Using a Product Line Approach to Develop Course Projects*

Gerald C. Gannod†‡ and John J. Doherty
Dept. of Computer Science & Engineering, Arizona State University
Box 875406, Tempe, AZ 85287-5406
E-mail: {gannod,doherty.j}@asu.edu

Abstract

Product Line and Product Family approaches are development techniques that take advantage of commonalities that exist among a set of current or planned products. The use of a product line approach allows for speedier integration of new requirements and capabilities to account for product variation without the timely process of reorganizing and reengineering an entire product. While the product line approach is an emerging paradigm in the software development research and industrial communities, little attention has been placed on its use as a methodology for developing and maintaining course projects in an engineering curriculum. As part of an Embedded Systems concentration at Arizona State University we are creating a course in Embedded Systems Engineering that focuses on systems integration and applications development. For this course we are developing a home automation product line. By using a product line approach, students can gain exposure to new technologies in successive offerings while still receiving instruction on core concepts. Furthermore, instructors can easily provide a wide variety of experiences for students with a minimal amount of incremental course refinements.

1 Introduction

One of the many challenges facing engineering educators today is the need to keep pace with technological advancements. Many factors must be considered before integrating new technology into a curriculum. Traditionally, if the technology varies widely from technology in current course offerings, a new course is considered. Otherwise, an existing course is modified or updated. In either case, the process of integrating the new technology into a curriculum can be both expensive and time consuming.

Product Line and Product Family approaches are development techniques that take advantage of commonalities that exist among a set of current or planned products. These shared features, or assets, may include artifacts such as software and hardware reference architectures, software and hardware components, processes used to develop and integrate components, and test plans. The use of a product line approach allows for speedier integration of new requirements and capabilities to account for product variation without the timely process of reorganizing and reengineering an entire product. While the product line approach is an emerging paradigm in the software development research and industrial communities, little attention has been placed on its use as a methodology for developing and maintaining course projects in an engineering curriculum. In treating a course as a targeted product market domain we have been able to apply the same techniques used in software product line development to course development.

At Arizona State University we are currently developing a concentration track in embedded systems[1]. As part of the curriculum we are creating a course in Embedded Systems Engineering

*This research supported in part by NSF Experimental and Integrative Activities Grant EIA-0122600.
†This author supported in part by NSF CAREER Grant CCR-0133956.
‡Contact Author.

(ESE) that focuses on systems integration and applications development. For this course we are developing a home automation product line that includes products such as security systems, air quality-monitoring systems, and lawn and garden maintenance systems. As a result, by using a product line approach, students can gain exposure to new technologies in successive offerings while still receiving instruction on core concepts. Furthermore, instructors can easily provide a wide variety of experiences for students with a minimal amount of course refinements.

The remainder of this paper is organized as follows. Section 2 describes background material in the areas of software product lines, embedded systems, and a methodology for course development. Section 3 presents the activities that occurred during course development including the use of a product line to develop an infrastructure for course projects while Section 4 describes related work. Finally, concluding remarks as well as future investigations are described in Section 5.

2 Background

This section discusses background material in the areas of software product lines and course development, and describes an embedded systems concentration track which serves as the context for the course described in this paper.

2.1 Software Product Lines

Clements and Northrop define a software product family, or *product line* to be "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way[2]." A few of the critical elements to the process of developing process lines include the recognition and development of core assets and the development of reusable processes that define how assets are used to assemble products. The benefits of product lines are numerous and include[2] improved time-to-market, improved product quality, increased customer satisfaction, increased ability to meet reuse goals, and decreased staffing requirements.

A software product line achieves these goals with savings across the entire development spectrum[2]. Requirements no longer require detailed analysis. Only the deltas to the common requirements will require analysis, thus reducing time and costs. The activity of creating an architecture is minimized and requires only small levels of fine tuning from the base architecture for each product. Additionally, components are reused as needed from the asset base, with new development only required to meet variabilities between products. Modeling, analysis and testing can rely, with a high degree of confidence, on previous members of the product line and will only need to focus on deltas. Planning and processes for a new member of the product family will be minimal and should already be in place. Finally, the organization will require fewer people to build products and those people will have stronger domain knowledge across the product line.

The three essential activities for a successful product line project are core asset development, product development, and management[2]. These essential activities translate into practice areas, as shown in Table 1.

2.2 Course Development Guidelines

Davidson and Ambrose describe an approach for planning and developing courses that is similar to the process used to develop proposals[3]. Figure 1 lists guidelines that focus on identifying the constituents of a course (e.g., the students), their backgrounds and experience, and course ob-

Software Engineering	Technical Management	Organizational Management
Architecture Definition	Configuration Management	Building a Business Case
Architecture Evaluation	Data Collection, Metrics, and Tracking	Customer Interface Management
Component Development	Make/Buy/Mine/Commission Analysis	Developing an Acquisition Strategy
COTS Utilization	Process Definition	Funding
Mining Existing Assets	Scoping	Launching and Institutionalizing
Requirements Engineering	Technical Planning	Market Analysis
Software System Integration	Technical Risk Management	Operations
Testing	Tool Support	Organizational Planning
Understanding Relevant Domains		Organizational Risk Management
		Structuring the Organization
		Technology Forecasting
		Training

Table 1: Software Product Line Practice Areas [2]

jectives based on course and program goals. The objectives lead to identification of a scope for the course as well as the learning activities that meet the objectives, outcomes, and scope constraints. One of the important aspects of these guidelines is that the preliminary activities (Steps 1–3) are consistent with typical practices required for accreditation.

In this paper, we describe the development of a course based on the use of these guidelines. Specifically, this paper discusses our efforts in following Step 3 and the use of a product line approach for the design of course projects. In addition, we examine the impact of the approach upon the overall design of the course.

Steps in Planning a Course (The New Professor's Handbook):

1. For each course, determine the backgrounds and interests of the students likely to enroll.
2. Choose the objectives of the course based on these backgrounds and on the knowledge and skills which you deem appropriate to teach, as well as on your interest and expertise.
3. Choose the scope and content of the course based on time and money constraints.
4. Develop the learning experiences to achieve the objectives, within the scope previously determined. These experiences may include in-class activities such as lectures, recitations, and group meetings, as well as out-of-class activities such as required readings and homework assignments.
5. Plan feedback and evaluation of student learning through tests, written reports, and other assessment techniques.
6. Prepare a syllabus based on the considerations above.

Figure 1: Course Development Guidelines [3]

2.3 A Concentration Track in Embedded Systems

At Arizona State University, we are completing the implementation of a novel infrastructure for a concentration track in embedded systems that combines important aspects of academic content

with the latest in research and industrial practices[1]. The concentration track emphasizes fundamental issues such as the balance between hardware and software and the respective trade-offs of building embedded systems.

Our curricular project spans the entire spectrum of activities related to the design and delivery of educational and research efforts and is characterized by three main innovative components: 1) a new industry-university collaborative model for integrating basic and applied research into a degree program[4], 2) creation and delivery of state-of-the-art course content and appropriate laboratories, and 3) creation of capstone projects that are implemented through internships.

The curricular project involves the synthesis of the core of an embedded systems program based on the latest research and close cooperation with industry. The content of the program draws heavily upon advanced research and development in industry and academia and is reinforced by specially structured internship activities that have been developed as part of this effort. The core material, which is not currently found in traditional computer engineering programs, provides the content that industry consultants have specifically identified as critical for engineers to function productively in the area of embedded systems. Figure 2 shows the structure of the embedded systems concentration track. The track is built upon the standard computer engineering core, and includes four courses: Embedded Systems Development, Embedded Systems Engineering, and a pair of capstone courses.

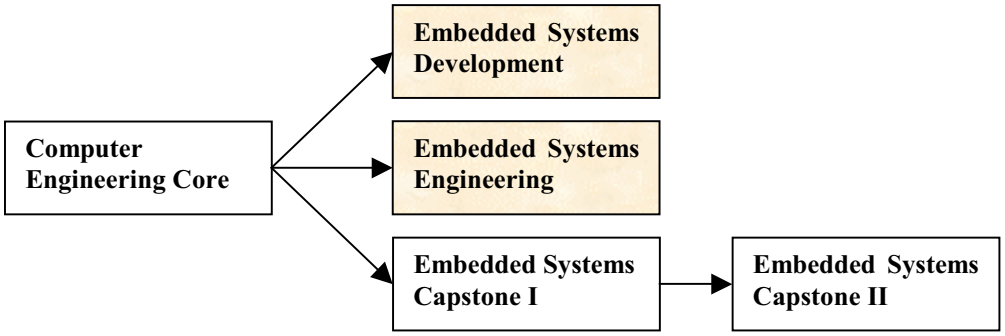

Figure 2: The Embedded Systems Curriculum at ASU [1]

3 Course Development

A meta-goal of the work described in this paper was to enable the study of software product lines by applying the techniques underlying software product line development to the design of course learning activities. Specifically, a software product line approach was directly applied to the development of course projects and had a secondary effect on identifying topics for course laboratories and lectures. In this section we describe the process used in the planning and development of the Embedded Systems Engineering (ESE) course at Arizona State University and the impact of software product line development techniques upon the course development process. The team assembled to perform this process included several PhD and MS graduate research assistants and a supervising faculty member.

3.1 Preliminaries

Steps 1–3 of the course development methodology suggested by Davidson and Ambrose[3] involve a number of preliminary activities meant to help determine course content. This section

summarizes these preliminaries for the ESE course context.

3.1.1 Student Backgrounds and Interests

For the initial offering of the ESE course, the students had primarily a background in Computer Science (rather than Computer Engineering). The students all had at least taken an introductory software engineering course that included project planning, requirements analysis, design, etc. In addition, a majority of the students had received a semester of advanced software engineering with topics including software architecture, object-oriented analysis and design, and design patterns.

3.1.2 Course Objectives and Outcomes

The objectives of the course were based on meeting a number of program goals including the establishment of a concentration track in embedded systems[1]. At a lower level, we were interested in establishing an embedded systems course that would allow computer science students to gain experience developing embedded software. That is, we were interested in a course that catered to more than just computer engineering or electrical engineering students.

Figure 3 summarizes the course objectives and outcomes for the ESE course. The objectives focus upon the design, analysis, and development of applications that utilize both hardware and software. The outcomes indicate the observable characteristics expected from students upon completion of the course. To summarize, the course is intended to address quality of service issues (e.g., safety and reliability), system modeling and analysis, and application development for systems that use both hardware and software components.

As the development of the course progressed, a modest number of objectives and outcomes were modified to reflect the scope of the course. The objectives and outcomes shown in Figure 3 represent the final product of the course development activity rather than an initial working set.

3.1.3 Scope

The scope of the course was determined by examining a number of constraints including both time and funding. For this particular course, generous support was received both from ASU and the National Science Foundation (NSF). The support resulted in a laboratory supplied with host machines and embedded devices for both the ESE course as well as other courses in the concentration track.

It was during the scope stage in the course development that the software product line approach was first considered. The motivating reasons for considering the approach included limited time (15 week single semester) as well as the limited knowledge in student backgrounds necessary to develop full-blown embedded systems applications (e.g., Computer Science, but not necessarily Computer Systems Engineering, or Electrical Engineering).

Benefits of using product line development techniques include short development cycles and increased levels of reuse. With respect to the course objectives and outcomes, we found that the focus on application development and architecture qualities fit well into the strengths of product line development. Specifically, the focus on a software and application architecture in the course objectives was facilitated by the focus on application architecture reuse on the product line side. In addition, the emphasis on component and asset reuse in product line platform approaches fit well into long-term constraints of the course. That is, there was a strong desire to reuse the hardware acquired to support the course. Furthermore, variability analysis allowed us to identify the

1. To be able to understand and address (within a software architecture) the critical issues most often associated with embedded software including high availability, survivability, reliability, and safety

   (a) The student can identify high-integrity concerns such a availability, survivability, reliability, and safety that will impact the design and implementation of an embedded system

   (b) The student can use well-known design patterns in the design and implementation of an embedded systems to address high-integrity concerns such a availability, survivability, reliability, and safety

2. To be able to model and analyze a system in order to observe system characteristics at the system architecture level

   (a) The student can design a system architecture that incorporates both hardware and software behaviors

   (b) The student can analyze a system architecture using different techniques including model-based analysis and simulation

3. To be able to develop systems that utilize both hardware and software components

   (a) The student can implement software for controlling basic embedded hardware components

   (b) The student can implement applications that incorporate the use of both hardware and software components in a networked environment

   (c) The student can integrate hardware and software using middleware technologies such as Jini

   (d) The student can perform system level tests of systems solutions

Figure 3: ESE Course Objectives and Outcomes

family of potential products that could be developed using the product platform. As such, the course projects could be continually updated from semester to semester with little modification of supporting course materials and lectures.

From the standpoint of software product line goals, the scope analysis for course development facilitated the identification of organizational goals for current and future offerings of the ESE course. Specifically, we were able to identify the need for a horizontal platform (e.g., a platform that meets the needs of several markets), where products produced in the platform fall into a certain class of products, and the how the production of those products would meet certain pedagogical goals (namely, embedded systems application development).

3.2 Product Line Development

Our primary efforts in applying the software product line approach for course development were encompassed in the development of appropriate course projects for ESE. As a primary guideline for our efforts, we used a number of process patterns suggested by Clements and Northrop[2]. At the onset, we used the *Factory* process pattern[2]. Figure 4 shows a diagram depicting the components of the Factory pattern, which is a composite pattern that includes the use of the *What to Build*, *Product Parts*, *Product Builder*, and *Assembly Line* patterns. In addition, the Factory pattern is impacted by the *Cold Start*, *In Motion*, and *Monitor* patterns.

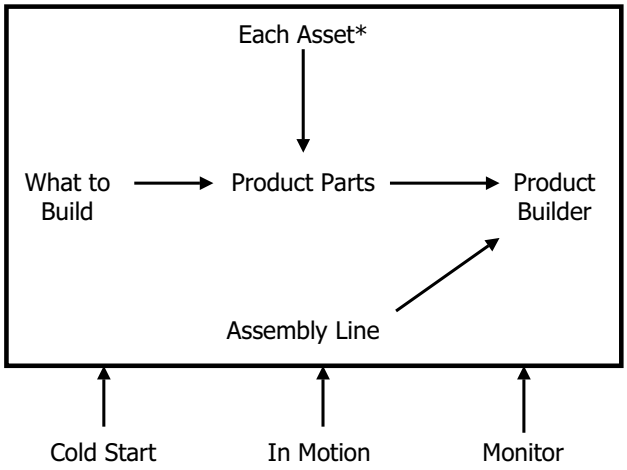

Figure 4: Factory Pattern [2]

The Factory pattern works in the following way. As an initial activity, the *What to Build* pattern is applied in order to determine the scope of the product line and to determine commonalities and variabilities. Upon completion of the *What to Build* activity, the assets are developed (*Each Asset**) and appropriate assets necessary to assemble individual products are identified along with a product architecture (*Product Parts*). Each Asset* represents the patterns necessary to create individual assets. The *Assembly Line* pattern is used to develop organizational production capabilities in regards to developing the products. Finally, the *Product Builder* is the activity of building a product using the product line assets and production capability. In the context of the ESE course, the *Product Builder* and *Each Asset* activities represent the tasks given to students enrolled in the course. The *Assembly Line* pattern represents the lectures and laboratory assignments provided to the students to build up their production capability. All other activities are performed by a course development team consisting of the course architect, research assistants, and teaching assistants.

3.2.1 What to Build

The purpose of the *What to Build* activity is to identify the appropriate scope of the product line and an associated business case that justifies the product line for a target market. The activities that embody this pattern include performing market analyses, studying technology and the domains in which a product release would operate within. With respect to the ESE course, these markets and domains had the constraint of needing to meet course objectives. For the course, it was settled that the market and domain would fall under home automation products. Table 2 shows the products and definitions identified during the scope analysis. In the table, those products listed in italics were ultimately excluded from the product line scope due to the nature of the control necessary to complete the products. However, these products remain as potential products for future product line evolution.

The remaining set of products in the product line scope allowed for easy entry into the "market" through the use of X10-based products and were candidates that facilitated future evolution to more complex product capabilities. In addition to the X10 decision, the Java-enabled TINI processor was selected as the controller platform. TINI is an internet enabled controller that is small, cheap, and

Product	Description
Security	Used to identify when unauthorized personnel gain physical access to restricted areas. The security products consist of sensors that identify when doors, windows, drawers, or other ”locked” areas have been accessed without proper authorization.
Lighting	Used to control the lighting within a control area. Capabilities include on and off functions, timed lighting, and dimmer control.
Environmental Control	Used to control heating and cooling within a control area. Capabilities include on and off functions and environmental programming based on expected access to an area.
Lawn and Garden	Used to control watering functions in a lawn and garden setting. Capabilities include on and off functions, environmental sensing, and timed watering.
Entertainment	Used to control home entertainment functions including audio, visual, and gaming products.
Area Access	Used to control physical access (as opposed to the security products which perform detection). The area access products consist of actuators that physically control locks to doors, windows, and garage doors, etc.
Power Control	Used to control of the On/off and timing of power outlets.
Fire Suppression	Used to monitor and control anti-fire systems including alarms, sprinklers, and notification.
Smart Appliances	Used to control kitchen and other domestic appliances.
Water Control	Used for hot water control, water flow control for managing leaks (burst pipes, unusual water usage levels associated with disasters), water pressure monitoring
Air Quality Monitoring	Used for the monitoring of Radon, CO2, Smoke, and other dangerous accumulations of gases and subsequent notification of their presence.
Swimming Pool Access Control	Used for the monitoring and detection of pool safety and security; entry to pool area, monitoring of floating bodies, splashing, etc.
Swimming Pool Maintenance	Used for the monitoring and control of pool maintenance activities.

Italics indicate exclusion from product line

Table 2: Product Line Scope

simple to use[5]. This decision was motivated by the backgrounds of the students and allowed for a clear justification of the project business case as it would allow students to develop products with a moderate amount of educational overhead. Figure 5 depicts the chosen laboratory environment that includes the use of host machines for development, server or application machines, the target systems (TINI) and the home automation infrastructure (X10).

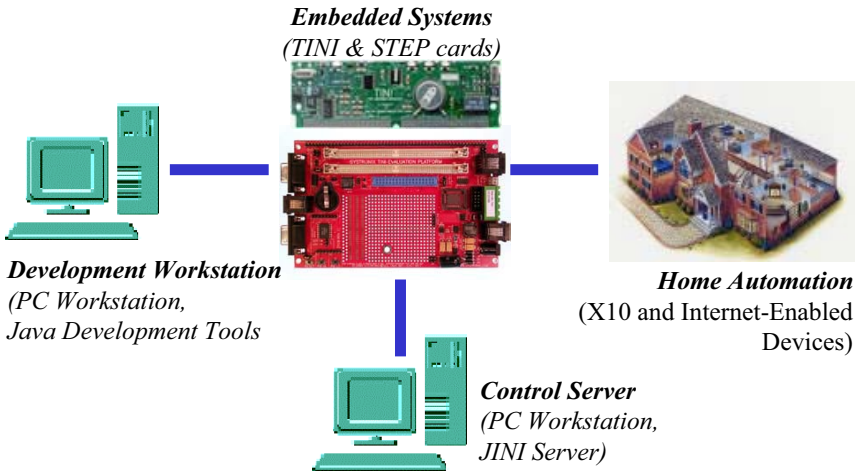

Figure 5: ESE Laboratory Environment

3.2.2 Commonality and Variability Analysis

One of the approaches that was applied to address scope and requirements issues was commonality analysis[6]. This section describes the approach and our experience applying it to course project development.

Process. Weiss describes a process for identifying commonalities and variabilities for *product families*[6]. The approach, called *Family-Oriented Abstraction, Specification, and Translation* or FAST, is a five-step process that involves a team of domain experts along with a moderator in a series of meetings meant to identify the scope and requirements for a software product line or software product family. In the approach, a *commonality* is a product feature that is expected to exist in every possible product within a product line. The term *variability* is the anti-thesis of commonality; it refers to those behaviors that vary between products within the product line.

The five steps in the FAST process include *Preparation*, *Planning*, *Analysis*, *Quantification*, and *Review*[6]. The main activity within this process is the analysis step, which involves the creation of a commonality analysis document. The steps of the analysis activity include creating a dictionary of terms for the given product line scope, identification of commonalities based on the definitions, identification of variabilities, and resolution of issues. With respect to the definitions, Weiss has found that definitions often lead to identification of commonalities[6].

Application to the ESE Course. In addition to identifying a product scope, the course development team was involved in a significant commonality and variability analysis. Using the approach suggested by Weiss[6], the team spent a significant amount of effort in the analyze task by first identifying and refining definitions relevant to the home automation product line and subsequently

translating those definitions into either commonalities or variabilities. To facilitate this activity, a definition to product map was created as shown in Table 3. The table shows definitions along the vertical axis while products are shown along the horizontal axis. The "+" in a given cell indicates that the definition or concept is pertinent to the given product context.

	Product							
Term	Security	Access	Pool Access	Lighting	Lawn and Garden	Power	Fire	Air Quality
Sensor	+	+	+	+	+	+	+	+
Monitor								
Device Controller	+	+	+	+	+	+	+	+
Actuator	+	+	+	+	+	+	+	+
Alarm	+	+	+				+	+
Device	+	+	+	+	+	+	+	+
User	+	+	+	+	+	+	+	+
Unauthorized user	+	+	+					
Authorized user	+	+	+					
Interface	+	+	+	+	+	+	+	+
User Interface	+	+	+	+	+	+	+	+
Validation Authorization	+	+	+					
Access	+	+	+					
Alarm Event	+	+	+					
Event	+	+	+	+	+	+	+	+
Signal	+	+	+	+	+	+	+	+
Receiver	+	+	+	+	+	+	+	+
Transmitter	+	+	+	+	+	+	+	+
Trigger	+	+	+	+	+	+	+	+
Automation	+	+	+	+	+	+	+	+
Home Network	+	+	+	+	+	+	+	+
Web Interface	+	+	+	+	+	+	+	+
Keypad	+	+	+	+	+	+	+	+

Table 3: Definition to Product Map

While this table indicated that several of the definitions may initially appear to be commonalities, we found that variabilities existed in the way certain activities may occur. For instance, while all the products required sensing, variabilities existed in the way each product performed sensing. Nonetheless, Table 3 was used as the basic guideline for constructing the product line commonalities. The product line requirements were formed by using the commonalities as the basic identification of generic product capabilities. Additionally, after a product domain analysis was performed for each individual product, product line variabilities were developed to account for envisioned product behaviors. These definitions of product line requirements, as well as product variabilities, served as the basis for defining course projects. In this respect, the product line requirements serve as the general description of the projects that are intended to remain constant over the lifetime of course offerings, and the product variabilities serve as per offering course project descriptions. Furthermore, since all products are built from a single platform, variability can ex-

ist within a product domain. For instance, within the security product domain, several different security product variations can be created.

3.2.3 Product Parts

The *Product Parts* pattern[2] involves the development and identification of the *parts* needed to construct a product within the product line. Figure 6 shows a diagram depicting the context of the *Product Parts* pattern. With respect to this pattern, the activity involves the development of re-

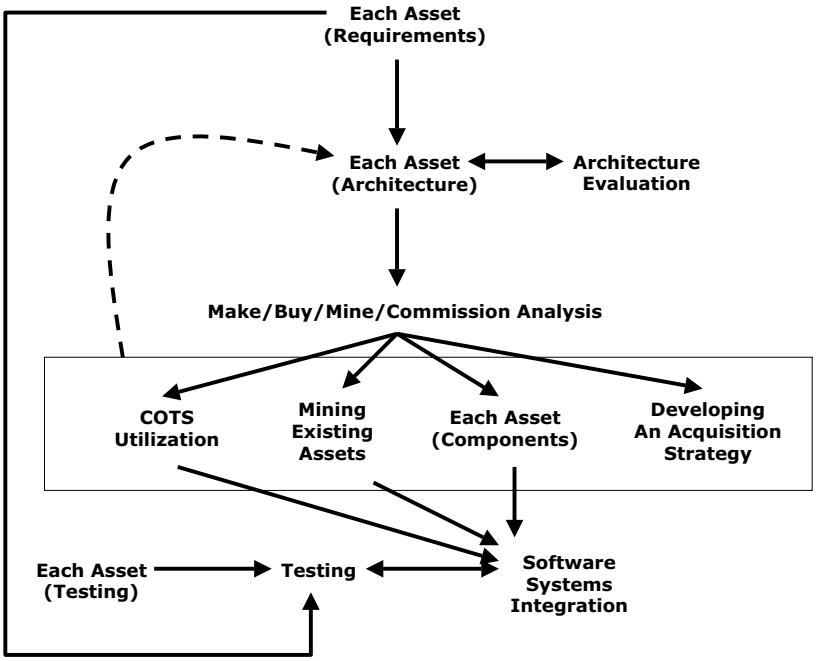

Figure 6: Product Parts Pattern [2]

quirements (*Each Asset** - Requirements), an architecture (*Each Asset** - Architecture), evaluation of that architecture, development of components via a number of different approaches, and integration. This series of activities is a one-time only process and is intended to be used at the product line level. The requirements and architecture along with identification of some of the commercial off-the-shelf (COTS) components were performed by the development team. Other activities are left to the students as part of their product development task.

Technology Assessment. Part of the *Product Parts* activity involved the identification of technology that was appropriate for constructing the end-products. For instance, one of the course objectives was to expose students to the use of middleware as an enabling integration technology. As a result, appropriate middleware technology had to be identified by the course development team that met course constraints (specifically the use of X10 and TINI processors).

3.3 Learning Activities

This section describes the learning activities that resulted as a result of applying the software product line approach to development of the ESE course and course project.

3.3.1 Projects

The *Product Builder* pattern[2] serves as the basis for every course project currently underway or planned in the future for the ESE course. Figure 7 shows a diagram depicting the elements of the *Product Builder* pattern. As is typical with a generic waterfall software product lifecycle, the *Product Builder* pattern begins with a requirements definition for a specific product within the product line. The product requirements lead to the definition of a product architecture, which in turn leads to component development, integration, and finally testing. The difference within this context is that many of the assets needed to complete this process have been developed as part of the *Product Parts* activity, leaving the students to develop the controllers and product specific capabilities. For instance, given a home security product, the product architecture is based on the product line architecture with modifications created in order to handle requirements specific to home security.

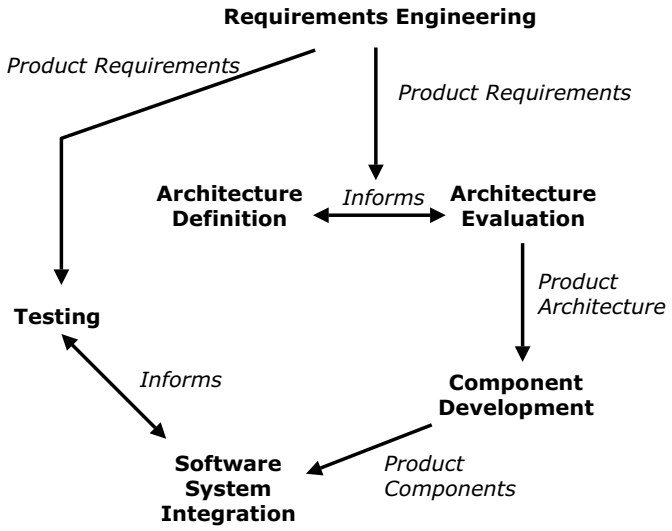

Figure 7: Product Builder Pattern [2]

3.3.2 Laboratories

The course laboratories for ESE consist of exercises meant to familiarize students with the chosen implementation language (Java), the hardware technology (TINI and X10), and the interconnection platform (JINI[7]). The ESE course laboratories were developed with the intention of supporting training on the various technologies to be used within the applications developed from the product platforms. In this respect, the course was designed to use a collaborative learning approach, whereby an initial amount of information is provided through lecture but the responsibility of finding and understanding the technological details is left to the students. The intent being to allow the students the freedom to explore and identify those capabilities that were available for the course projects while leaving the decision on which capabilities to use up to the individual teams. In the context of the Factory pattern, the laboratories were viewed as part of the training, which is part of the assembly line pattern[2].

3.3.3 Lectures

The project for the ESE course dominates the architecture of the course. Specifically, the utility appears to come from the experience gained in the development of the individual products. The focus of the lectures for this course primarily center upon two aspects: general issues for the development of embedded software, general issues necessary to develop the course project (product). The product line notion is in effect hidden from the students in that we do not provide any instruction on the product line approach. The reason for this is that the product development task, which the students are most concerned with appears to be exactly the same structure seen in previous software engineering courses and the fact that a reuse structure underlies that becomes superfluous.

3.4 Lessons Learned

We have found that the product line approach has helped to provide structure to the course project development activity. Specifically, it has facilitated long-term planning regarding the offering of the ESE course and has helped to identify the set of course project assignments that will be used in years to come. Furthermore, the product line approach has made the identification of course laboratories natural rather than contrived. Finally, the creation of the product line has allowed us to develop a software and embedded systems product line that can be used in the context of both education and research.

A number of questions remain open regarding the effort described here. For instance, we have yet to determine the impact on student experiences, for which only time will tell. Specifically, we are interested in determining how the use of the *Product Builder* pattern will impact how projects are developed by students and the effect of structuring a course to mirror product line capable organizations on instruction and final student capabilities. Additionally, we are interested in determining the impact of the product line approach upon course evolution issues. As such, we plan on applying software product line evolution techniques to the issue of course evolution.

4 Related Work

Recently software product lines have been used successfully by several companies including work by Cummins Inc., NRO and Raytheon's CCT[2].

Cummins Inc. Cummins Inc. was in the process of developing six software projects, each working independently of each other and each using its own approach to development[2, pages 417–442]. They also had another twelve projects on the schedule with insufficient staff to handle the additional workload. The solution Cummins took was to stop all current development and instead use a product line approach. Although the change in the way software was developed caused a small amount of employee turnover, organizational capacity increased dramatically. It also resulted in a reduction of both product delievery time and staffing requirements. Another side benefit was improved efficiency allowing for reduced risk in exploring other markets. In their projects they were able to get a 75% rate of reuse while significantly reducing average product cycle time.

National Reconnaissance Office (NRO) and Raytheon. NRO was facing the prospect all government agencies faced in the 1990's, a reduction in their budget and staffing without a reduction in workload. One of the solutions they used was to incorporate a product line approach to the development of their satellite C2 software[2, pages 443–483]. The NRO with assistance from SEI contracted Raytheon to develop a package containing the core assets needed to develop any satellite

C2 software and tools to assist in producing the final product. The NRO knew it would reap future benefits with a reduction in production times and project costs. Raytheon, who completed the CCT project on schedule and within budget, hoped to benefit by winning future contracts in the Spacecraft C2 business. The CCT was used in the development of the Spacecraft C2 System with great success.

In this paper, we describe the application of product line approaches to the development of course projects. The goal is to achieve reuse and cycle time benefits similar to the ones discussed in the above projects.

5 Conclusions and Future Investigations

The benefits of using a software product line approach are beginning to be realized in some fairly significant projects[2]. The strength of the approach is that it allows an organization to realize enterprise level reuse within domains where the scope is well-known. In the context of a project-oriented course, the challenge from semester to semester is the definition of new and innovative projects while maintaining a minimal amount of administrative overhead required to modify projects. By using a product line approach, the scope and infrastructure of course projects become well-defined, thus allowing for significant amounts of variability while maintaining the core infrastructure.

As of the reporting of this work, the course produced based on this product line concept is underway. Initial benefits have been in the area of course planning and development. Specifically, the use of the product line approach has made the definition of course projects, laboratories, and support lectures straightforward. As part of future investigations, we plan on studying how other product line research issues like evolution impact course maturation and development.

Acknowledgements

The authors would like to thank Yu Chen, Shilpa Murthy, Divya Ramasubban, and Swami Venkataramani for their contributions to this work.

Bibliography

[1] G. Gannod, F. Golshani, B. Huey, Y. Lee, S. Panchanathan, and D. Pheanis. A Consortium-based Model for the Development of a Concentration Track in Embedded Systems. In *Proceedings of the 2002 American Society for Engineering Education Annual Conference and Exposition*. ASEE, June 2002.

[2] Paul Clements and Linda Northrop. *Software Product Lines*. Addison Wesley, 2002.

[3] Cliff I. Davidson and Susan A. Ambrose. *The New Professor's Handbook*. Anker Publishing Co., Inc., 1994.

[4] Consortium for embedded and internetworking technologies. [Online] Available http://www.eas.asu.edu/embedded. Arizona State University.

[5] Don Loomis. *The TINI^{TM} Specification and Developer's Guide*. Addison–Wesley, 2001.

[6] D. Weiss. Commonality Analysis: A Systematic Process for Defining Families. In *Second International Workshop on Development and Evolution of Software Architectures for Product Families*, Feb. 1998.

[7] W. Keith Richards. *Core Jini*. Prentice-Hall, 1999.

Biographies

GERALD C. GANNOD is an assistant professor in the Department of Computer Science and Engineering at Arizona State University. He received the MS('94) and PhD('98) degrees in Computer Science from Michigan State University. His research interests include software product lines, software reverse engineering, formal methods for software development, software architecture, and software for embedded systems. He is a recipient of a 2002 NSF CAREER Award.

JOHN J. DOHERTY is a graduate student in the Department of Computer Science and Engineering at Arizona State University. He received a BA('92) in Mathematics from University of St. Thomas. He previously worked at Simula Safety Systems as a Logistics Engineer. His research interests include software product lines and software development for embedded systems.