# Model-driven Testing and Domain Analysis of Product-line Architectures

Jules White, Brian Dougherty, and Douglas C. Schmidt
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37203, USA
{jules, briand, schmidt }@dre.vanderbilt.edu

## Introduction

*Product-line architectures (PLAs)* enable the development of a group of software packages that can be retargeted for different requirement sets by leveraging common capabilities, patterns, and architectural styles [1]. The design of a PLA is typically guided by *scope, commonality, and variability* (SCV) analysis [2]. SCV captures key characteristics of software product-lines, including their (1) *scope*, which defines the domains and context of the PLA, (2) *commonalities*, which describe the attributes that recur across all members of the family of products, and (3) *variabilities*, which describe the attributes unique to the different members of the family of products.

Although PLAs simplify the development of new applications by reusing existing software components, they require significant testing to ensure that valid variants function properly. Not all variants that obey the compositional rules of PLA function properly, which motivates the need for powerful testing methods and tools. For example, connecting two components with compatible interfaces can produce a non-functional variant due to assumptions made by one component, such as boundary conditions, that do not hold for the component to which it is connected [3]. Moreover, the numerous points of variability in PLAs yield variant configuration spaces with hundreds, thousands, or more possible variants. It is therefore crucial that PLAs undergo intelligent testing of the variant configuration space to reduce the number of configurations that must be tested.

A key challenge in performing intelligent testing of the solution space is determining which variants will yield the most valuable testing results, such as performance data. Model-driven engineering (MDE) is a promising approach to help developers analyze product-lines and identify the most valuable areas of the configuration space to test intelligently. MDE uses high-level models of an application to capture solution design properties that cannot easily be represented/analyzed in 3rd generation programming languages or textual documentation. Effectively leveraging MDE to improve test planning and execution, however, requires determining precisely what PLA design properties to model, how to analyze the models, and how best to leverage the results of these analyses.

**Solution approach $\rightarrow$ Model-driven testing and domain analysis of product-line architectures.** This chapter focuses on techniques and tools for modeling, analyzing, and testing PLAs. First, we introduce the reader to feature modeling, which is one of the most widely used modeling methodologies for capturing PLA variability information. Second, we describe approaches for annotating feature models with probabilistic data, obtained from application testing, that can be used to help predict potentially flawed configurations. Next, we present numerical domain analysis techniques that can be used to help guide the production of PLA test plans. Finally, we present the structure and functionality of a FireAnt, which is an open-source Eclipse plug-in for modeling PLAs, performing PLA domain analysis to derive test plans, and automating and orchestrating PLA testing for Java applications.

## Outline of the Proposed Chapter

**Section 2: Motivating Example**. To explore the challenges of testing PLAs, we will present an Enterprise Java Beans (EJB)-based *Constraints Optimization System* (CONST) that assigns freight shipments to vehicles. As shown in Figure 1, CONST manages a list of freight shipments that must be scheduled for pickup, a list of times that the shipments must arrive by, and a list of vehicles and drivers that are available to perform the pickup. It uses a constraint-optimization engine to find a cost effective assignment of drivers and trucks to pickups.

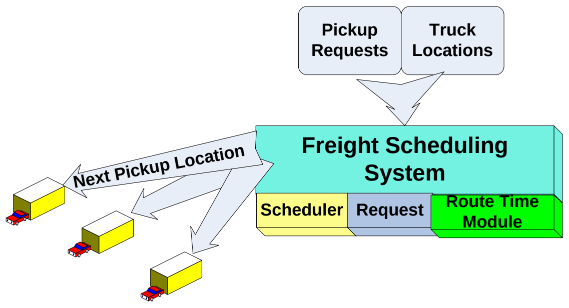

Figure 1: Highway Freight Shipment Scheduling Architecture

CONST's optimization engine can be used to schedule a wide variety of shipment types. In one configuration, for example, the system could schedule oversized-loads, such as mobile homes, whereas in another configuration the system could dispatch hazardous material transport trucks to chemical freight shipments. CONST's optimization engine must therefore be customizable at design-time to handle these various domains effectively.

**Section 3: PLA Modeling, Domain Analysis, and Testing Challenges.** Although PLAs can increase software reuse and amortize development costs, PLAs configuration spaces are hard to analyze and test manually. Deploying, configuring, and testing a PLA in numerous configurations without intelligent modeling, domain analysis, and automation is expensive and/or infeasible. Large-scale product variants may consist of thousands of component types and instances [4] that must be tested. This large solution space presents the following key challenges to developing a PLA:

*Challenge 1 – Creating a model of the PLA's variant solution space.* Traditional processes of identifying valid PLA variants involve software developers determining manually the software components that must be in a variant, the components that must be configured, and how the components must be composed. Such manual approaches are tedious and error-prone and are a significant source of system downtime [5]. Manual approaches also do not scale well and become impractical with the large configuration spaces typical of PLAs. In CONST, for example, there may be thousands of variations on freight types, licensing requirements, freight handling procedures, and local laws applying to transportation that require the PLA to have a substantial amount of variability.

*Challenge 2 – Determining what PLA configurations to test through domain analysis.* With hundreds or thousands of potential configurations, testing each possible configuration may not be feasible or cost effective. Developers must determine which PLA configurations will yield the most valuable information about the capabilities of different regions of the PLA configuration space. Figuring out how to perform this domain analysis is hard. For example, it may not be clear which freight routing algorithms in CONST yield poor performance when used together in a configuration.

*Challenge 3 – Managing the complexity of configuring, launching, and testing hundreds of valid configuration and deployment. Ad hoc* techniques often employ build and configuration tools, such as Make and Another Neat Tool (ANT) [6], but application developers still must manage the large number of scripts required to perform the component installations, launch tests, and report results. Developing these scripts can involve significant effort and require in-depth understanding of components. Understanding these intricacies and properly configuring applications is crucial to their providing proper functionality and quality of service (QoS) requirements [7]. Incorrect system configuration due to operator error has also been shown to be a significant contributor to down-time and recovery [5]. Developing custom deployment and configuration scripts for each variant leads to a significant amount of reinvention and rediscovery of common deployment and configuration processes. As the number of valid variants increases, there is a corresponding rise in the complexity of developing and maintaining each variant's deployment, configuration, and testing infrastructure. Automated techniques can be used to manage this complexity [8,9,10].

*Challenge 4 – Evolving deployment, configuration, and testing processes as a PLA evolves.* A viable PLA must evolve as the domain changes, which presents significant challenges to the maintenance of configuration, deployment, and testing processes. Small modifications to composition rules can ripple through the PLA, causing widespread changes in the deployment, configuration, and testing scripts. Maintaining and validating the large configuration and deployment infrastructure is hard. Moreover, as PLA components evolve, it is essential that intelligent regression testing be performed on PLA variants to identify those that may become non-functional due to unforeseen side effects. For example, a change in a CONST component for assigning costs to shipments may have wide ranging affects on numerous configurations of the optimization engine. With a large variant solution space, it becomes–even more difficult to rapidly evolve and validate the PLA.

**Section 4: Model-driven Testing and Domain Analysis Techniques for Product-line Architectures.** This section will cover MDE-based PLA testing and domain analysis techniques that can address the four challenges described in Section 3. The techniques will focus on:

1. **SCV capture.** Feature models are an MDE technique for specifying the common and variable parts of PLAs. We will introduce the basics of feature modeling and then move to more advanced topics on how feature models can be annotated to incorporate data that can be used to make more intelligent testing decisions.

2. **Numerical domain analysis of feature models.** Feature models can be transformed into a mathematical representation called a *constraint satisfaction problem* (CSP). We will show how CSP domain analysis techniques, developed by Batory, Czarnecki, Benavides, and others can be used to help analyze feature models and identify high-valued configurations to test.

3. **Model-based test infrastructure generation.** Model interpreters can be used to traverse models and automatically generate code, such as configuration scripts, needed to configure, launch, and test variants. We will describe how feature models can be used to generate test harnesses and scaffolding for complex PLAs.

4. **Model-based test plan evolution.** Model-based development approaches allow target artifacts, such as test scripts, to automatically be regenerated as the model changes. We will explain how models can be automatically re-analyzed using numerical techniques to adjust test plans and then regenerate testing infrastructure.

In addition to surveying and taxonomizing PLA testing and domain analysis techniques, we will also present concrete examples of how these techniques can be implemented, using an MDE-based PLA testing tool, called *FireAnt,* that we have developed using Eclipse. FireAnt allows application developers to describe PLA structure using feature models, analyze PLA domains using numerical optimization techniques, and automate test infrastructure generation and orchestrations.

**Section 5: Concluding Remarks.** This section will present concluding remarks, summarize key lessons learned (both pro and con), and provide links to open-source project source code and demo examples.

## Relevance for This Book

Testing is a critical activity in PLA development that is difficult to manage manually. One of the key themes of this book is that modeling and domain analysis can be used synergistically to identify critical software design properties that are not otherwise discernible. This chapter would provide concrete demonstrations of how modeling and analysis can be used to explore complex PLA configuration spaces and improve test planning. Moreover, the chapter would provide software development effort metrics to demonstrate the reduction in complexity of a model-driven approach over traditional manual processes.

## References

1. P. C. Clements and L. Northrop, *Software Product Lines – Practices, and Patterns*, Addison-Wesley, 2001.
2. J. Coplien, D. Hoffman, D. Weiss, „Commonality and Variability in Software Engineering," IEEE Software, Volume 15, Issue 6, Nov.-Dec. 1998 Page(s):37-45.
3. E. J. Weyuker, "Testing Component-based Software: A Cautionary Tale," IEEE Software, September/October 1998
4. D. Sharp, "Avionics Product Line Software Architecture Flow Policies," Proc of the 18th IEEE/AIAA Digital Avionics Systems Conference (DASC), Oct 1999, St. Louis, MO.
5. D. Oppenheimer, A. Ganapathi, D. Patterson, "Why do Internet Services Fail, and What can be Done about It?," Proc of the USENIX Symposium on Internet Technologies and Systems, Mar 2003, Seattle, WA.
6. Apache Foundation: Apache Ant. http://ant.apache.org.
7. A. Krishna, E. Turkay, A. Gokhale, D. Schmidt, "Model-Driven Techniques for Evaluating the QoS of Middleware Configurations for DRE Systems," I Proc of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium, Mar 2005, San Francisco, CA.
8. A. Sloane, "Modeling Deployment and Configuration of CORBA Systems with UML," Proc of the 22nd International Conference on Software Engineering, June 2000, Limerick, Ireland.
9. G. Edwards, G. Deng, D. Schmidt, A. Gokhale, B. Natarajan, "Model-driven Configuration and Deployment of Component Middleware Publisher/Subscriber Services," Proc of the 3rd ACM Conference on Generative Programming and Component Engineering, Oct 2004, Vancouver, CA
10. A. Memon, A. Porter, C. Yilmaz, A. Nagarajan, D. Schmidt, B. Natarajan. "Skoll: Distributed Continuous Quality Assurance," pp. 459-468, 26th International Conference on Software Engineering, May 2004, Edinburgh, Scotland.

## Author Biographies

**Dr. Jules White** is a Research Assistant Professor at Vanderbilt University. He received his BA in Computer Science from Brown University, his MS in Computer Science from Vanderbilt University, and his Ph.D. in Computer Science from Vanderbilt University. Dr. White's research focuses on applying a combination of model-driven engineering and constraint-based optimization techniques to the deployment and configuration of complex software systems. Dr. White is the project leader for the Generic Eclipse Modeling System (GEMS), an Eclipse Foundation project.

**Brian Dougherty** is a Ph.D candidate in Computer Science at Vanderbilt University. Brian's research focuses on hardware/software co-design, heuristic constraint-based deployment algorithms, and design space exploration. He is the co-leader of the ASCENT project, a tool for analyzing hardware/software co-design solution spaces. Brian is also a developer for the Generic Eclipse Modeling System (GEMS). He received his B.S. in Computer Science from Centre College, Danville, KY in 2007.

**Dr. Douglas C. Schmidt** is a Professor of Computer Science and Associate Chair of the Computer Science and Engineering program at Vanderbilt University. He has published 9 books and over 400 papers that cover a range of topics, including patterns, optimization techniques, and empirical analyses of software frameworks and domain-specific modeling environments that facilitate the development of distributed real-time and embedded (DRE) middleware and applications. Dr. Schmidt has over fifteen years of experience leading the development of ACE, TAO, CIAO, and CoSMIC, which are open-source middleware frameworks and model-driven tools that implement patterns and product-line architectures for high-performance DRE systems.