DELTAJ 1.5 : A TOOL FOR DELTA ORIENTED PROGRAMMING

Reeta Kumari ¹, Dr. Mahua Banerjee ²

¹Department of Information Technology, Mrs. KMPM Voc.College, Jamshedpur, India
²Department of Information Technology, Xavier Institute of Social Science, Ranchi, India

ABSTRACT:

A Software Product Line (SPL) development approach is used to develop a software family with a common core feature or product along with varying features which make them different. Delta-oriented programming (DOP) is a novel approach for implementing Software Product Lines. In DOP, a product line is implemented by a set of delta modules, which are containers of modifications to a program. A delta-oriented product line is specified by its code base, i.e., the set of delta modules, and a product line declaration specifying the set of possible product variants. DeltaJ 1.5 is a useful tool for the development of Delta Oriented Software Product Lines. It is available in eclipse plugins. The fresh prototypical implementation DELTAJ 1.5 provides full integrated access to the object-oriented features of JAVA. The extensions include delta operations to fully integrate the JAVA package system, to declare and modify interfaces, to explicitly change the inheritance hierarchy, to access nested classes and enum classes, to change field declarations, and to remove overloaded methods unambiguously. It provides the facilities to create a product line using delta modules and different feature along with the configurations of the features and order of delta modules.

In this paper we are going to explain the features and capabilities of DeltaJ 1.5 for developing Delta Oriented Software Product Lines over the earlier versions like DeltaJ 1.0 & DeltaJ 1.1.

Keywords: Software Product Line, Delta-oriented programming, delta modules, code base etc.

[1] INTRODUCTION

A Product Line generally includes the products having some commonalities and variabilities and applied in industries to produce product families. The concept of (hardware) product lines
from industries such as automobiles, avionics, or mobile phones were adapted to software product line engineering (SPLE) [1]. It helped to decrease the drawbacks of maintaining and developing multiple software systems for one domain for the needs of different customers. So, a Software Product Line (SPL) can be defined as is 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]. Various approaches to develop a SPL are Feature Oriented Programming, Aspect Oriented Programming, and Delta Oriented Programming etc. Delta Oriented Programming (DOP) [1] is a flexible programming paradigm to implement SPLs [3] by source code transformation. DOP can also be used directly to support evolution of SPLs, which is in contrast to other approaches, which allow either flexible, fine-grained code manipulation or modularization of code and are not directly capable of handling SPL evolution [4]. The language DeltaJ 1.5 [5] is designed to combine the object-oriented programming language Java with the DOP programming paradigm. DeltaJ 1.5 supports full Java 1.5 features. Delta modules in DeltaJ modularize a Java code base according to the feature selection of a concrete product configuration. A delta module in DeltaJ is included for product derivation depending on the value of a boolean application condition determining whether a combination of features is present in the configuration and support creation of different products or variants. The rest of the paper is organized as:

- Section-2 describes Delta oriented Programming and Delta Oriented Software Product Lines
- Section-3 introduces different versions of DeltaJava
- Section-4 describes features of DeltaJ 1.5
- Section-5 explains the steps of creating a SPL using DeltaJ 1.5
- Section-6 gives conclusion and future work.

[2] DELTA ORIENTED PROGRAMMING & DELTA ORIENTED SPLS

Delta-oriented programming (DOP) is an implementation approach to develop software product lines (SPLs). Delta-oriented SPLs evolve over time due to new or changed requirements and need to be maintained to retain their value [6]. Delta-oriented programming (DOP) is a flexible paradigm to implement SPLs [5]. It’s having the versions as follows:

- **Core DOP**: It is the original formulation of DOP, the product-line code base consists of:
  - A Core Module which contains the implementation of the base product variant (a JAVA program); and
  - A set of delta modules that expresses modifications to the product introduced by the core module, for adding, modifying or removing features. A product line declaration connects the delta modules to the features. An application order or a partial order on the delta
modules is provided to capture the necessary dependencies between the delta modules and to ensure that for every feature configuration a uniquely defined product is obtained. A product is generated by selecting the delta modules associated to the product features and incrementally applying them to the core module according to the application order.

- **Pure DOP**: It dropped the notion of core module. Every product variant is generated only by applying delta modules where the first delta module that is applied can only contain additions.
- In the latest approach a delta-oriented SPL consists of a code base comprising a set of delta modules and a product line declaration linking delta modules to the product features. A delta module encapsulates modifications to an object-oriented program. A particular product in a delta-oriented SPL is generated by applying the modifications contained in the suitable delta modules to a core program which is assumed to be empty that is, without loss of generality [7].

Hence we can define in general a Delta Oriented SPL consists of a Code Base and a Product Line declaration [5]. The code base consists of a set of delta modules which are the container of modifications to the features while the product line declaration expresses the connection between the delta modules and the product line variability specified in terms of product features. As we have defined a delta module is a container of modifications to a JAVA like program. Three kinds of operations are provided to implement modifications.

1. Addition of a new class definition through adds class operation.
2. Modification of an existing class definition through modifies class operation. Within the modifies class operation the following operations can be used.
   (a) Change of the immediate superclass.
   (b) Addition of a new field to the class.
   (c) Addition of a new method to the class.
   (d) Modification of a method defined in the class which replaces the body of the method by a new method body. The new method body may contain calls of the form original(...), that represent calls to the original version of the method.
   (e) Removal of a field defined in the class.
   (f) Removal of a method defined in the class.
3. Removal of an existing class definition through remove operation.

The product line declaration specifies:

1. The set of valid feature configurations.
2. A mapping that associates to each valid feature configuration a set of delta modules that must be used to generate the corresponding product.
3. A partial order of the delta modules (called the application order) that captures the necessary dependencies between the delta modules (which are usually semantic requires relations) and ensures that for every feature configuration a uniquely defined product is generated.

[3] **INTRODUCTION TO DELTAJAVA**
Delta-oriented programming (DOP) is a transformational approach to implement Software Product Lines (SPL) which is very flexible. Product features are mapped to Delta modules which contain the source code. Products are defined by a selection of product features. A product will be generated by applying the Delta modules. DeltaJava or DeltaJ is a programming language which introduces DOP to Java. DeltaJ is available as an Eclipse plug-in. It is based on the Xtext Framework therefore it provides good integration into Eclipse.

Currently there are four DeltaJ prototypes [8]:

- **DeltaJ 1.0 for Core DOP** based on Core DOP. This prototype is available at http://deltaj.sourceforge.net.
- **DeltaJ 1.1 for Pure DOP** based on Pure DOP. This prototype, available at http://deltaj.sourceforge.net/new-version, is based on the notion of Pure DOP.
- **DeltaJ 1.1 with Refactorings** with improved tool support for DeltaJ 1.1. This prototype, available at https://www.tu-braunschweig.de/isf/research/deltas/, is a version of DELTAJ 1.1 for Pure DOP with an improved tool support which provides source code refactoring.
- **DeltaJ 1.5 with Java 1.5** provides full access of the Java 1.5 syntax and corresponding extensions and improvements of the delta-orientated operations. This prototype is available in https://www.isf.cs.tu-bs.de/cms/research/deltas/downloads/plug-in/.

[4] FEATURES OF DELTAJ 1.5

DELTAJ 1.5 is a prototypical language supporting DOP for full JAVA 1.5. The design of DeltaJ 1.5 allows access to the object-oriented features of JAVA 1.5 within DOP which makes DELTAJ 1.5 one of the first programming languages realizing a modular programming paradigm that is fully integrated into JAVA. It supports a notion of delta module that supports program transformations involving the JAVA 1.5 package system, JAVA interfaces, the inheritance hierarchy, nested classes, enum classes and field qualifiers. It also supports a new unified code removal operation (integrated into the delta module construct) and an improved language to express the product line declaration. It is having a built an Eclipse plug-in for DELTAJ 1.5 that provides tool support with syntax highlighting, syntax checking and product generation.

The DeltaJ project structure [9] includes the following:

- **src**: this folder contains the DeltaJ source files
- **src-gen**: this folder contains the auto generated java files, where each package is the translation of an DeltaJ file
- **JRE System Library**: indicates the imported JRE java library
- **.classpath** (hidden): it also indicates that the 'src' and 'src-gen' folders contains java sources

[5] STEPS TO GENERATE A SPL USING DELTAJ 1.5

1. Select File ->new -> Project-> Xtext -> DeltaJ Project
2. Create Project, Features & Deltas : Enter a project name & press next
3. Enter a name for the SPL which is optional if we want to use the name of the project for it
4. Enter the names of the features & delta modules and press finish
We get the SPL defined as below:

Figure-1: SPL FOR SALES RETURNS

5. Next we define the delta modules for adding, modifying & removing the classes.
6. On saving the SPL products are generated according to the feature combination.

[6] CONCLUSION & FUTURE WORK

A SPL consists of a Feature Model consisting of features which denote the abstract description of functionality & Products consisting of group of features. It is developed using a set of reusable artifacts which includes a Base Program (Which is assumed empty) and deltas that denote a set of class operations like adds, removes, and modifies. There is a partial order to define the activation of deltas. DOP aims to maintain modularity & flexibility. It takes the product (set of features) as input and generates the variants as output by applying the activated deltas in order [10]. We found DeltaJ 1.5 can be an efficient tool to generate a Delta Oriented SPL.

Further DOP can be extended to develop Dynamic Delta Oriented SPLs that support runtime configuration and variability management.
REFERENCES

[1] Tim Winkelmann, Jonathan Koscielný, Christoph Seidl, Sven Schuster, Ferruccio Damiani, Ina Schaefer, “Parametric DeltaJ 1.5: Propagating Feature Attributes into Implementation Artifacts”, {ft.winkelmann, j.koscielný, c.seidl, s.schuster, i.schaefer}@tu-bs.de, ferruccio.damiani@unito.it, 1. Technische Universität Braunschweig, Germany, 2. Università di Torino, Italy


[6] Sandro Schulze, Oliver Richers, Ina Schaefer, “Refactoring Delta-Oriented Software Product Lines”, TU Braunschweig Braunschweig, Germany


[8] DeltaJava@ TU Braunschweig

[9] SourceForge.net