An Approach to the Improvement of Software Development Process with use of Template Generator

IVAN GRBAVAC [1], KREŠIMIR FERTALJ [2], VEDRAN BATOŠ [1]
[1] Department of Electrical Engineering and Computing
University of Dubrovnik
Ćira Carića 4, Dubrovnik 20000
CROATIA
[2] Department of Applied Computing
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, Zagreb 10000
CROATIA
ivan.grbavac@unidu.hr, kresimir.fertalj@fer.hr, vedran.batos@unidu.hr

Abstract: - There comes a time where developed software becomes obsolete and needs a change. The normal life cycle suggest that the next steps in software development would be planning, building and revising, which is not a problem for well-structured projects with clear hierarchy. But life cycle of small and medium applications, which are still very useful and needed, but were developed prior to wide usage of code generators or were built just without code generators on not so structured way and not led by good programming practice, can pose a challenge from an economic and developers point of view. To reduce time and cost of the life cycle and to simplify the software source code developers' labor is not sufficient. Therefore this paper presents concept of template generator which could be used to discover structures in source code and suggest parameterized source code templates. Produced templates would be used in code generation in order to produce well-structured source code.

Key-Words: - template generator, source code generator, code generation, code maintenance

1 Introduction

This modern information era society imposes great necessity of various software applications which now have an increasingly important impact on human life. To make up that demands enterprises have to make more applications in shorter time frames, but also have to maintain existing software [1]. Because of mentioned reasons, the idea of code generators, programs that helps to write other programs, was born in eighties of past century.

With today's complex code-intensive frameworks, such as Java 2 Enterprise Edition (J2EE), Microsoft's .NET, and Microsoft Foundation Classes (MFC), it's becoming more important that programmers' skills are used to build and/or use source code generators, which assists in building applications and downgrading time and expenses needed for accomplishing projects [2]. Also, in addition to financial benefits, code generators preserve identical code structure and way of coding which makes next cycle in software lifecycle easier.

But building new applications is not the only developers' job. Software companies may also reuse and improve built software which was developed by other companies, or by former employees. If the source code, in specific project, was not well-structured or was not made by the help of code generator, its maintenance becomes time consuming and more human resources are allocated to assist in software lifecycle. The problem becomes larger when the cycle comes to a point without support for a framework in use or programming language. Then it is necessary to retype the entire project into a new development environment.

This paper describes a proposal for improving reuse and lifecycle of software using the template generator. Its task is to analyze the original hand-written code, to discover proper code structure and to make a template that could automatically generate source code and consequently speed up the process of developing and maintaining software.

2 Software Development Process

Traditional software development process consists of analysis, design, implementation, and testing

[3][4]. In case when developers didn't use code generators, application program code produced can be categorized into three classes (Figure 1.) [5] [6]:

- **common class**, common to most of developed applications in software company, consisted only of common libraries;
- **resemble class** similar to common but, made without use of code generators, is handmade for every application and different by writing style depending on developer;
- **different class** that depends completely on application, also handwritten and cannot be automatically generated.

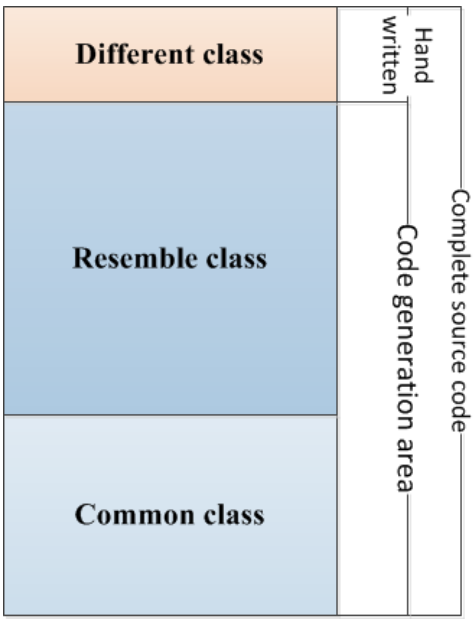

Figure 1. Application Code Classification

During long development process applications may encounter problems, for example, application development can be put aside and subsequently continued but without the developers who started development. In this case, it is necessary first to examine the source code previous employees made, which was not, in many cases, properly structured and which varies depending on the programming style. Of course, it is a time consuming process. A similar problem may occur after the end of the software development and deployment. Then, the software manufacturer has to work on maintenance, further development and improvement of the system, usually in way as indicated in Figure 2. In such continuation of the life cycle it is expected, not only that one developer will replace other, but because of the length of life of the software is expected that several employees works on maintenance of a particular part of code.

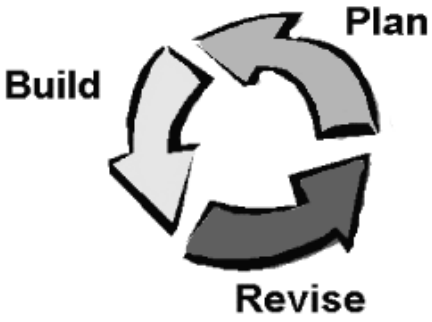

Figure 2. Software lifecycle

2.1 Template Generator Role in Software Development Process

As previously indicated, the software lifecycle and maintenance, if it was not developed with the help of code generators, can be expensive and time-consuming. As shown in Figure 1, a specific part of the code is common or resemble to all applications and modules while a small piece of code varies depending on the application role. These two basic parts of the software, resemble and common class, due to its similarity have the potential to, with the help of text mining processes, detect certain patterns which can be converted into templates for future generation of source code. In this way the module, layer or library could turn into a set of templates that could, with minor modifications via a graphical interface, generate structured code. Those parameterized templates could be used in the further development of the software in a way that they generate new files.

Of course, one part of the code can never be turned into template and parameterized as this code performs a specific function in a particular application. Template generator should indicate the existence of such code and protect it from potential modification [7]. By doing all mentioned, following effects might be achieved:

- **Reduced development time** of new functionality or modules: if patterns, by which old parts of the application were developed, exists there is likelihood that at least some parts of the code could be turned into templates and that new source code could be produced automatically.
- **Reduced number of errors:** templates, generated from existing code, are used during new development and the possibility of errors is minimized. If template contains an error, it can be easily fixed.

ISBN: 978-960-474-344-5

- **Easier maintenance and better performance**: in case that existing modules need to be amended, it is enough to change the template or parameters and again generate modules, same as in conventional template code generation.
- **Code reuse:** if software company works on similar projects templates used in one project might be used in others. So the whole cycle of analysis, design, implementation and testing is being shortened once again.

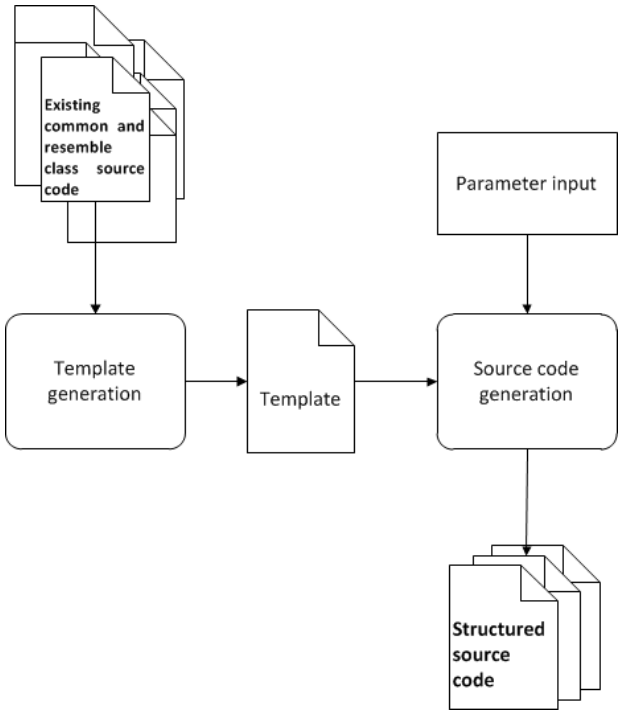

Figure 3. Code maintenance with template generator

3 Templates generator model

3.1 The initial model

Development of suggested model started with simple problem: creating model for converting one source code file into template [8]. During a research of similar works [9][10][11]. Simple model was adopted as shown in Figure 4.

In presented model, source code file is parameterized, and parameters from it are extracted. Initial parameter values are put in XML format and saved in the database, while parameterized source code goes for further processing. Parameterization is done by identifying names and types of variables, names of objects and methods, and replacing them with keywords with references to default values. Problems that should be resolved in this step are related to the language dependent syntax. Also parameterization must be able to recognize the loop in order to mark it for the next step.

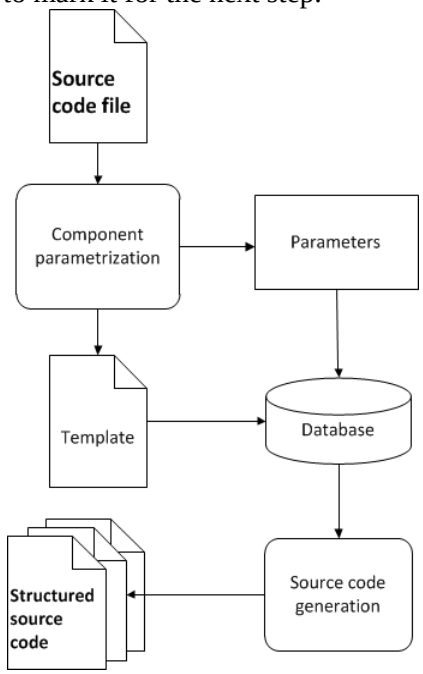

Figure 4. Generation process - one source file[8]

During the following process, parameterized code would be turned into XSL template that could be saved in the database and would later (with suggested default values) be used as input to the source code generator. For code generation process, the best is to use existing generator with ability to be adopted to own needs. In this case, open source generators, as MyGeneration [12], are the best choice.

3.1 The advanced model

The previous section describes the concept of a simple template generator that requires of developer to take a part of code, or one particular file that will serve as input to our program and be turned into template. This process speeds up the development of software and does not require of the developer any specific knowledge about template coding, parameterization and metalanguages. However, developers work could be reduced even more.

The next step in the development of the concept of template generator is development of a model that would accept and analyze multiple files as shown in Figure 5. Developer would, in that case, give as template generator input a number of files from the project. Those files should have a similar program code. The program would then, as in the initial concept, perform parameterization of source

code. Parameterization would be followed by text mining process, which would also include algorithms for preservation of original source code of different class and definitions for specific programing language.

The process of text mining would take only parts of the code which can be converted into one or more templates, while other parts of code would be unchanged and their existence indicated to developer. Selected parts of code should then be turned into a template and saved in the database for further use.

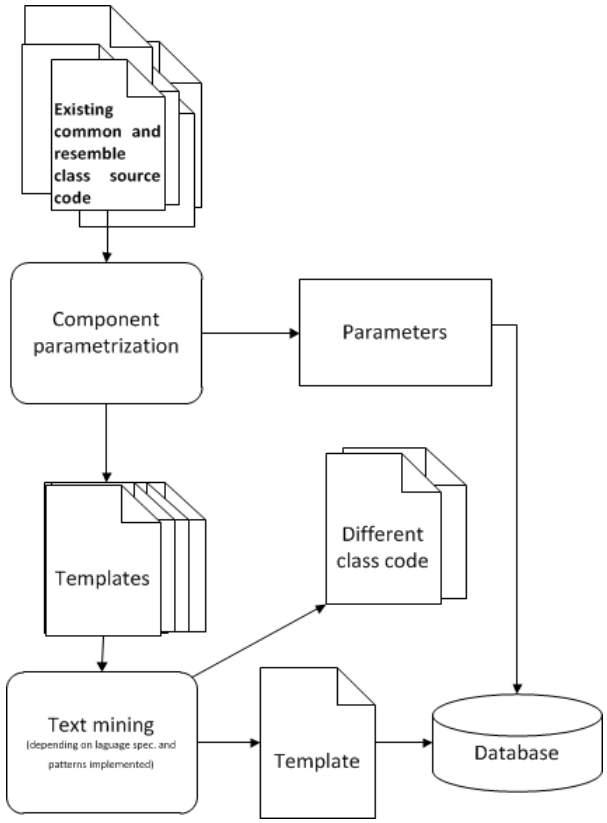

Figure 5. Template generator model

4 Conclusion

Maintaining the existing program code, or continuation of uncompleted software projects can be very difficult and demanding job. To facilitate the work of developers and help them to be able to focus on more important things such as development of new functionality and a new code, rather than time-consuming studying someone else's code, its structuring and refinement, this paper proposes a model of template generator. Its purpose is, through text mining process, to find similar parts of code and turn them into one or more templates which will be stored for later development or code maintenance. Resulting templates can also be used in other similar projects.

Continuation of research activities on suggested model would include the development of prototype which, for the specific language and pattern database, could provide described results.

References:

[1] S. McConnell, Rapid Development. Redmond: Microsoft Press, 1996

[2] J. Herrington, Code Generation in Action, Manning Publications, 2003

[3] R.B. Grady, Successful software process improvement, Prentice-Hall,1997.

[4] B.W. Boehm, Software Cost Estimation with COCOMOll, Prentice Hall, 2000.

[5] M. Yoshida, N. Iwane, An Approach to the Software Product Line System for Web Applications, IEEE International Conference on Computing & Informatics, 2006, Kuala Lumpur, Malaysia

[6] M. Yoshida, N. Iwane, Towards the Software Life Cycle Cost for Integrated Software Product Line Systems, IEEE International Conference on Industrial Informatics, 2006, Singapore

[7] K. Fertalj, D. Kalpic Preservation of Manually Written Source Code in Case of Repeated Code Generation, Proceedings of the IASTED International Conference on Computer Science and Technology, 2003, Cancun, Mexico

[8] T.Helman, K.Fertalj, Application Generator Based on Parameterized Templates Proceedings of the International Conference on Information Technology Interfaces - ITI 2004, Cavtat, Croatia

[9] M.C. Franky, J.A. Pavlich-Mariscal, Improving Implementation of Code Generators: A Regular-Expression Approach, Proceedings of the XXXVIII Latin America Conference on Informatics (CLEI 2012), 2012, Medellín, Colombia

[10] I. Liem, Y. Nugroho, An Application Generator Framelet, Proceedings of the Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008, Phuket, Thailand

[11] K. Fertalj, D. Kalpić, V. Mornar, Source Code Generator Based on a Proprietary Specification Language, Proceedings of the 35th Hawaii International Conference on System Sciences, 2002, Hawaii,USA

[12] M. Griffin, J.Greenwood, MyGeneration Code Generator, www.mygenerationsoftware.com, [19/09/2013]