ABSTRACT

This dissertation presents a framework for the application of compositional modularity, a module model that facilitates extensive reuse of highly decomposed software.

Compositional modularity supports not only the traditional notions of program decomposition and encapsulation but also effective mechanisms for module recomposition. Based on a previously developed model, a suite of operators individually achieve effects of adaptation and combination on a simple notion of modules viewed as self-referential namespaces. This dissertation extends the previous model by introducing the notion of hierarchical nesting as a composition operation. Furthermore, this work shows that compositional modularity is unifying in scope. Important effects and idioms of advanced modularity, including several varieties of inheritance in object-oriented programming, find convenient expression within this model.

Compositional modularity can be applied within a wide range of systems that manipulate self-referential namespaces. To demonstrate, four distinctively differing systems based on the model are presented: an interpreter for a module extension to the programming language Scheme, a programmable linker for composing compiled object files, a compiler front-end for a compositional interface definition language, and a compositional document processing system. It is shown that systems such as the above derive important benefits from incorporating compositional modularity.

To facilitate the application of compositional modularity, the model is itself realized as a generic, reusable software architecture — an object-oriented *application framework* named ETYMA. ETYMA comprises a collection of interacting classes corresponding to the essential concepts of the model. The framework may be reused to efficiently build *completions*, i.e., tools for compositionally modular