International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:12, 2015

From Modeling of Data Structures towards Automatic Programs Generating

Valentin P. Velikov

***Abstract*—**Automatic program generation saves time, human resources, and allows receiving syntactically clear and logically correct modules. The 4-th generation programming languages are related to drawing the data and the processes of the subject area, as well as, to obtain a frame of the respective information system. The application can be separated in interface and business logic. That means, for an interactive generation of the needed system to be used an already existing toolkit or to be created a new one.

***Keywords*—**Computer science, graphical user interface, user dialog interface, dialog frames, data modeling, subject area modeling.

I. Introduction

AUTOMATED software generation is a popular problem in the sphere of software industry, associated with automating work in order to save time, brain work and resources [3]. Generating programs allows creating syntactically clean and logically correct modules, starting with designing and modeling the subject area up to obtain the final product. Another target of the automatic generation is to drop out of the need gathering an expert team to update the existing software. This implies a development of a wide class of systems for automated software creation or separated modules for: modeling of a subject area; describing the requirements for the design software; creating documentation for specific purposes. Software generation can be reviewed in various aspects. There are different specifications, one of which is based on the functional status:

1) interface in order to connect with the user – related to the automatic interface generation;
2) functional part – related to the automatic generation of business logic, i.e. – application logic.

The fourth generation programming language implies the usage of graphic or natural-language interface with helps to describe the task. Usually it is being done by a specialist in the subject area rather than a software specialist. This is the basis on which the respective application software is being generated.

II. Detailed Description

There have been created multiple toolkits [3], which tend to develop the automated software creation in different stages. Some of them assist only in the process of designing; others come to finalize the process by generating the appropriate and its supplying documentation. In the group of partial automation software it is possible to found free or open source software, while in the group of comprehensive automation software (CASE-systems) the representatives are only a few, and the price is extremely high. That offers many companies a niche market in providing their own tools and frames with different features and at with different prices. They help the automate creation and production of the whole software or of its individual stages separately.

One possible idea for a system, that helps the software creation, is shown in Fig. 1:

1) The first stage is a description / modeling of the subject area. This can be accomplished in various methods [2], but it is preferred to be done by graphic editors such as BPMN, UML, etc. [1]. They should be very intuitive and easy to use because their users are very often good specialists in a respective subject area rather than software ones. The result of this subsystem is a well described subject area. It is possible to be defined static object as well as some actions between them.

The described subject area can be stored into a common for the whole system pre-defined internal machine representation, or to be transferred for further processing by some of the conventional free text formats, e.g.: XML-file.

2) The graphically described subject area is presented to a Java-code generator. On the basis of the graphic description are generated Java-classes with corresponding to them fields, as well as methods that need to be processed.

The final result is a defined structure of a database, along with functions that need to be processed. Many of the basic functions are known in advance:

1) methods for accessing the classes' fields (get- and set-methods);
2) subprograms that can realize basic manipulations in a database (add, copy, delete, search by key, sorting, etc.).

The tasks are being easily determined and generated this way.

One more functionality can be added to be already defined ones - generation of documentation for the product and for the subprograms. That prepared module is the first step of a CASE-system for automated software creation. The ultimate goal is such a class system is usually generation of elements for three-layer client-server applications: generation of database; generation of the client part (which usually works as a Java-applet in a browser); generation, related to the business rules in the middle layer. This system can be used as a separate application that corresponds with the following subsystems or the external ones trough an appropriate internal

V. Velikov is with the University of Ruse, Bulgaria, dept. of Informatics and Information Technologies (phone: ++359 886 011 544; e-mail: vvelikov@ami.uni-ruse.bg; home page: www.valveliko.com).

machine representation of the subject area or through XML-files.

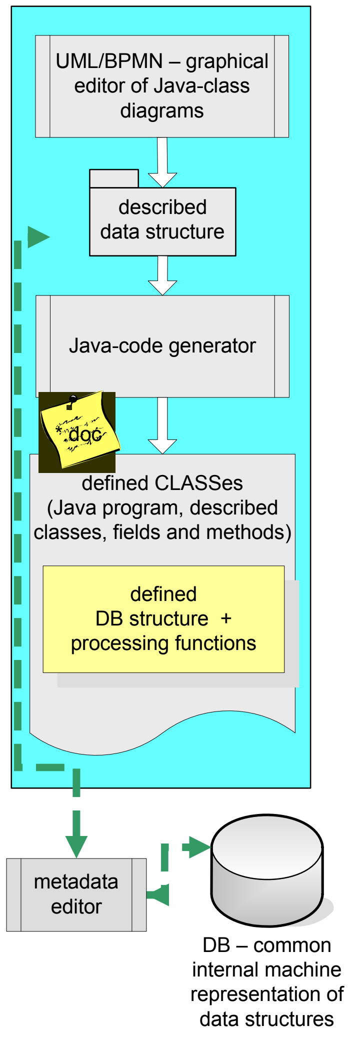

UML/BPMN – graphical editor of Java-class diagrams
described data structure
Java-code generator
*.doc
defined CLASSes (Java program, described classes, fields and methods)
defined DB structure + processing functions
metadata editor
DB – common internal machine representation of data structures

Fig. 1 Unit for Data structure + software generation

CASE systems, code generators [3] and development methods have been analyzed for the design of this module, and a market niche for realizing an application and documentation generator has been sought after. This defines the purpose of the study - to be established a methodology which will develop an application generator in different subject areas, as well as a prototype of this generator.

One of the problems in designing consists in choosing the methodology by which the project would be realized (incl. the internal machine representation of the selected structures), as well as choosing the appropriate technical primitives – for example - programming language. In order to develop the general project as a program language for the creation of the subsystem is selected Java.

A. A Graphical Editor

The first module in this subsystem is the created Graphical editor for creation, editing, and visualization of diagrams of Java classes, which should be developed to a full-featured UML/BPMN graphical editor. During its creation, the following things are required:

1) In concern with the interface and the environment (Fig. 2):
- The process of creating and editing the different elements, that constitute the class diagram, should be accomplished in a graphical environment;
- The environment visually illustrates the set of data, elements and prepared diagrams; it should allow an intuitive and efficient work with them;
- The interface has to provide the necessary tools for working with the program; they have to be accessible and intuitive to use;
- The presentation of the diagrams should follow the standards of UML 2.0. Classes should able to be displayed together with their attributes and methods, as well as the links between them;

2) Requirements in concern with the system functionality:
- To be realized: drawing, processing (by entering or deleting of data) and moving the different elements of the classes' diagrams;
- Saving the project should be possible, as well as loading a project from a stored file;
- Selecting the options for working with the program should be done through the selection menu, using a mouse;
- Entering and erasing text data (names of attributes, methods, etc.) to be done using the keyboard;

3) Requirements in concern with the errors management:
- Errors to be processed during saving and loading a project;
- To be processed the errors by incorrect data input, invalid element position, etc.

When creating this subsystem it should be well understood that it is not designed for a wide range of users, but a small circle of experts who are familiar with the subject area.

The architecture and the implementation of this module are being detailed described in [6].

B. Java-Code Generator

The task that this subsystem fulfills is to generate a programming Java code from the available graphic image (a model of the corresponding subject area, created by the UML-editor), that corresponds to the description.

To generate program code in Java through an appropriate graphical user interface (such as in Fig. 3) in each class there must be filled in the necessary information, which can be done by a junior programmer or designer.

Since the created data structures can be displayed in a variety of inheritable relations and relations between them, therefore the dialog window in Fig. 3 in which they are being displayed has a different or changing content during runtime.

Since the number of classes (Fig. 3) is unknown in advance (unlimited, too), the most appropriate thing for their storage is to be used some kind of a dynamic linear structure (in this case - arrayList).

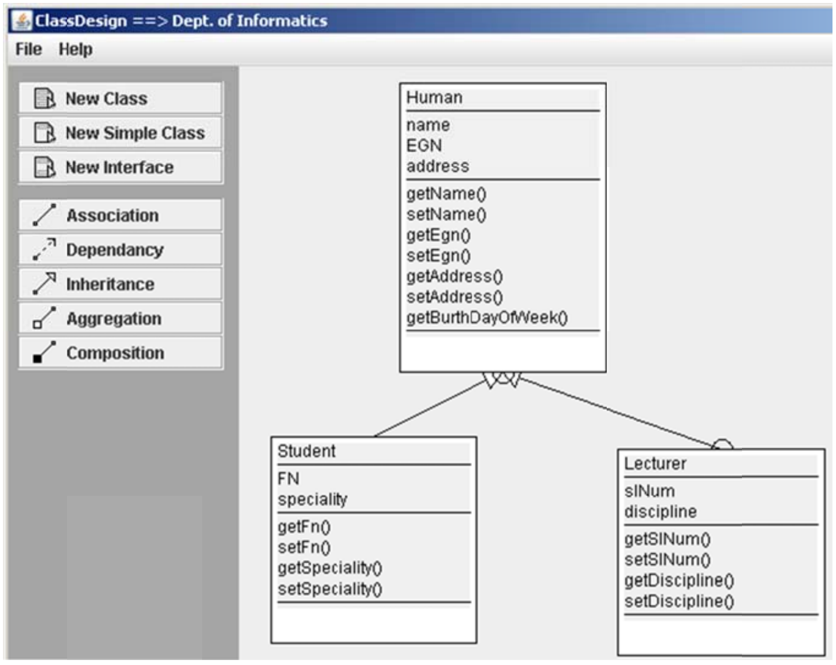
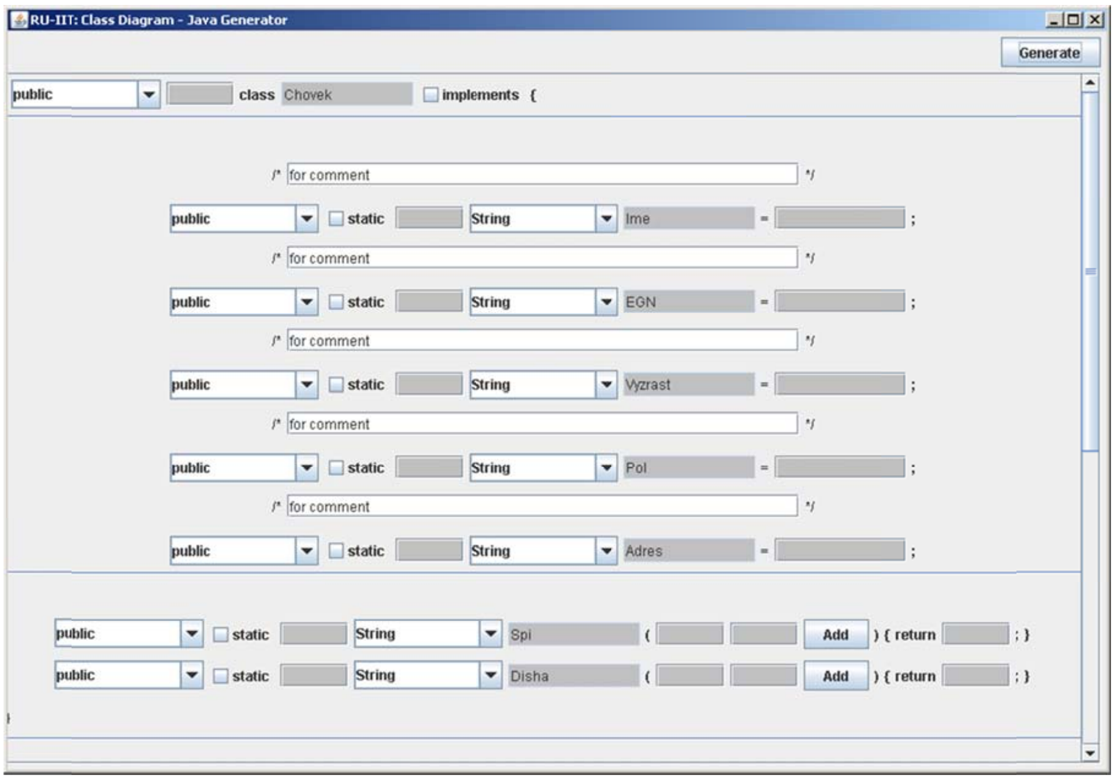


Fig. 2 The Graphical editor



public class Chovek implements {

/* for comment */
public static String Ime = ;
/* for comment */
public static String EGN = ;
/* for comment */
public static String Vyzrast = ;
/* for comment */
public static String Pol = ;
/* for comment */
public static String Adres = ;

public static String Spi ( ) { return ; }
public static String Disha ( ) { return ; }

Fig. 3 Java-code generator

If it is known in advance the kind and the type of the database (e.g. MySQL) and if an information about the metadata is available, then the type of the fields and many of the modifiers could be taken automatically from there. When modeling with the UML- editor, the needed information for the names of the fields and the classes could be taken from the metadata. If the type of database is already known (i.e. – the language for accessing and manipulating the data is known) there could automatically be generated a code for creating the basic manipulating functionalities that have influence over the database: create, delete, update. It is appropriate in such a case, while being in graphic mode (UML- editor) to be indicated the manipulated fields and/or the basic actions. It may also be appropriate for the UML-editor to be functionally supplemented by some properties for QBE (Query By Example).

If the process is not supervised by a human specialist, the subsystem will be generating empty methods with modifiers by default. Those values can be changed in the settings of the environment.

The architecture and the implementation of this module are described in detail in [4].

C. Generator of Documentation

It is possible to be performed in various ways and to cover in different levels the created application.

A lot of software companies recommend writing a self-documented code, i.e. - using generally available recommendations (conventions) in the dispensation of subprogram names, variables, structures and so on. Since many national or private company standards include some additional requirements towards the documentation of the created software, it is often needed an additional description of the code that is being generated. Using some tools would support the process.

The Java language itself offers some additional opportunities for documenting the code, which are cleverly used in the most common environment (Eclipse) for this programming language, as well as the Javadoc tool. This opens up a field for an additional or a new development, when using the existing tools.

D. DB – A Common Internal Machine Representation of Data Structures

With the graphical editor is described a subject area: classes, relationships, actions. The result should be stored on permanent media. And there are various possible options. In order for maximal speed - some of optimal is to use frames [7] with pre-defined internal machine representation of the subject area. The disadvantage of this approach: it is possible to process these frames by that system only, or from someone who has a frame structure.

E. Metadata Editor

When defining a group of frames to describe a subject area, they can be stored in a particular database. If describing another subject area there can be defined another group of frames and so on.

Working with a database of frames requires an editor that allows all the basic operations with the database to be executed: examination of the frames in the subject area, addition of new ones, editing of existing ones, deleting, and some other specific manipulations.

F. Converter from Java-Code to XML-File and vice versa

One of the ideas is for an existing program (Java-class, method, library, interface) - to be modified by an external application (on an external developer, for intermediate processing), aiming to add a new functionality that currently does not exist into the newly created system. That means Java-code (or UML, or another diagram) to be transferred to an external module, to be processed and returned. This purpose requires a common format for presentation of the code, data, charts, etc. The best in the case is using a text format or its variation, e.g. XML-format. This raises the need for creation of a converter that transforms Java-code to XML-file. The so existing Java application can be exported via XML to another system, which would modify the code (further processing, update of existing programs or generating new sub-programs), and the merged code (in XML-format) would be imported back again.

It is possible in a particular moment for a system to be part of another system (with different developers), and to be used in preconditioning of a code. In this case, the code should be transferred (import/export) to another system. For this purpose, there are needed some converters from Java-code (Java class) to XML-format and vice versa.

The main goal is to create a Java application, which can convert a Java-class (program) into XML text file. Well formatted, the output file can be easily read by people outside the programming field. Presented in such a way, the task is to be created an application which can convert not only Java-classes, but also the basic Java language constructs such as: a class, a package, import, constructors, methods, method calls, variable values, variable worthless, conditional statements such as- if, if-else, for, while, single- and multi-line comments. With the development of this subsystem, other language constructs will be added.

The architecture and the implementation of this module are described in detail in [5].

III. Conclusion

As an application to a developed methodology, it was created a subsystem for an automatic software generation (Fig. 1). In the beginning, a specialist in the respective subject area (not an IT-specialist) describes the task, using a graphical (UML) editor. On the next stage, an IT-specialist completes the description, assisting the software generation and the supporting documentation. To complete functionality, if necessary, it can be used external modules – the communication with them is performed in an XML-format. The frames that describe a subject area can be stored in a common database for internal machine representation of the used structures.

Acknowledgment

This work is supported by the National Scientific Research Fund under the contract ДФНИ - И02/13.

In the course of this study has been used equipment purchased under the project BG161PO003-1.2.04.-0011-

S0001 "Development of applied research at the University of Ruse" implemented with the financial support of OP "Development of the Competitiveness of the Bulgarian economy 2007-2013 ", co-financed by the European Union through the European Regional Development Fund and the state budget of the Republic of Bulgaria."

REFERENCES

[1] Hristova, P – Studying UML in The “Master degree” in Informatics education level, Proceedings, vol.47, book 5.1, Mathematics, Informatics and Physics, Ruse, 2008

[2] Stanev I. Generation of Computer Programs for Robot Control Specified through Limited Natural Language Texts. In Elektrotechnica & Electronica. Vol. 5/6 1999г. Pp. 18 – 23.

[3] Velikov V., I. Kamenarov, Software development aid systems, ITHET 2014,UK, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7155668&filter%3DAND%28p_IS_Number%3A7155664%29

[4] Velikov V., K. Dobreva. Generator from Java class-diagrams into Java source code, III International Scientific-Practical Conference “Information Systems and Technology: Management and Security”, December 2014, Тольятти, ISBN 978-5-9581-0340-9

[5] Velikov V., M. Makarieva. Parser Java-code to XML-file.// Proceedings of the Union of Scientists - Ruse, Book 5 - Mathematics, Informatics and Physics, 2014, No Vol. 11, pp. 72 - 79, ISSN 1314-3077.

[6] Velikov V., K. Grigorova, A. Iliev. Graphical editor for creation and editing Java-class diagrams. In Proceeding of University of Ruse, 2014, vol. 53, book 6.1, p.121-126. (In Bulgarian - Великов, В., К. Григорова, А. Илиев. Графичен редактор за създаване и редактиране на Java клас-диаграми. В: Научни трудове на РУ & СУ, Русе, т.6. 2014)

[7] Minsky M., A Framework for Representing Knowledge. 1975, NY, McGraw Hill, pp.211-277. (in Russian: Минский. М. Фреймы для представлении знании. “Энергия”, Москва, 1979. http://www.raai.org/library/library.shtml?extbooks 2014 г.)

Valentin P. Velikov, PhD was born in Straklevo, Ruse, Bulgaria on 21 march 1962. In 1980 was graduated from the multidisciplinary high school “Hristo Botev” in Ruse, Bulgaria. In 1987 he took his diploma as an engineer in the fields of computer engineering (software profile) from The University of Ruse. In 2014 he took a Philosophy Degree in Informatics (Computer science) from the same university.

1980-1982 - Compulsory military service in Bulgaria. In 1987 he obtained a position as an assistant professor of Informatics (computer science). In 1999 he was promoted to the position of a senior assistant (senior lecturer) in Dept. of Informatics. He is lecturer in C/C++, Java SE/EE/ME, Android etc.

Dr. Velikov is a member of the Union of Scientist in Bulgaria and in the Union of Mathematicians in Bulgaria (branch Ruse).