ASADAL: A Tool System for Co-Development of Software and Test Environment based on Product Line Engineering

Kyungseok Kim, Hyejung Kim, Miyoung Ahn, Minseok Seo, Yeop Chang, Kyo C. Kang
Computer Science and Engineering Department
Pohang University of Science and Technology
Pohang, South Korea
{nexters, unicorn, myahn, darkeyes, ranivris, kck}@postech.ac.kr

ABSTRACT

Recently, product line software engineering (PLSE) is gaining popularity. To employ PLSE methods, many organizations are looking for a tool system that supports PLSE methods so that core assets and target software can be developed and tested in an effective and systematic way.

ASADAL (A System Analysis and Design Aid tool) supports the entire lifecycle of software development process based on a PLSE method called FORM (Feature-Oriented Reuse Method) [6]. It supports domain analysis, architecture and component design, code generation, and simulation-based verification and validation (V&V). Using the tool, users may co-develop target software and its test environment and verify software in a continuous and incremental way.

Categories and Subject Descriptors

General Terms
Design

Keywords
Product Line Engineering, CASE Tool, Reuse, Simulation

1. INTRODUCTION

Product line software engineering (PLSE) is emerging as a paradigm that helps organizations develop quality products from reusable core assets rather than from scratch. Since it is a relatively new paradigm, there are few CASE tools supporting the entire lifecycle of PLSE. For example, some tools (RequiLine [12], AmiEddi [8]) support only domain analysis or version management. Others (COVAMOF [11], Ménage [2]) support software design or variability management only. Those who are interested in adopting PLSE need CASE tools that support the entire PLSE process. ASADAL is a tool system that supports domain analysis, software design, code generation, and V&V. Variability management is the key feature that is built into various tools of ASADAL.

PLSE consists of two major processes: asset development and product development. In the asset development process, an asset developer analyzes the domain of a product line and develops architectures and reusable components. In the product development process, a product developer analyzes product requirements, selects features, adapts components and generates codes for a specific product. ASADAL provides capabilities for building various models defined in the PLSE method and generating target source codes from the models.

Copyright is held by the author/owner(s).
ICSE '06, May 20-28, 2006, Shanghai, China.
ACM 1-59593-085-X/06/0005.

Figure 1. Product Line Engineering Processes [6]

Efforts to improve quality of embedded control software systems led to the development of many methods and tools. Of these, simulation is a method that can increase the user’s level of confidence within a relatively reasonable time and cost. Because most embedded control software processes many asynchronous inputs and interacts with environment objects in a complex manner, the concept of “co-modeling” and “co-simulation” has been developed in which the environment together with target software are modeled and simulated [10]. By using ASADAL, test environment of control software can be created rapidly by specifying behavior, function, and form (shape) of environment objects in formal languages and composing objects in object-oriented manner, and simulated in conjunction with the control software. Therefore, co-development makes it possible that software can be designed and tested incrementally from core parts to the whole system. As shown in Figure 2, incremental development and simulation of both the target software system and its environment objects is a salient feature of ASADAL.
Figure 2. Co-Development and Incremental Development of Target Software and Environment in ASADAL

This paper is organized as follows. Domain engineering process and application engineering process based on PLSE method are introduced in sections 2 and 3, respectively. Then, the virtual environment modeling and co-simulation mechanism are explained in section 4. In section 5, we describe our demonstration.

2. DOMAIN ENGINEERING

The domain engineering process consists of activities for analyzing systems in the domain and creating reference architectures and reusable components on the basis of the analysis results. Details on domain analysis and software design are explained in section 2.1 and 2.2, respectively.

2.1 Domain Analysis

At first, the exact scope of the domain and possible interactions with the external world are specified in the context diagram. Based on the understanding of the domain context, commonalities and variabilities in a domain is analyzed in terms of features and modeled in the feature model [4]. Then, feature binding information (when, how, what features are included in a product and delivered to customers) should be analyzed to develop product line components [9].

Figure 3 shows two feature analysis editors of ASADAL. Feature modeling editor is on the left and feature binding analysis modeling editor is on the right. The latter one can automatically extract feature binding units from a feature model. Based on the binding information, the asset developer can decide which features will be made available to customers at asset development, product development, pre-operation, or during operation time.

Figure 4 shows hierarchical behavior specification during operational modeling, functional commonalities and variabilities of the applications in a domain is analyzed via Statechart [3] and DFD [1]. Common functions are specified based on mandatory features and optional and alternative features embed variabilities into model through a hierarchical refinement. For example, one state, which is marked by a dashed rectangle in the upper Statechart in Figure 4, is refined into two different Statecharts at the bottom to accommodate the variabilities identified in the feature model.

2.2 Software Design

Figure 5 shows various design modeling editors of ASADAL based on design models of PLSE. The models are: (1) a conceptual architecture model, expressing abstract high level functional elements of a product line and data flow or control flow between them, (2) a process architecture model, modeling concurrently executable elements and relationships between them, (3) a component architecture model, showing reusable concrete components that will be used to develop a system, and (4) a design object model, modeling objects for implementing the
components. It is possible to instantiate a product-specific design model from the product line design model by allocating features in a feature model to each element in the design model. Then, in product development phase, a product-specific design can be instantiated through feature selection.

Figure 6. Process and Component Specification

ASADAL provides a macro language [6] for the development of flexible and reusable core assets. As shown in Figure 6, developers can use the macro language to embed variabilities in the process and component specifications. Various process or component instances can be generated from the specifications through feature selection and macro-processing. And, behaviors and functions of a process are specified by allocating and adapting some part of Statechart and DFD specifications produced in analysis phase, as shown in Figure 7.

Figure 7. Behavior and Function Specification of Process

3. APPLICATION ENGINEERING

In the product development phase, products can be produced from various core assets developed in the asset development phase. Product developer analyzes product requirements for a specific product, selects features from the feature model, and then selects architecture models, as shown in Figure 8. With this information, ASADAL generates executable code (e.g., Java) by processing macros embedded in various design models and components.

Figure 8. Code Generation of Target Software in ASADAL

4. TESTING

ASADAL provides a method for modeling a virtual environment in which control software will be embedded and tested. Statechart and DFD editors are used for behavior and function specifications of virtual objects, and FSL (Form Specification Language) [10] for 3D shape (form), spatial relations and interactions of objects in the virtual environment. As shown in Figure 9, these form, behavior and function specifications are integrated into a virtual object. Then, virtual objects are incrementally composed, inter-related, and constrained to form larger objects, and finally a testing environment is created.

Figure 9. Environment Object Modeling

These specifications are used to generate code and a virtual environment is simulated interacting with the generated code. As shown in Figure 10, in environment simulation, the behavior/function simulator executes the generated code from Statechart and DFD, and the form simulator simulates transformation of virtual objects based on the values and events from the behavior/function simulator, visualizes spatial objects, and updates spatial relations and interactions between objects periodically. This environmental simulation is controlled by the behavior/function simulator executing target software generated from analysis model and/or design model.
5. DEMONSTRATION

Using ASADAL, we will demonstrate a Home Service Robot (HSR) [5][7] example. The demonstration consists of four phases: (1) domain analysis and asset design (architecture, components) of the HSR product line; (2) application generation from core assets by feature selection; (3) virtual environment modeling for testing; and (4) simulation through interactions between the HSR target software and the virtual environment.

In particular, this HSR example is designed on reconfigurable software architecture and we will demonstrate reconfiguration of various services of HSR while it is running. The simulation phase consists of: (1) simulating different configurations of HSR products of the HSR product line, and (2) binding or unbinding of various services through dynamic reconfiguration of the HSR product as presented in Figure 11.

6. REFERENCES


