ADAPTIVE REUSABLE COMPONENTS IN PROGRAM GENERATION SYSTEMS

L. P. Babenko, E. M. Kokhova,
V. V. Sinyagovskaya, and
M. R. Taranovskii

The notion of integrating program specification that allows automatic program generation is described. A special problem decomposition method is considered, which allows accumulation and adaptation of reusable components and integration of different aspects of specification.

INTRODUCTION

We describe an integrating program specification system which allows automatic program generation. The distinctive features of the system are the following:

— availability of a special problem decomposition method which makes it possible to organize program development as integration of ready-made reusable components in the program specification stage;

— adaptivity of the reusable components to specific application domains and changing user requirements;

— integration of various aspects of program specification into a single environment, capable of generating both stand-alone data-processing programs and workstation software systems for managerial decision support.

MAIN CONCEPTS

The target user is the person for whom the data-processing system is intended and the target information system (TIS) is the system of workstation-support programs. The TIS is maintained in operational state by system analysts and application programmers, who are responsible for the analysis and formalization of knowledge about the application domain, generalization of this knowledge, and its representation in the form of systems of reusable components. They are also responsible for maintaining the system in a form that meets the target user’s changing needs.

One of the criteria for software quality is adaptability to the changing environment. The notion of software portability relative to the "executing" hardware should be extended to cover other unstable factors in the software environment, such as the operating system, the data management system, user and program interfaces, system of application-domain concepts, data transformation functions, etc. To improve program portability, each of these factors, acting as a possible cause of "destabilization" of program execution, must be localized to the greatest possible extent, so that its modification can be carried out with a minimum of effort. In accordance with this requirement, the problem is decomposed into components, with each component related only to one of the factors and reflecting only the associated aspects of problem solving. Program specification is divided into textually localized parts, each representing the aspects of a particular factor.

Knowledge formalization and identification of reusable modules corresponding to a particular factor is carried out while ignoring the possible effect of the other factors. The factors are thus treated as "mutually orthogonal." As a consequence, the specification language based on reusable modules has a clearly pronounced modular structure, founded on the concept of functional processing module (FPM), first introduced in Cobol. Each FPM is a collection of language constructs that relate to clearly defined aspects of problem formulation corresponding to one of the destabilizing factors. The constructs of a particular FPM may be modified independently of other FPMs.
A classification of abstraction types that determine the composition of FPMs for data processing tasks is given in [1]. Examples of FPMs include access to data in a particular DBMS; a loop that scans the available data-access FPMs and evaluates the resulting situations; standard data aggregation functions; editing of print files; organization of man–machine interaction.

It is important to bear in mind that the collection of constructs forming a separate FPM is designed as a compatible set of reusable modules intended for serving a particular circle of target users and can be adapted to meet changing target-user requirements. The specification languages formed by the collection of FPMs have a much higher level of problem orientation than the traditional programming languages; compared to database languages, they have a much higher universality and more flexible composition tools for creation of composite functions from relatively simple operators, which are functionally orthogonal, simple to code, and compatible across interfaces. The FPM constructs are not restricted to procedure definition operators. There are tools intended for providing a user-friendly “view” of the data structure and for establishing concepts that ensure generalization, classification, and other abstractions of data values. Compositions of reusable modules from different FPMs may form complex reusable modules parametrized so as to allow their use with a minimum knowledge of programming.

The set of relevant problems for a fixed group of target users can be interlinked by specifying the interaction rules between different problems and the interaction rules between the problem and the user of the results. This interlinked set, in turn, may be used as a reusable module. Thus, the system allows a wide spectrum of reusable modules, ranging from the constructs of a single FPM to the limiting case when the composition of various reusable modules constitutes a specification of a computerized decision-support workstation. Examples of parameters of such composite reusable modules are data names and values, introduced concepts, data reliability criteria, calculation formulas, and other formalized knowledge about the application domain stored outside the program specification process.