International Journal of Soft Computing 10 (5): 344-347, 2015
ISSN: 1816-9503

Methods of Integration and Execution of the Code of Modern Programming Languages

R.A. Valiyev, L.A. Galiullin and A.N. Iliukhin
Branch of Kazan Federal University in Naberezhnye Chelny, 10A Sujumbike Avenue,
The City of Naberezhnye Chelny, The Republic of Tatarstan, Russian Federation

Abstract: The study shows that in the modern projects for software design the need for a specialized programming tool for solution of tasks relating to a particular subject area arises frequently. However, there is no universal programming language that would be equally efficient when applied in any area. In this case, the most efficient method is application of the domain-specific languages dedicated to solution of a narrow range of tasks. At the same time, design of a specialized programming language is labor-consuming and expensive process that is not always possible within the frameworks of a particular project. The review of the methods of extension of the modern programming languages specified by the researches and used for classification of extensions by method of integration and execution of the extension codes is provided. The application of extensions in the real programming systems with examples of the source codes in the extended languages is considered, the advantages and disadvantages of each of the methods discussed are analyzed. The simplest and the most frequently used methods of integration of the extension code into the core code not requiring changes in the basic programming system are considered. In the presented review, the information about kinds of extensions in the modern programming languages, about the methods of integration and execution of the extension codes is systematized. This material provides the idea of the existing enhanced capabilities in known programming languages that allows orientating in the variety thereof and choosing the most appropriate tool for solution of the specific specialized tasks in the most convenient and efficient way with the use of programming language extensions. This information will also be useful by selection of the optimal methods for implementation of proprietary extensions if necessary. The methods described in this study have been used by development of the application for an automated information system for diesel engine testing.

Key words: Programming languages, software, design (engineering), development, information architecture, information systems, programming methods, code integration

INTRODUCTION

In the modern projects for software design the need for a specialized programming tool for solution of tasks relating to a particular subject area arises frequently (Wei *et al*., 2014). The modern programming languages of common purpose (for example, C, C++, Object Pascal, Java) provide to a developer a wide range of options, however, the use thereof within a specific application environment may appear to be inconvenient and inefficient. This means there is no universal programming language that would be equally efficient when applied in any area (Wei *et al*., 2014; Ferreira and Berretta, 2014; Wu *et al*., 2015; Gibbons, 2015). In this case, the most efficient method is application of the domain-specific languages dedicated to solution of a narrow range of tasks (Ferreira and Berretta, 2014). At the same time, design of a specialized programming language is labor-consuming and expensive process that is not always possible within the frameworks of a particular project. Besides, often along with the specialized options a programming language shall at the same time feature a wide range of functional capabilities peculiar to the languages of common purpose (Wu *et al*., 2015). As the result of many years of history of the programming languages design the developed languages of common use appeared that feature many benefits, provide required basic functionality and feature the comprehensive tool support. By virtue thereof instead of designing a completely new programming language in many cases the optimal solution is application of different methods of alignment of a code in the basic programming language of common used with the code of the dedicated extensions in other words-implementation of extensions of the already existing programming language. The present review has been prepared for the purpose of forming an idea of the variety of dedicated tools in the modern programming languages and systems as well as

Corresponding Author: R.A. Valiyev, Branch of Kazan Federal University in Naberezhnye Chelny, 10A Sujumbike Avenue, The City of Naberezhnye Chelny, The Republic of Tatarstan, Russian Federation

orientating a software developer by selection of the appropriate methods where there is a need to implement appropriate extensions of the programming languages. In total, the authors have distinguished 8 methods of the programming languages extension: four methods of integration of the extension code into the programming languages and four methods of the extension code. Besides, the analysis of the domain-specific extension of the traditional programming language by means of the task-solving tools followed by data in the tabular form has been performed (Achten *et al*., 2015) with substantiation of selection of the appropriate methods for extension of the core language (hereinafter-CL). This extension has been implemented (Bonami *et al*., 2015) on the basis of the programming aids of the ERA system (ephemeral estimates in astronomy) (Carvalho and de Oliveira, 2015) and is used for solution of the ephemeral astronomy tasks.

CLASSIFICATION OF EXTENSIONS BY METHODS OF INTEGRATION IN THE CORE LANGUAGE

In the researchers opinion in general, according to the experience available, a few main methods of alignment (integration) of different language structures within one program may be distinguished and used for classification of extensions of the modern programming languages. It is also proposed to use the term 'extension method' (more specifically, the method of the extension integration or the method of the extension execution) in respect of each of such methods. Classification of extension by methods of integration thereof in the CL represents a list of the following categories (here they are presented in the order of enhancement of expressivity of the linguistic means used for description of the dedicated extensions):

- Enhanced capabilities are implemented separately, usually in a different language and are available in the CL program through the API interface
- The CL extensions are implemented in the same language and are available through a particular CL subset
- The structures of another language are introduced to the text of the CL program 'as is' but in the form of a string value transferred as a parameter in the API interface for interpretation thereof
- The core language is extended through the new language structures

It shall also be noted that in the real programming system a single method is rarely used for enhancement of the language capabilities (Fomeni and Letchford, 2014). Most frequently a combination of two or more methods is used.

INTEGRATION THROUGH API

This is the simplest (linear) method of the extension introduction to the CL to the extent that enhancement of the CL expressive capabilities does not take place (however, the new functional capabilities appear) the extended options are introduced in the program text by ordinary calling the functions, subprograms, handling the objects, modules, i.e., by means of the API interface expressed in the CL structures. Implementation of the enhanced capabilities themselves is performed separately from the basic program, most frequently in another programming language and is provided in the form of the ready executable code as a rule without the source text in the form of the library object modules or DLL (and in some cases is embedded in the execution program run-time system). In this case the code of the extension execution is related to the object code of the basic program at the stage of the application assembly by a linker and in case of the extension implementation in the DLL form, dynamically by means of calling functions from DLL during the program running by the operating system aids. This is a rather common method of introduction of enhanced capabilities in the programming language of common use. The shortcomings of this method that can be mentioned are: limitedness of the expressive means of extensions represented by external variables and functions only as well as overhead costs of execution of the external functions the amount of which may be rather substantial.

INTEGRATION IN THE FORM OF THE CORE LANGUAGE SUBSET

This is a relatively technically simple method of the extension introduction to the CL of common use to implement dedicated capabilities in the same CL and to provide them to a user (programmer) in the form of a CL subset featuring the domain-specific semantics and a specific set of rules (formal requirements) a user shall follow to be able to work with these dedicated capabilities. However, this method is not always efficient. In cases, when the semantics of extensions and semantics of the core language is rather or completely incompatible the

selection of this method may result in introduction of such restrictions as to the use of CL expressive means that will cause inconvenience of programming, result in increase in the volume of manual coding and as a rule, inefficiency of execution of such code. And the source code will be characterized by poor readability and support difficulty.

INTEGRATION BY MEANS OF STRING PARAMETERS OF API

In this case, the program code in the specialized language is written in the form of a string value to the text of the CL programming program and this string is transferred as a parameter through the API for aces to the extension execution library. For example, one may print the query in the SQL language and use it as a DAO object parameter in the program in VBA in Microsoft Access or execute instruction in the similar way in the DDL language.

INTEGRATION BY MEANS OF NEW LANGUAGE STRUCTURES

All three methods of introduction of enhanced capabilities in the CL do not change the existing language structures and differ only through the method of integration of the basic program with the external execution of extensions. The fourth method considered here represents the deepest integration of the CL expressive means with the extension structures.

We mean, namely, the literal overlapping of the program code in two languages within a single program which is more convenient to a user (programmer) for a number of reasons.

Firstly, it becomes possible to express the dedicated functionality with the use of the most appropriate language structures and secondly, searching for errors during the application design process is facilitated, since the syntactic and the semantic control of the extension structures along with the control of the CL structures may be performed by a compiler at the translation stage.

Usually by such overlapping the text in the specialized language is marked in the program code in the CL by means of the surrounding marking structures that are easily lexically allocated sometimes even at the stage of pre-processing of the program text (the stage of preliminary analysis prior to parsing). In other cases, the presence of the specialized structures is unambiguously defined from the context and no such allocation is required then within a single program a 'mix' of the language structures with no externally-defined boundaries appears.

SUMMARY

The methods described in this study have been used by development of the application for an automated information system for diesel engine testing (Zubkov and Galiullin, 2011; Biktimirov *et al*., 2014).

CONCLUSION

In the presented review, the information about kinds of extensions in the modern programming languages, about the methods of integration and execution of the extension codes is systematized.

This material provides the idea of the existing enhanced capabilities in known programming languages that allows orientating in the variety thereof and choosing the most appropriate tool for solution of the specific specialized tasks in the most convenient and efficient way with the use of programming language extensions. This information will also be useful by selection of the optimal methods for implementation of proprietary extensions if necessary.

REFERENCES

Achten, P., P. Koopman and R. Plasmeijer, 2015. An introduction to task oriented programming. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8606: 187-245.

Biktimirov, R.L., R.A. Valiev, L.A. Galiullin, E.V. Zubkov and A.N. Iljuhin, 2014. Automated test system of diesel engines based on fuzzy neural network. Res. J. Applied Sci., 9 (12): 1059-1063.

Bonami, P., A. Lodi, A. Tramontani and S. Wiese, 2015. On mathematical programming with indicator constraints. Mathematical Programming, pp: 33.

Carvalho, L.B. and P.P.B. de Oliveira, 2015. Extending standard evolutionary programming with self-adaptive stable distributions. International Journal of Parallel, Emergent and Distributed Systems, pp: 29.

Ferreira, D.J. and L.O. Berretta, 2014. Facilitation of creativity in programming. IEEE Latin America Transactions, Art. No. 7014524, 12 (8): 1530-1538.

Fomeni, F.D. and A.N. Letchford, 2014. A dynamic programming heuristic for the quadratic knapsack problem. INFORMS J. Comput., 26 (1): 173-182.

Gibbons, J., 2015. Functional programming for domain-specific languages. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8606: 1-28.

Wei, Q., F.Y. Wang, D. Liu and X. Yang, 2014. Finite-approximation-error-based discrete-time iterative adaptive dynamic programming. IEEE Transactions on Cybernetics, Art. No. 6912005, 44 (12): 2820-2833.

Wu, Z.Y., J. Tian and J. Ugon, 2015. Global optimality conditions and optimization methods for polynomial programming problems. Journal of Global Optimization, pp: 25.

Zubkov, E.V. and L.A. Galiullin, 2011. Hybrid neural network for the adjustment of fuzzy systems when simulating tests of internal combustion engines. Russian Eng. Res., 31 (5): 439-443.