FireAnt: A Tool for Reducing Enterprise Product Line Architecture Deployment, Configuration, and Testing Costs

Jules White
jules@dre.vanderblit.edu
Vanderbilt University, Department of Electrical Engineering and Computer Science,
Box 1679 Station B, Nashville, TN, 37235

Douglas C. Schmidt,
schmidt@dre.vanderbilt.edu
Vanderbilt University, Department of Electrical Engineering and Computer Science,
Box 1679 Station B, Nashville, TN, 37235

Abstract

Product-line architectures (PLA)s are a paradigm for developing software families by customizing and composing reusable artifacts, rather than handcrafting software from scratch. Extensive testing is required to develop reliable PLAs. Each PLA may have hundreds of valid variants that can be constructed from the architecture's components. It is crucial that each of these variants be thoroughly tested to ensure the quality of these applications on multiple OS platforms and hardware configurations. Setting up test environments and running tests can become extremely complex and expensive as the number of variants and the complexity of their deployment and configuration increases. Once a variant is deemed ready for deployment and configuration in a production environment, it is crucial that these activities be done identically to the tested configurations and upholds the assumptions of the component developers. Rapidly setting up numerous distributed test environments and ensuring that they are deployed and configured correctly is hard. This poster paper presents FireAnt, which is a tool for the model-driven development (MDD) of PLA deployment plans.

1. Introduction and Motivation

Product-line architectures (PLAs) offer developers the ability to rapidly produce software packages that are targeted for different requirement sets by leveraging a common set of capabilities, patterns, and architectural styles. The design of a PLA is typically guided by *scope, commonality, and variability* (SCV) analysis. SCV captures key characteristics of software product-lines, including (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.

After constructing a PLA, product-line engineers identify each product variant that must be produced from the architecture and its requirements. Once a PLA variant's requirements are obtained, these requirements must be mapped to the variabilities in the PLA. Traditional manual processes of mapping the requirements to the PLA involve software developers taking each requirement and attempting to determine the software components that must be in the variant, the components that must be configured, how each component must be changed, and how the components must be composed. Such manual approaches are tedious and error-prone and are a significant source of system downtime.

An alternate approach to mapping the variant's requirements to the PLA involves the use of *Model-driven development* (MDD) techniques and tools, which are designed to reduce the effort needed to capture system requirements and map them to the underlying PLA infrastructure. Models of PLA variants developed with MDD tools can be constructed and checked for correctness (semi-)automatically to ensure that application designs meet their requirements. MDD tools can also be used to generate the customization, composition, packaging, and deployment code to implement PLA variants.

2. Overview of FireAnt

FireAnt is a MDD tool designed to allow developers to capture the SCV of their PLA and automate the process of discovering and testing valid variants. It manages the three main challenges of deploying, testing, and configuring PLAs that arise from the large number of variants:

1. **Large Variant Solution Spaces** – A variant may have hundreds of components that can be composed to form a variant. Each component, itself, may have multiple valid configurations. These numerous points of variability lead to a large number of valid variants that each must be thoroughly tested. The large solution space makes it more difficult for developers to anticipate all the possible component compositions and predict accidental complexities.
2. **Complexity of Developing Deployment Scripts** – Any non-trivial variant will require numerous deployment and configuration steps to be installed and tested. Each variant will require different configuration and deployment steps and installation sequences. With such a large number of possible variants, it is very difficult to manage and maintain the deploy-

ment and configuration infrastructure required to deploy, configure, and test variants.

3. **Test Automation and Coverage –** Testing the large numbers of valid variants of a non-trivial PLA is infeasible or prohibitively expensive to do without automation. One solution is only to test a very limited range of variants. This solution, however, is unsatisfactory for PLAs that need to support a wide range of variants or if the variants of interest are not known a priori. With solution spaces that may involve tens, hundreds, or thousands of valid variants, it is essential that testing be automated. Automation is also crucial when the PLA itself is continually evolving and must be regression tested.

FireAnt was developed using the *Generic Eclipse Modeling System* (GEMS), which is an open-source MDD environment built using Eclipse by the Distributed Object Computing (DOC) Group at the Institute for Software Integrated Systems (ISIS) at Vanderbilt University. A GEMS-based metamodel describing the problem domain was constructed and interpreted to create the FireAnt DSML for PLAs. FireAnt models constructed in the domain are used to explore the variant solution space and automate the testing, deployment, and configuration of PLAs. This approach is similar to other model-driven efforts that the DOC group has used for the deployment and configuration of component systems in prior work.

FireAnt's modeling environment makes use of several different views to capture the SCV, deployment, configuration, and testing requirements of a PLA. The user-managed views specify a grammar, visualized as and/or trees, that governs the construction of PLA variants. FireAnt generates the tool managed views by exploring the grammar trees and generating the valid variants.

The user managed views specify the deployment, configuration, and artifact dependency rules for the system. Each PLA is divided into a set of assemblies of components which must be deployed on the target environment. The *Logical Composition View* determines which components must be present in each assembly and what the valid combinations of components are. The *Logical Deployment View* determines which assemblies may be collocated and which nodes each group of collocated components may be used as deployment targets. Figure 1, illustrates the logical deployment view for a constraint optimization system's geo database. The *Dependency View* specifies which physical artifacts, such as Java Archive Files (JARs) and XML files, must be present for each component to function.

FireAnt creates the *Physical Deployment View* by traversing the Logical Composition Tree and calculating all possible combinations of Assemblies that can be deployed to each node. FireAnt then takes each of these possible variants and determines the unique packaging combinations of components (called "eggs") that are required for all possible valid deployments, which allows developers to determine exactly how many unique package configurations are required and which package configurations are required for each deployment configuration of a variant. This design simplifies the job of deploying variants. FireAnt uses the Physical Deployment View to provide automated orchestration of variant deployment and testing. FireAnt can calculate what eggs are needed for each variant and generate the deployment scripts required to install and test them.

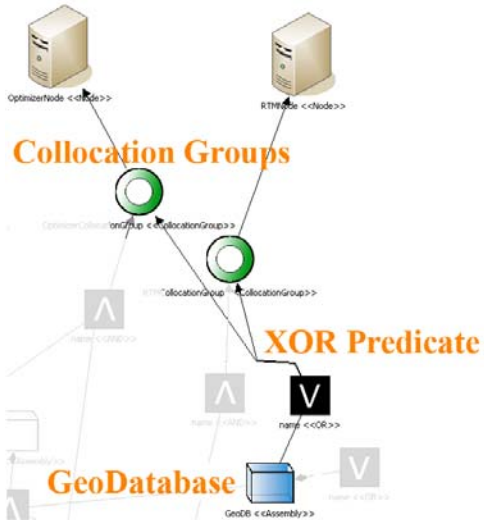

Figure 1, Logical Deployment Tree for the GeoDatabase Assembly

By using model based techniques to formally capture the compositional and deployment variability in a PLA, we have shown that much of the deployment, configuration, and testing of PLAs can be automated. This leaves developers to focus on the implementation of reusable components and deployment and configuration scripts for known working units of functionality. FireAnt can be used to automate the discovery and creation of the deployment, configuration, and testing scripts which glue the PLA's myriad pieces together.

3. Concluding Remarks

FireAnt is an MDD tool, built with the Generic Eclipse Modeling System (GEMS), designed to manage the inherit complexity of deploying, configuring, and testing the large number of possible variants in a PLA. Models of a PLA's deployment and configuration grammar, constructed in FireAnt, can be analyzed by FireAnt's model interpreters to automate the deployment, configuration, and testing of all valid variants of a PLA. Open-source versions of FireAnt are available for download from www.sf.net/projects/fireant. GEMS is also open-source and can be obtained from www.sf.net/projects/gems.