Applying Formal Concept Analysis to Extract Framework Reuse Interface Specifications from Source Code

Jukka Viljamaa

Department of Computer Science
P.O.Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki, Finland
jukka.viljamaa@cs.helsinki.fi

Helsinki, August 2004, 206 pages
ISSN 1238-8645
ISBN 952-10-1949-2 (paperback)
ISBN 952-10-1950-6 (PDF)

Abstract

For decades, reuse of software components has been acknowledged to be one of the most effective means to reduce software development costs and improve product quality. Gradually, we have also learnt that reuse of software architecture and high-level design is equally important.

Object-oriented application frameworks provide an established way of reusing the design and implementation of applications in a specific domain. Using a framework for creating applications is not a trivial task, however. The complexity, variability, and abstract nature of frameworks make them difficult to specialize. It is therefore essential that a framework is delivered with proper documentation describing its purpose, usage, overall architecture, and detailed design.

For an application developer it is particularly crucial that the framework’s reuse interface is well documented. Design patterns provide a means to express that interface in a systematic way. Special tools are also valuable in supporting the framework specialization process. In our JavaFrames framework engineering environment, a pattern formalism is introduced to enable task-driven assistance for framework specialization. Based on the patterns that are used to specify framework reuse interfaces, JavaFrames offers code generation, dynamically adjusted user documentation, and the validation of architectural constraints.

Unfortunately, reuse interface specifications typically become quite extensive and complex for non-trivial frameworks. In this thesis, we discuss the possibility of reverse engineering a reuse interface specification from the source code of a framework and its example applications. A formal concept analysis algorithm is adapted to produce JavaFrames specialization patterns from Java source code, and the algorithm is implemented within the JavaFrames tool set. In two case
studies, automatic pattern extraction is conducted for the Struts and JUnit frameworks. The effectiveness of the method is assessed through comparisons to manually prepared annotations of the same frameworks.

The case studies indicate that it is possible to specify the intended rules governing the framework’s specializations with a precise pattern-based formalism to allow effective tool support for framework specialization. It is also possible to automatically extract a considerable portion of those specifications from source code, which makes reuse interface modeling faster and can also increase the quality of the produced models.

Computing Reviews (1998) Categories and Subject Descriptors:
D.2.2 [Software Engineering]: Design Tools and Techniques – computer-aided software engineering (CASE), modules and interfaces, software libraries
D.2.6 [Software Engineering]: Programming Environments – integrated environments
D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement – reverse engineering
D.2.13 [Software Engineering]: Reusable Software – reusable libraries, reuse models

General Terms:
Design, Documentation, Experimentation, Languages

Additional Key Words and Phrases:
Formal Concept Analysis, Framework, Pattern, Reuse, Reverse Engineering