Experiences With Software Product Line Development

Dr. Paul Jensen
Overwatch Systems

Overwatch Systems recently transitioned to a software product line (SPL) approach. Using its SPL, Overwatch Systems provides both software products and custom software system development in the domain of intelligence planning, collection, and analysis to the U.S. DoD and intelligence community. This article describes the approach taken in Overwatch Systems' transition, describes the product line architecture that is a key to the Overwatch Intelligence Center (OIC) SPL, and provides the lessons learned during the transition.

Overwatch Systems focuses on the development and fielding of multi-discipline data analysis software systems. Areas of expertise include data fusion, all-source analysis, signal intelligence acquisition and analysis, sensor network technology, and visualization. Starting in 2003 and continuing through 2007, the company transitioned to an SPL approach, producing the OIC product line, of which all new applications and systems are members. Multiple systems have been fielded as members, including a Signal Intelligence (SIGINT) collection and analysis system and an all-source analysis system.

During the Overwatch Systems transition, the company desired more information related to the application of SPL practices in business environments similar to our own: fulfilling government contracts by delivering members of an SPL. The goal of this article is to make such information available so that introducing this approach can be easier in the future.

Software Product Lines

As defined by the Software Engineering Institute (SEI), an SPL is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular mission and that are developed from a common set of core assets in a prescribed way [1]. In other words, an SPL consists of a family of software systems that have some common functionality and some variable functionality. To take advantage of the common functionality, reusable assets (referred to as core assets) are developed, which can be reused by different members of the family. Following an SPL approach involves not only developing core assets but also focusing on the systematic production and delivery of the different member systems (or variants) of the SPL family.

OIC Background

In 2003, Overwatch Systems was in a transition period. In previous years, a large portion of the business was dedicated to developing a large all-source analysis system for a single customer. This system consisted of approximately 10 million lines of code (LOC) and hundreds of components. To produce solutions for new customers, this large analysis system was cloned, modified, and extended to meet new requirements. Often, this approach led to difficulties caused by the complexity and dependencies encountered while adapting a legacy system to a new mission.

"Following an SPL approach involves not only developing core assets but also focusing on the systematic production and delivery of the different member systems of the SPL family."

Simultaneously, Overwatch Systems was beginning a large multi-year effort to produce situation understanding and data fusion capabilities for a second large customer. Although rigid constraints were imposed on the architecture and development of this software, the company made a commitment to create shared value for both of the current customers through the reuse of key components.

Transition to the SPL Approach

Preliminary Steps

Based on a clear business case, the decision to transition to an SPL approach was made by the CEO of the company. From the beginning, the general manager, chief architect, and vice president of engineering were the champions of the product line vision. Having strong support was critical to the successful SPL creation and operation.

Transitioning affects all parts of a business and introduces many risks. Clements and Northrop [1] identify 29 business practice areas with activities that are essential for product line success (e.g., architecture definition, process definition, funding, etc.). These practice areas cover all aspects of the business including software engineering, technical management, and organizational management. During this preliminary transition, a particular focus was put on the practice areas of mining existing assets and training. Although these two practice areas were critical to the transition, a lack of focus on other practice areas, such as architecture definition, funding, and structuring the organization, caused many problems that are discussed later in this article.

To reduce the risks, an incremental transition plan was created. This approach introduced product line concepts to various practice areas incrementally and relied on the creation of core assets to be realized as part of ongoing development activities. A specific customer system was targeted to be the first member of the SPL. This target provided motivation by creating a sense of urgency and forcing employees to focus on the product line transition instead of focusing only on normal daily issues.

The transition plan was derived from the SPL Factory Pattern, which is a composite pattern for an entire product line organization [1]. Activities that took place in this preliminary stage included scope definition, market analysis, a technical probe, training on product line concepts, and a determination of funding and organizational models. The technical probe was used to assess the current state of the organization in the context of a

product line transition and to track the progress of the transition over time. The training consisted of seminars for employees and required reading materials, such as [1]. The seminars covered product line concepts and how practice areas would be affected by the transition. The training was provided by the SEI's Paul Clements, the vice president of engineering, and the chief architect. This mixture of training provided both an expert view on SPLs and a description of product line practice areas specific to Overwatch Systems.

Based on our experience, the development of the funding model and organizational model are particularly important for success with an SPL approach. The organizational model that Overwatch Systems initially implemented was based on a division of domain engineering activity where reusable assets are created and customer-specific engineering activity in which the assets are assembled, customized, and extended. This model, as described in [2], appeared to be optimal. A domain engineering group was formed to produce reusable assets. Multiple customer-specific engineering groups would use those assets for customer system development. The funding for the domain engineering group was initially derived from internal Overwatch Systems funds. The plan was that these funds, occasionally augmented with funds from customer projects, would fully support the employees required for domain engineering.

Two major problems were encountered in operations that caused the company to eventually change this funding and organizational model. The first problem was that difficulties were encountered in funding a domain engineering group at a constant rate. Internal funds were insufficient to pay for a majority of the core asset development that was required. Fluctuations in customer project funding caused unmanageable fluctuations in the size of the domain engineering group. The second problem was that experts in certain technical areas were continuously pulled from the domain engineering group to the customer-specific engineering group to fill critical gaps.

In the second year of the transition, Overwatch Systems changed its organizational and funding model in order to address these problems. The company transitioned to an organizational model described as *mixed responsibility* [2], in which customer-specific engineering groups shared responsibility for developing reusable core assets. The lesson learned from this change is to ensure, through analysis and accurate estimation, that the funding and organizational model chosen for an SPL approach aligns with the demands and characteristics of one's business.

Starting Out

The initial domain engineering group consisted of eight engineers and would grow to more than 25 before the organization transition. The domain engineering group's mission was to create the necessary number of core assets in a new architecture that would allow the company to begin making systems that are members of the SPL.

> *"The initial SPL architecture was developed and focused primarily on the dependencies between components in the product line, emphasizing specific rules that governed what types of dependencies were allowed between product line assets."*

The initial SPL architecture was developed and focused primarily on the dependencies between components in the product line, emphasizing specific rules that governed what types of dependencies were allowed between product line assets. The architecture neglected to address issues related to the product engineering or assembly aspects of the products and did not define much of the infrastructure that, as the company discovered later, is needed to operate a product line efficiently.

Drawing from a legacy software baseline consisting of approximately 10 million LOC, the core asset mining process began. Several issues were encountered and had to be overcome. The company lacked experience in performing domain analysis, which is the process of determining the commonality and variability across all current customers and potential customers. Consequently, domain analysis was not performed adequately and the deciding factor in determining what assets to mine and what variations to implement became the requirements of the first customer system to use the product line. Domain analysis processes had to be determined, tested, and applied to overcome this issue.

The architecture and infrastructure developed were not sufficiently designed to support an SPL. Insufficient thought was put into determining how the core assets would be assembled and how the architecture could exploit commonality. As the core asset base expanded, many cases were observed where commonality was not being taken advantage of and software components were not well integrated. To address this problem, a revised architecture was created later.

The company did not possess expertise with variations in non-software core assets (e.g., related artifacts such as requirements and test procedures). The mining process started before the processes and tool modifications needed to support non-software core assets were executed. The result was an inadequate representation of variation and commonality, and a lack of variation dependency in these artifacts.

Despite these difficulties, approximately 4.5 million LOC were mined and new software components were developed, resulting in approximately 200 software core assets. These core assets provided a range of capabilities in the all-source and SIGINT intelligence domains, including data ingestion, management, processing, fusion, as well as geospatial, temporal, and relational visualization. In 2005, the first customer system was made using the OIC's SPL.

Course Correction

During the *Starting Out* phase of the transition, it became clear that although products could be produced from the product line, the architecture and infrastructure would not be sufficient in the long term. After a technical analysis, a decision was made to implement a new product line architecture.

The new architecture, called Viper, focused on creating composite applications from product line assets. Composite applications are formed by combining functionality drawn from several different sources within a service-oriented architecture and packaging them together into a single-user interface or

work process. In the context of an SPL, a composite application is a member of the product line composed or assembled from the reusable software assets.

Viper implements several aspects of service-oriented architecture, including a high degree of core asset decoupling. The architecture introduced the definition and control of interfaces and asset and data discovery. A common messaging mechanism, or service bus, was introduced to allow software core assets to publish and subscribe to objects, events, and interfaces. A common object model was added to impose object commonality on all software core assets regardless of their origin. A single-user interface was introduced to allow for the assembly of a system that can be presented to the user as a single application instead of numerous applications with different user interfaces. The unified-user interface also enabled an increase in the commonality for functionality such as with editing data (i.e., a single data editor was used instead of multiple data editors). Lastly, the architecture improved product engineering support by providing SPL-attached processes via a Software Development Kit (SDK). The SDK enabled those performing the construction of product line systems to use automation tools for certain aspects of the system assembly process.

The Overwatch Intelligence Center

The first version of the Viper architecture and SDK were completed in 2007 and serves as the core of the OIC SPL. All development relevant to the product line—including legacy code mining, new development, and third-party code integration—uses the Viper architecture. Legacy components are being migrated from the original product line architecture to the Viper architecture.

In its current state, the OIC contains core assets related to intelligence planning, collection, analysis, visualization, and data management. The specific analysis areas addressed by the product line include SIGINT, Human Intelligence, and data fusion (e.g., data correlation, aggregation, and threat estimation). These core assets are designed to be deployed either in the Viper product line architecture or integrated directly into a customer's enterprise architecture.

The product line consists of approximately 270 software core assets with a corresponding number of associated core asset artifacts. The 270 core assets are composed of approximately 900 components and 4.7 million LOC. The origin of the core assets include a legacy code base, newly developed software, and acquired third-party components.

Achievements and Improvements

In spite of missteps at the inception of the effort, there have been several achievements and improvements realized. Two licensed products have been created as members of the product line: an all-source analysis and a SIGINT product. These products are pre-configured collections of core assets that are customized to individual customer specifications.

The all-source analysis product provides data management, link analysis, text extraction, and geospatial capabilities. The first version of this product, based on the original SPL architecture, was developed in less than 90 days. The company estimated that this improvement got products to-market approximately 2.5 times quicker.

The SIGINT product provides collection, analysis, and processing capabilities. The product has been delivered, with modifications and extensions, to two customers thus far. Software reuse between the two customers is approximately 70 percent. There has also been interest from a government customer to acquire these capabilities as an SPL instead of as individual products.

More than 10 customer systems have been completed that are members of the OIC SPL. Software reuse (within these deliveries) is estimated to range between 40 and 70 percent. The software that implements the Viper architecture and SDK have been sold to a U.S. government customer and is serving as the basis for integrating Overwatch Systems, government-developed, and other contractor software. In 2007, the OIC was nominated for inclusion into the SEI's SPL Hall of Fame, which exists to acknowledge excellence in the field and influence in the software engineering community. Metrics to measure other improvements, such as product quality, system cost differential, integration speed, and customer satisfaction are not yet available, but anecdotal improvements have been noted in these areas as well.

Economics

The SEI provides an economic model, called the Structure Intuitive Model for Product Line Economics, that can be used to determine if a product line approach is economically positive for a particular organization [3]. Typically, the benefits outweigh the costs (that is, a positive return on investment) of a product line approach with the third system built from an SPL. Using this as the economic model, Overwatch Systems' experience is in line with the typical return on investment. From 2003 to 2007, Overwatch Systems' revenue nearly tripled. It is believed by management that this growth could not have been achieved without the speed and reduced costs enabled by an SPL approach.

Lessons Learned

During the transition, a number of lessons were learned:

- **Support of organization leadership is critical.** At many points during the transition, if not for the support of senior management—both in terms of funding and organization direction—the effort would not have succeeded. In the case of Overwatch Systems, the general manager, chief architect, and vice president of engineering are product line champions.
- **An architecture specifically designed to support a product line is essential.** Overwatch Systems' first attempt at a product line architecture was rooted in our legacy applications. It failed to properly address certain quality attributes such as modifiability, configurability, and extensibility in the architecture. These attributes were later addressed with the Viper architecture. More infrastructure design and development should have been done before the mining of core assets began. A proper architecture evaluation, as described by the SEI's Architectural Trade-off Analysis Method, is strongly recommended to uncover product line architecture deficiencies [4].
- **It is important to address product line requirements in support tools and processes early in the effort**. Overwatch Systems attempted to change these tools and processes (e.g., requirements management) in parallel with creating core assets. The result was tools that didn't meet requirements, frustration for product line users, and software artifacts that didn't optimally address commonality and variation. Special focus should be made on the requirements for these tools, which are related to creating and maintaining relationships between variation points manipulated in each tool. One way to accomplish this

Coming Events

March 2-5
25th Annual Test and Evaluation National Conference
Atlantic City, NJ
www.ndia.org/meetings/9910

March 2-6
8th International Conference on Aspect-Oriented Software Development
Charlottesville, VA
www.aosd.net/2009/

March 4-5
TechNet Tampa 2009

Tampa, FL
www.afcea.org/events/tampa/09/Introduction.asp

March 22-27
2009 Spring Simulation Multi-Conference

San Diego, CA
www.scs.org/confernc/springsim/springsim09/cfp/springsim09.htm

March 23-26
SEPG 2009 North America
San Jose, CA
www.sei.cmu.edu/sepgna/2009/

April 20-23
21st Annual Systems and Software Technology Conference

Salt Lake City, UT
www.sstc-online.org

Coming Events: **Please submit coming events that are of interest to our readers at least 90 days before registration. E-mail announcements to: nicole.kentta@hill.af.mil.**

is to determine how the selection of a variation point in requirements is communicated to the related variation point in test procedures.

- **Early in the transition, determine the process for domain analysis that best fits your organization**. Overwatch Systems arrived at a fast, lightweight process, centered on the creation of a high-level architecture artifact that can be performed in the small amount of time that is typically available in our projects. To determine our process, the company examined and used concepts from several references including [5, 6, and 7].
- **Put processes in place to perform domain analysis activities as early in the project life cycle as possible**. Overwatch Systems creates a domain analysis document as early as the proposal stage of a project and it evolves during the project life cycle. This approach ensures that all project participants are in agreement with respect to the project's relationship to the product line.

Conclusion

A product line approach to developing software for government customers is viable and holds tremendous potential for shortening time-to-market and delivering a better value proposition. Adopting a product line approach impacts engineering, technical management, and organizational management aspects of a business and can be adopted by a company like Overwatch Systems in the span of a few years. When executing a transition to a product line approach, special attention should be paid to the product line architecture, the tools and processes that must be modified to support the product line, and techniques related to domain analysis.

In spite of missteps during the transition to a product line approach, Overwatch Systems has successfully produced the OIC SPL. The company has created and delivered multiple software systems from its product line to multiple defense organizations, allowing the government to receive the benefits that a product line approach promises, including decreased development time and reduced costs from planned reuse.◆

References

1. Clements, Paul, and Linda Northrop. Software Product Lines: Practices and Patterns. SEI Series in Software Engineering. Addison-Wesley Professional, 2001.
2. Bosch, Jan. Software Product Lines: Organizational Alternatives. Proc. of the 23rd International Conference on Software Engineering. Toronto, 2001: 91-100.
3. Clements, Paul C., John D. McGregor, and Sholom G. Cohen. "The Structured Intuitive Model for Product Line Economics (SIMPLE)." SEI, Carnegie Mellon University. Technical Report CMU/SEI-2005-TR-003. Feb. 2005 <www.sei.cmu.edu/publications/documents/05.reports/05tr003/05tr003.html>.
4. Clements, Paul, Rick Kazman, and Mark Klein. Evaluating Software Architecture: Methods and Case Studies. Addison-Wesley Professional, 2002.
5. Kang, Kyo C., et al. "Feature-Oriented Domain Analysis Feasibility Study." SEI, Carnegie Mellon University, 1990. Technical Report CMU/SEI-90-TR-021.
6. Cohen, Sholom G., et al. "Application of Feature-Oriented Domain Analysis to the Army Movement Control Domain." SEI, Carnegie Mellon University. Technical Report CMU/SEI-91-TR-28, ESD-91-TR-028. June 1992 <ftp://ftp.sei.cmu.edu/pub/documents/91.reports/pdf/tr28.91.pdf>.
7. Gomaa, Hassan. Designing Software Product Lines With UML. Addison-Wesley Professional, 2004.

About the Author

Paul Jensen, Ph.D., is the chief architect for Overwatch Systems, Tactical Operations. Overwatch Systems makes intelligence planning, collection, analysis, and visualization software for the DoD and other government agencies. He has 14 years of experience in designing and building complex software systems. At Overwatch Systems, Jensen has served as an architect for numerous complex software-intensive projects, led the adoption of an SPL approach to development, and led the adoption of product innovation processes. He has a doctorate in physics from the University of Texas at Austin.

Overwatch Systems
P.O. Box 91269
Austin, TX 78709-1269
Phone: (512) 358-2600
E-mail: paul.jensen@overwatch.textron.com