An Integrated Tool Environment for DoD Product Line Engineering

Christopher P. Fuhrman
Institute for Software Research
1000 Technology Drive
Fairmont WV 26554 USA

Nancy Solderitsch
ProLogic, Inc.
1000 Technology Drive
Fairmont WV 26554 USA

Sherif Yacoub and Hany Ammar
Department of Computer Science
and Electrical Engineering
West Virginia University,
Morgantown WV 26505 USA

ABSTRACT

We propose an environment that supports the development of software products using a product line engineering approach in the domain of embedded weapon fire control systems. The environment is a set of integrated tools that facilitate the various processes of product line engineering. Some of the tools are commercial, off-the-shelf (COTS). Some tools are proposed as aids to integration of the environment; other experimental tools that are aimed at automated conformance checking using design patterns are discussed.

Keywords: product line engineering, software reuse, COTS software, CASE tools, embedded systems, UML, design patterns, software architecture frameworks.

1. INTRODUCTION

In the domain of weapon/fire control systems, embedded software will be a key cost driver in the next generation. The omnipresence of embedded systems in today's market and the quality of requirements that these systems are expected to meet, put a great deal of pressure to bear on the techniques used to develop these systems. It has been recognized that development of standardized product line (or reference) architectures with concomitant infrastructure technology, tools, and design methodology is an enabler to control software cost and complexity.

In its infancy, software engineering was concerned with the development of single software systems. To develop each system, an organization must invest in analyzing requirements, designing of software architectures, documenting, planning of schedules, testing, etc. As organizations began to develop multiple software systems, it became evident that, by minimizing the redundant work done in software development of multiple products, significant gains could be made. This is the basis behind the idea of software product line engineering (PLE).

Through a Small Business Innovative Research (SBIR) grant with the U.S. Army, we have begun a case study of an effort involving PLE. This PLE work involves the development of a limited set of software products that are part of the Crew Station Decision Aids of the Crusader—the US Army's next-generation cannon artillery system. The limited set of software products is associated with terrain mapping. The goal of this study is the development of standardized infrastructure technology, tools and design methodology, an Integrated Tool Environment (ITE), for the management and control of cost and complexity of software used in embedded combat vehicle and indirect fire weapon applications.

We also consider constraints imposed on the technical architecture of the software products. Even in some general cases, software products must conform to guidelines imposed by the market for which they are developed. In the specific case of software developed for the US military, guidelines might include the Defense Information Infrastructure (DII) Common Operating Environment (COE) [1] and the Joint Technical Architecture (JTA) [2].

We propose an Integrated Tool Environment (ITE) that facilitates the processes of product line engineering, but specifically in a DoD context. We consider some COTS tools for some of the processes as well as methods for integrating them into the ITE. Finally we discuss using design patterns as a means to extend the ITE to both support DoD guidelines and to overcome some integration challenges.

2. PROCESSES IN PRODUCT LINE ENGINEERING

The product line software development environment [3] in Figure 1 consists of several levels of engineering. At the Enterprise Engineering level, management looks at all engineering within the organization and the products created.

At the Product Line Engineering level, technical staff considers the product lines to understand and codify the commonalities and variances within and across product lines. Domain engineering products representing these commonalities and variances become assets available for reuse in the evolution of the product line. A technical architecture would be one of the domain-engineering products.

The Asset Management level provides management for the product line assets. At the application level, the organization is creating products using assets created through enterprise and domain engineering, and those products also become assets available for reuse in the product line. Asset management is a necessary part of these life cycles, in that it manages the enterprise and domain-specific assets for use in the application engineering that takes place.

The focus in our case study is on the Product Line Engineering and Asset Management areas. The Product Line Engineering level lists a number of activities to be performed: Product Line Scoping and Identification; Domain Analysis; Product Line Architecture Development; Infrastructure Development. In our case study the Product Line Scoping and Identification as well as the Domain Analysis steps have already been performed. While it would be a good idea to document these steps, it is not crucial if the group is small and the product line well understood by participants. The focus of this project is in Product Line Architecture and Infrastructure Development.

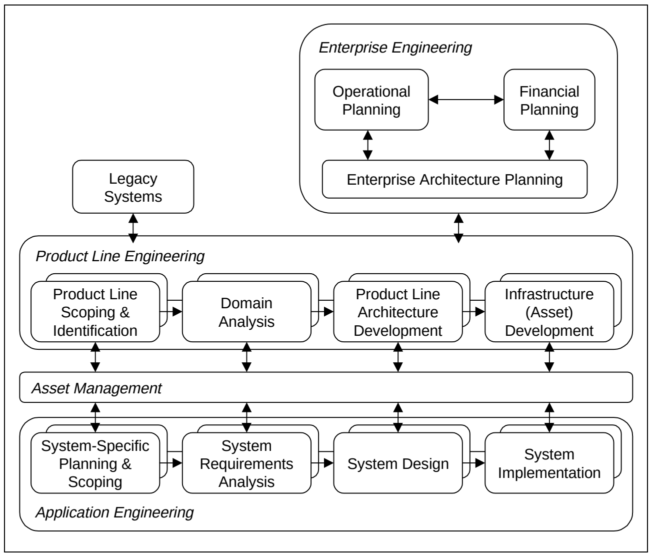
Enterprise Engineering
Operational Planning
Financial Planning
Enterprise Architecture Planning
Legacy Systems
Product Line Engineering
Product Line Scoping & Identification
Domain Analysis
Product Line Architecture Development
Infrastructure (Asset) Development
Asset Management
System-Specific Planning & Scoping
System Requirements Analysis
System Design
System Implementation
Application Engineering

Figure 1—Overview of product line engineering

The Software Engineering Institute defines Product Line Architecture as a "description of the structural properties for building a group of related systems (i.e., product line), typically the components and their interrelationships. The guidelines about the use of components must capture the means for handling variability discovered in the domain analysis or known to experts. (Also called a reference architecture)" [4].

Infrastructure Development refers to the process that develops the components, software, hardware, documentation, etc. that become the asset base for the product line.

3. INTEGRATED TOOL ENVIRONMENT

Figure 2 shows how the individual parts of the Integrated Tool Environment will support Product Line Engineering processes. The Product Line Scoping & Identification and Domain Analysis processes have been completed for our case study, and the basic tool support for these processes is a repository to store information from Legacy Systems and the processes. The Scoping & Identification and Domain Analysis processes are well documented by the Army Reuse Center. The group involved with the case study is actively developing a scalable, evolvable product line architecture, and the infrastructure—the actual assets that can be used to build new products. The ITE will provide active tools to support these activities.

Infrastructure (Asset) Development

There are several individual contractors responsible for developing the various components of the Crew Station Decision Aids system. One of the goals of this study is to develop a standardized infrastructure technology. To achieve this goal, some decisions had to be made initially about the design environment and preliminary technical architecture. It has also been determined that the program will use an object-oriented approach to software development. While product line engineering does not require the use of object-oriented techniques, the decision to use the OO paradigm across the development process helps simplify the development of methodology and tools.

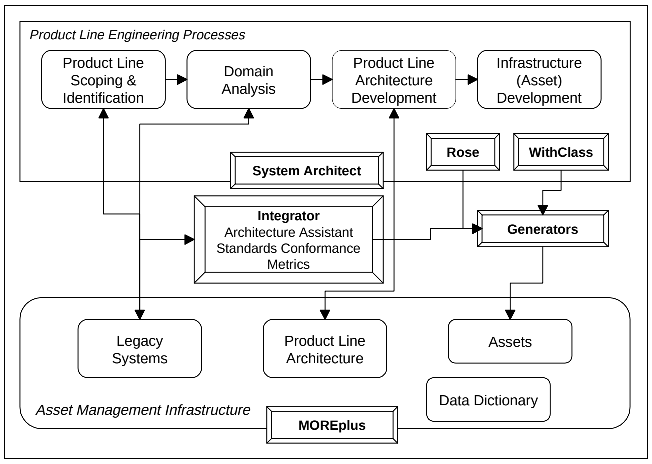
Product Line Engineering Processes
Product Line Scoping & Identification
Domain Analysis
Product Line Architecture Development
Infrastructure (Asset) Development
Rose
WithClass
System Architect
Integrator
Architecture Assistant
Standards Conformance
Metrics
Generators
Legacy Systems
Product Line Architecture
Assets
Data Dictionary
Asset Management Infrastructure
MOREplus

Figure 2—Proposed Integrated Tool Environment

The group responsible for the guidance of the software development in the case study has determined that the main design tool will be *Rose* (Rational Software, Inc.), the leading state-of-the-shelf, object-oriented analysis, modeling, design, and construction tool. Rational had a major role in the development of the Unified Modeling Language (UML), which is fast becoming the standard notation for object-oriented software architectures.

Commercially available design and development tools are optimized for the tasks they perform. There are strategic alliances between some vendors that provide some integration, but not for all. In particular, the individual tools have repositories that are designed for their information, and to support their design processes. These repositories do not handle other tools' information well, and do not present an integrated picture of the information to the user. Attempts to provide common data repository infrastructure, e.g., Portable Common Tool Environment (PCTE) [5], Common Ada Programming Support Environment Interface Set (CAIS) [6], have failed to attract commercial interest.

UML

The Unified Modeling Language (UML) has become the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML simplifies the complex process of software design, making a *blueprint* for construction.

On the other hand, we have two points of concern with respect to UML. First, although UML is a standard and is supported by many of the tools that could become part of the ITE, it is important to realize that many of these tools support variant UMLs and there is no UML interchange standard. The interoperability of multiple tools claiming to support UML can only be verified when the integration is performed. Second, modeling in UML does not guarantee that combinations of components will work. The use of design patterns and frameworks for product line architecture definition is how we intend to assure compatibility of collections of components. This is described in the subsection below on Integrators.

Asset Management

The core of the ITE will be a user-oriented repository of product line information. The commercial product *MOREplus* (MountainNet, Inc.) will be used to catalog and group information on different aspects of the Crew Station Decision Aids product line. *MOREplus* is a web-based front end to an *Oracle* database that has been optimized to support product line repository activities. *MOREplus* is used by a number of commercial and Government agencies, including the Defense Information System Agency (DISA) and the Army Reuse Center (ARC). *MOREplus* will be modified and 'add-ins' integrated to support the automatic generation of meta-data on components for the repository from the UML tools used in the system design and development (e.g., Rational *Rose*, MicroGold Software's *WithClass*).

Examples of information to be stored in the ITE repository include

- data dictionary information (from the Crew Station Decision Aids product line);
- product line architecture information;
- assets, e.g., designs, application program interfaces (APIs), documentation, links between assets, legacy systems; and
- pointers to external information, e.g., JTA-A, DII COE home pages, etc.

This information will be entered into and extracted from the repository in the course of enacting the development process. This process, focusing on Product Line Architecture and Infrastructure development, will coordinate with and complement the processes maintained by the Army Reuse Center. The ITE will provide automated support, integrated with the development tools and MOREplus, to enter, link and extract the information.

Integrators (Architecture Framework and Patterns)

At a later phase of our work, the focus will be on integrating architecture framework and patterns concepts and tools into the ITE. One aspect of this will be to provide automated support for checking compliance with the Joint Technical Architecture and the Defense Information Infrastructure (DII) Common Operating Environment (COE). Similarly, as described above, this method can be used to achieve better tool integration using the UML.

Using the results obtained in work concerning pattern-oriented frameworks [7], we intend to augment the ITE with experimental tools. These tools would facilitate the generation of applications using design patterns such that the applications can conform to particular technical architectures. The results of this technique are promising in limited cases [8], and we shall investigate how it applies in a broader sense.

Commercial tools supporting framework and patterns are evolving rapidly. We will eventually determine which ones should be integrated into the ITE. Finally, we will consider appropriate product line engineering metrics and incorporate support for collecting these metrics into the ITE.

Generators

As discussed above in the section on Asset Management, additional tools are proposed that can automatically generate the meta-data about UML design artifacts for the ITE repository. The artifacts remain in their tool repositories; only the meta-data resides in the ITE Repository.

Both *Rose* and *WithClass* have *Virtual Basic for Applications* (VBA) integrated into them. At this time, it appears that using VBA to generate the meta-data will provide a tool framework that can be used in both commercial tools, and possibly in others tools.

Additional tools are proposed for development that support architectural connections (e.g., generation of APIs that conform to Army Weapons Systems Technical Architecture Working Group (WSTAWG) standards), and other development artifacts that correspond to architectural constraints and rationale.

4. CONCLUSIONS

In this work we present an Integrated Tool Environment (ITE) that is aimed at standardizing and facilitating the processes of software product line engineering in the domain of embedded weapon fire control systems. We have discussed how several COTS tools can be used in the environment, and we have proposed tools used for (1) the generation of meta-data and APIs conforming to DoD standards and (2) the automated integration of the software into technical architectural frameworks using design patterns.

REFERENCES

[1] Defense Information Systems Agency, "Defense Information Infrastructure Common Operating Environment, Version 3.1, Baseline Specifications," Joint Interoperability and Engineering Organization, Defense Information Systems Agency, 29 April 1997.

[2] Department of Defense, "Joint Technical Architecture, Version 2.0," U.S. Defense Information Systems Agency (DISA), 26 May 1998.

[3] Army Reuse Center, "An approach for implementing an object-oriented repository for Army Technical Architecture Models," U.S. Army, August 1997.

[4] P. Clements and L. M. Northrop, "A Framework for Software Product Line Practice-Version 1.0," Software Engineering Institute, September 1998.

[5] ISO/IEC 13719-1:1998, "Information technology -- Portable Common Tool Environment (PCTE) -- Part 1: Abstract specification," 1998.

[6] DOD-STD-1838, "Common APSE Interface Set (CAIS)," October 1986.

[7] S. Yacoub and H. Ammar, "Towards Pattern Oriented Frameworks," to appear in *Journal of Object Oriented Programming JOOP*, 1999.

[8] S. M. Yacoub and H. H. Ammar, "The development of a client/server architecture for standardized medical application network services," presented at *IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET'99)*, Dallas, Texas, USA, 1999.