable, three-times-a-year build schedule** for every member of the product line, with supporting activities (such as the requirements review cycle) scheduled to support the builds;
- a **two-tier internal governance structure**, comprising program planning and program execution, both with a product line perspective.

On the customer side, governance activities include:

- a **three-tiered approach** comprising technical, programmatic, and strategic perspectives with a well-defined flow of issues from one to the next;
- a small number of **key planning artifacts** (new development fielding plan, branch and merge plan, and build plan) that, together, define the vast majority of all of the work in the CSL and are produced and maintained by stakeholder consensus;
- a **cost-sharing policy** to pay for common defect fixes, which follows from (but also helps to promote) the "Build it once, fix it once!" theme of the product line.

11. What's in Store for AEGIS?

The product line approach is growing within CSL and also across the Navy where AEGIS is but one combat system.

Lockheed Martin is planning to add more shared assets to the product line "factory" paradigm shown in Figure 4. There is more to be done to add test artifacts to the mix, and design specifications (for example, UML specifications using the Rhapsody design tool) are on the short list of additions. Planning documents, Statements of Work, basis of estimates reports, and other program management artifacts are also being made common for sharing.

The Navy, meanwhile, has an active effort under way to expand the product line approach to the entire surface combat fleet (for example, adding aircraft carriers and other non-AEGIS surface combatants to the family). A common product line architecture is under development and roll-out, which the Navy wants to use to define a standard set of components. The architecture we very briefly described in Section 2 is a compliant instance of that architecture.

Both of these trajectories reinforce the confidence that the respective parties have in the product line approach and the substantial savings it brings about.

12. Acknowledgments

We are grateful to Angie Beach, Aaron Miller and Gary Maloche of the U.S. Missile Defense Agency, who provided thorough and helpful reviews.

13. References

[1] Bachmann, F., Clements, P., "Variability in Software Product Lines," Technical Report CMU/SEI-2005-TR-012, Software Engineering Institute, 2005.

[2] BigLever Software, "BigLever Software's Product Line Engineering Solution," http://www.biglever.com/solution/solution.html

[3] Clements, P., Gregg, S., Krueger, C., Lanman, J., Rivera, J., Scharadin, R., Shepherd, J., Winkler, A. "Second Generation Product Line Engineering Takes Hold in the DoD," *Crosstalk – The Journal of Defense Software Engineering,* vol. 27, no. 1, January/February 2014.

[4] Clements, P., Krueger, C., Shepherd, J., Winkler, A., "A PLE-Based Auditing Method for Protecting Restricted Content in Derived Products," *Proceedings SPLC 2013,* Tokyo, 2013.

[5] Clements, P., Northrop, L. *Software Product Lines:* Practices *and Patterns,* Addison Wesley Longman, September 2001.

[6] Guertin, N., Clements, P. "Comparing Acquisition Strategies: Open Architecture vs. Product Lines," *Proc.* Seventh *Annual Acquisition Research Symposium*, U.S. Naval Postgraduate School, Monterey, California, 2010.

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

[8] Naval Surface Warfare Center, "AEGIS Combat System," http://www.navsea.navy.mil/nswc/dahlgren/ET/AEGIS/default.aspx

[9] Office of the Deputy Under Secretary of Defense for Acquisition and Technology. *Systems and Software Engineering. Systems Engineering Guide for Systems of Systems, Version 1.0.* Washington, DC: ODUSD(A&T)SSE, 2008. http://www.acq.osd.mil/sse/docs/SE-Guide-for-SoS.pdf

[10] Program Executive Office for Integrated Warfare Systems, "Surface Navy Combat Systems Development Strategy: Acquisition Management Plan (AMP)," v5.4, 27 Oct 2008.

[11] Richardson, D., Murphy, A., Sheehan, T. "The Importance of Systems Integration: System-of-Systems Enabler," *Journal of the American Society of Naval* Engineers, July 2008.

[12] Software Engineering Institute, "Framework for Product Line Practice (Version 5.0)" http://www.sei.cmu.edu/ productlines/frame_report/index.html