Intelligent Support to Program Development

Pavol Návrat*, Mária Bieliková, Igor Koziak, Viera Rozinajová, Mária Smolárová †
Slovak Technical University, Bratislava, Slovakia

Abstract

Intelligent support to program development is to be understood in quite a broad sense. From the point of view of activities to be supported, all the phases of the software life cycle deserve our attention. From the point of view of possible approaches to accomplishing the support, we consider using artificial intelligence techniques when appropriate, but do not restrict ourselves to any single approach.

Research reported here aims to tackle some of the problems connected with intelligent support to program development. Because the area is broad, we do not attempt to devise a single method that would solve the problem. At the present stage, our aim is more modest: to concentrate on some selected particular questions and to devise methods of solving them. In particular, we have devoted out attention to the following topics:

- Knowledge base on selecting data types in the process of program design,
- Programing paradigms and techniques that allow incorporating various kinds of knowledge[a],
- Reusing design patterns,
- Analyzing program to recognize program cliches,
- Building a software system configuration.

In the following, we give a very short description of goals, methods, and results achieved in researching the above topics so far.

Knowledge Base on Selecting Data Types

The relevant part of programming knowledge concerns various structured data types, both static and dynamic. Each data type has some properties which serve as clues for selecting them for certain class of applications. We have represented the system of knowledge pieces which mirrors the system of data types as system of classes and subclasses. Other classes of objects are problem specifications or problem solving methods. Specific knowledge pieces of programming know-how can conveniently be represented as rules. We have conducted several experiments[6]. Such externalisations of knowledge can be extremely important in areas where no mature scientific formalised theory could have been developed so far.

*Address for correspondence: Slovak Technical University, Dept. of Computer Science and Engineering, Ilkovičova 3, 812 19 Bratislava, Slovakia. E-mail: navrat@elf.stuba.sk. Fax: (+42 7) 720 415.

†The work reported here is partially supported by Slovak Science Grant Agency, grant No. G1339/94.
[a]This work has also been partially supported by COPERNICUS project No. CP 93:6638.

Programming Paradigms and Techniques

Here, programming techniques for representing various kinds of knowledge, including meta level modules are investigated. A Prolog technique has been proposed that allows structuring Prolog programs into modules. Moreover, meta level knowledge can be conveniently expressed. The technique was found useful in devising a query optimization procedure [1]. Properties of programming paradigms are studied and diverse application domains are discussed. For example, constraint logic programming was discussed in [7]. A case study of using logic programming and constraint logic programming for representing calendrical calculations was conducted, and the paradigms were compared [5]. A formulation of a method of programming is discussed that would better reflect the fundamental relations of abstraction, generality, etc.[4].

Reusing Design Patterns

Sofware reuse is identified as perhaps one of the most promising directions towards achieving progress in program development. Research was aimed to describe the current state of the art [8]. Next, we attempt to propose suitable representation techniques that would allow reusing design patterns.

Program Analysis

Analysis of programs is an important activity that takes place at different stages of the software life cycle. Recognition of standard programming constructs is of particular interest in the process of teaching programming. Our approach is based on having a library of cliches. A program is analyzed in order to recognize cliches that were used to build it. We devote our attention mainly to the problem of recognizing recursive cliches [3].

A Knowledge Based Method for Building a Software System Configuration

A method for building a software configuration from its model and requirements describing its properties is proposed. Our model of software systems reflects architectural and development-induced relations among component families and variants. The method builds a generic configuration first and then proceeds to building a bound one. In building both of them, a method for version selection plays an important role. It is controlled by heuristics supplied by a software engineer[2].

References

[1] M. Bieliková, B. Finance, G. Gardarin, Ľ. Molnár, P. Návrat, M. Smolárová, and Tang Z.-H. Modeling a query optimizer with multi-level logic programming. Technical report, STU - University of Versailles-St-Quentin, Bratislava - Versailles, 1995.

[2] M. Bieliková and P. Návrat. A knowledge based method for building a software system configuration. *Knowledge Based Systems*, to appear, 1996.

[3] I. Koziak. Improved graph-parsing algorithm for program analysis. In J. Breza, D. Donoval, and R. Redhammer, editors, *Proc. 3rd International Conference on Computer Aided Engineering Education CAEE'95*, page 137, Bratislava, 1995.

[4] P. Návrat. Hierarchies of programming concepts: Abstraction, generality, and beyond. *ACM SIGCSE Bulletin*, 26(3):17–21,28, 1994.

[5] P. Návrat and M. Bieliková. Representing calendrical algorithms and data in Prolog and Prolog III languages. *ACM SIGPLAN Notices*, 30(7):45–51, 1995.

[6] P. Návrat and V. Rozinajová. Making programming knowledge explicit. *Computers and Education*, 21(4):281–299, 1993.

[7] P. Návrat and M. Smolárová. Use of PrologIII in teaching artificial intelligence techniques. In J. Breza, D. Donoval, and R. Redhammer, editors, *Proc. 3rd International Conference on Computer Aided Engineering Education CAEE'95*, pages 311–316, Bratislava, 1995.

[8] M. Smolárová. Software reuse: Principles, methods, application. Technical report, Slovak Technical University, Bratislava, 1995.