IDEA GROUP PUBLISHING

701 E Chocolate Avenue, Hershey PA17033, USA
Tel: 717/533-8845; Fax: 717/533-8661; URL: www.idea-group.com

ITB7254

Chapter V

Designing Component-Based Heuristic Search Engines for Knowledge Discovery

Craig M. Howard
Lanner Group Limited, UK
University of East Anglia, UK

The overall size of software packages has grown considerably over recent years. Modular programming, object-oriented design and the use of static and dynamic libraries have all contributed towards the reusability and maintainability of these packages. One of the latest methodologies that aims to further improve software design is the use of component-based services. The Component Object Model (COM) is a specification that provides a standard for writing software components that are easily interoperable. The most common platform for component libraries is on Microsoft Windows, where COM objects are an integral part of the operating system and used extensively in most major applications.

This chapter examines the use of COM in the design of search engines for knowledge discovery and data mining using modern heuristic techniques and how adopting this approach benefits the design of a commercial toolkit. The chapter describes how search engines have been implemented as COM objects and how representation and problem components have been created to solve rule induction problems in data mining.

This chapter appears in the book, *Heuristic and Optimization for Knowledge Discovery*, edited by Ruhul Sarker, Hussein Abbass and Charles Newton. Copyright 2002, Idea Group Inc.

BACKGROUND

In traditional software projects the application was a single executable, built by a specific compiler and linker for a particular operating system and platform. As projects got larger, code was separated out into modules, or libraries, to improve the management of the source code but ultimately was still built into a single program file; this is referred to as static linking of libraries. An improvement on static libraries was dynamic libraries, where only the prototypes of available library functions are supplied to the core program at compile time. When the program is executed, the operating system has to locate the library using path rules to scan the file system and dynamically link the two together. As well as further improving the management of larger projects, dynamically linked libraries have the added advantage of being reusable by multiple programs. For example, when two programs use the same graphics library to build the user interface, the programs are built using function prototypes and require only a single copy of the dynamic library to be located on the system. By removing common functionality from the executables, the overall size of the executables is reduced and only one copy of the library is required to support multiple programs. In addition, isolating the shared functions is not only good software engineering practice but also reduces problems caused by numerous copies of the same code in different locations, making projects and version control easier to manage. Even with these improvements there are still a number of drawbacks. Firstly, if the library needs to be updated or changed (maybe for just one program), existing programs will need to be rebuilt to use this new version even if the prototypes have not changed and the library is backward compatible with earlier versions. Secondly, sharing libraries between different platforms and even compilers is not straightforward, for example, a dynamic library created for Digital Unix would not work under Solaris, and getting a Windows dynamic linked library (DLL) written in Microsoft Visual Basic to work with Borland C++ is not always an easy task.

The problem being addressed in this chapter is the design of component-based software with a particular application to rule generation for data mining. The problem of generating rules describing a class of records in a database can be formulated as an optimisation problem (Rayward-Smith, Debuse, & de la Iglesia, 1996). Heuristic search engines can be used to generate and evaluate rules whilst taking into account a number of constraints such as limiting the complexity of the rule and biasing either accuracy or coverage. A feasible solution in the data mining problem domain is represented by any valid rule that can be evaluated against the database. A number of data mining toolkits using techniques such as genetic algorithms and simulated annealing are listed on the KDNugetts Web site (KDNuggets, 2001), in particular the Datalamp toolkit (Howard, 1999a), which is based on much of the work described in this chapter.