svpp 08
Software Variability: a Programmers' Perspective
International Symposium

Better Software Variability Through Better Abstraction of Traversals

Karl Lieberherr
CCIS/PRL, Northeastern University, Boston, Massachusetts, USA

Traversal code is ubiquitous in software, polluting the real intent of programs with accidental details of the current structure in which the intent is expressed.

We present a novel model, called Functional Adaptive Programming (AP-F), for processing objects that supports improved software variability. The software variability is improved by using three approaches: (1) a novel separation of generic concerns (i.e., a functional model for processing objects), (2) constraint-controlled automatic adaptation, and (3) programmer-controlled, crosscutting adaptation using two aspect mechanisms (multi-methods and strategies). Regarding (1), object processing is broken down into three major concerns as in Adaptive Programming (AP): ClassGraph, WhereToGo, WhatToDo. The ClassGraph defines the structural relationships between the classes. WhereToGo uses a high-level traversal language a la AP. WhatToDo is different than in AP and has three subconcerns: Down, Leaf, Up. WhatToDo uses functional-style advice on a general traversal with strategies, inspired by, but different from a visitor. While a visitor uses the traditional Law of Demeter: "Talk only to your friends", WhatToDo uses the dual: "Listen only to your friends". This creates a flow of information through the traversal, separated into what information flows Down, is created at Leafs and flows Up.

Regarding (2), a type checker checks constraints that the multi-methods must satisfy, using strategies to constrain applicable ClassGraphs. Regarding (3), multi-methods and strategies are used to adapt generic WhatToDos in a powerful way: a small change to the program may have a broad effect cutting across the system being built.

A system that supports software variability should have two properties: easy creation of class-graph-generic (or datatype-generic) programs and powerful adaptation and composition capabilities to specialize generic programs. AP-F has both, with advantages over other class-graph-generic programming approaches such as SYB (Scrap Your Boilerplate), Strategic Programming, Polytypic Programming and Generalized Folds. A key feature of AP-F is that it smoothly integrates high-level traversal control with the other desirable features leading to a superior way to abstract a large variety of traversals. This simplifies programs because all traversal-related calls become implicit and they are no longer polluting the real intent of programs.

Examples will be shown in two implementations of AP-F: A Java version, called DemeterF and a Scheme version, called apf-lib.

Joint work with Bryan Chadwick, Ahmed Abdelmeged, and Therapon Skotiniotis.