Generators as Key to Effective Software Reuse
Ira D. Baxter
Semantic Designs, Inc.
idbaxter@scmdesigns.com

We argue that code libraries are a weak form of reuse, and that generative reuse is a more effective means to obtaining the payoff that reuse promises.

A simple and common approach to reuse in many organizations is a code library. While libraries can have beneficial productivity payoffs, the benefits are limited because libraries fail to address a number of key reuse issues:
1) how are entities other than code reused in an effective fashion?
2) how are reusable entities found?
3) how are they combined to make a reliable system?
4) how are they combined to make an efficient system?
5) how is the resulting system maintained?

Generative reuse systems construct programs by combining carefully predefined compatible components to produce application systems. Such system can reuse domain analysis by mechanizing the processing of domain problem descriptions, reuse code fragments at low and high levels of abstraction, and reuse domain engineering by having the problem description choose which components to combine, and knowing how to combine domain-specific components to produce the product software. YACC, widely known, accepts a BNF as a specification, and uses parsing domain engineering knowledge to build efficient parsers combining LALR(1) tables, specifier-provided code fragments, and fixed parser-table processing routines. YACC produces a new, reliable parser in seconds, giving enormous leverage over hand coding. For differential equation solver generators such as Sinapese, several trials of a few hours each are need; still enormously faster than the 6 months required to hand code such an application. If one records the problem specification and how the system combined its components to achieve the product, one captures the key design information needed to understand and therefore maintain the application. If tools exist to help modify that design, then maintenance is directly assisted [Bax92, Bax95]. Thus generative systems directly address these key reuse issues.

There is a significant cost to using generative reuse systems. One must somehow acquire all the infrastructure necessary: specification language, abstract code components, assembly knowledge and mechanism, etc. There are several critical choices implicitly made for each of these systems:
1) what is the structure of a specification?
2) how are components composed?
3) how are components represented?
The specification can be in terms of components to assemble with possible options (components with “features” as in Sinapese), an assembly scheme for designated components (Batory-style “type” expressions), or domain-specific languages (YACC BNF, GUI screen layouts). Domain engineering determines what components are required, and the macro requirements choosing and combining components to satisfy specification fragments. The composition scheme for the system determines how the components are represented and the micromechanics of combination.

A particularly valuable idea is representation of the components and the resulting system at several levels of abstraction. For Sinapese, we were able to analyze the information flow in the abstract, differential equation representation to acquire crucial optimization knowledge impossible to extract from the low level generated FORTRAN.

For DMS, we have chosen to represent components as methods (the micromechanics) that know how to generate code to achieve a particular effect (“sorting”), often by composing abstract procedural components (“abstract mergesort with performance constraints”). The micromechanics are determined by the representation of the procedural components in terms of explicit control and dataflow, which makes capture difficult, but allows excellent control of composition, with the added benefit of significantly simpler analysis and optimization of the composed system.

We believe this type of representation will be valuable for many application domains, and that tools supporting these choices will become the enablers of widespread generative reuse in the future.