Dynamic Frames-Based Generation of Web 2.0 Applications

Tihomir Orehovački
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42 000 Varaždin, Croatia
tihomir.orehovacki@foi.hr

Ivan Magdalenić
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42 000 Varaždin, Croatia
ivan.magdalenic@foi.hr

Danijel Radošević
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42 000 Varaždin, Croatia
danijel.radosevic@foi.hr

***Abstract*—Frame Technology (FT) and Generative Programming (GP) are two widely accepted paradigms of software product lines development. While GP addresses the automatic generation of source code, FT advocates its adaptation to diverse reuse contexts. With an aim to utilize benefits of both approaches, this paper presents the SCT dynamic frames model that supports the automatic generation of Web 2.0 applications. The SCT model encompasses three essential components: Specification (S), which refers to application features, Configuration (C), which describes application development rules, and Template (T), which denotes application building blocks. Owing to its flexibility, readability, interactivity, and other object-oriented features, the Python scripting language was selected for the implementation of the generator. In order to demonstrate the appropriateness and usefulness of the proposed approach, an example that illustrates the generation of a Web 2.0 application for database management is provided.**

Keywords—Web 2.0 Applications; Dynamic Frames; Generative Programming

I. INTRODUCTION

The term Web 2.0 refers to a second generation of web applications which enable users to interact with functionalities of their interfaces in a desktop-like fashion. Being dynamic in nature, Web 2.0 applications encourage users to create, share, publish, organize, and integrate a variety of artefacts thus contributing to the development of knowledge repositories. Given that Web 2.0 applications provide support for asynchronous and synchronous communication among users as well as collaboration on artefacts, they are commonly referred to as social web applications. According to Orehovački et al. [13], the most popular representatives of Web 2.0 applications are wikis, blogs, microblogs, social bookmarking sites, social networking sites, mashups, podcasting applications, e-portfolios, virtual worlds, online office suites, and knowledge management applications. Considering that evaluation presents indispensable part of every development process, recent research effort was focused on modelling their adoption [15], classification of quality in use metrics [42], measuring quality of collaborative editors [19][20], evaluating the quality in use of mind mapping [17][18][20] and diagramming services [17] by means of both objective and subjective instruments, assessment of mashup tools [16], as well as evaluation of artefacts [21] which represent an outcome of their use.

From technical perspective, Web 2.0 applications are flexible services implemented in client-side Asynchronous JavaScript and XML (AJAX) frameworks. On the server side, scripting languages such as PHP, Perl, Python, Ruby, and JSP are used for delivering content from files and databases to the client. Despite the fact that Web 2.0 applications are widely used for both private and business purposes, there is a lack of comprehensive models and methodologies for their systematic development. Namely, the majority of current approaches deals with the model driven interface design (e.g. [11][12]), development (e.g. [3][10]), and code generation (e.g. [1][4]) of Rich Internet Applications (RIAs).

Considering the complementariness of different software development paradigms, a number of authors (e.g. [23], [30]) have merged two or more approaches into one thus yielding significant synergy effects. With an objective to achieve similar results in the context of Web 2.0 applications, we integrated

concepts of frame technology (FT) and generative software development (GSD). Frame technology is a textual pre-processor which consists of two essential components: hierarchically organized code templates (frames), and a specification which contains particular features that can be adapted to different contexts [5]. On the other hand, generative software development supports mapping between a set of the features described by a domain specific language (DSL), and implementation components with all their possible combinations [2]. The aim of this paper is to illustrate appropriateness and usefulness of the use of SCT dynamic frames [7] in the generation of Web 2.0 applications.

The remainder of the paper is organized as follows. Overview of current research is provided in the second section. Features of the SCT generator model and generator design steps in the context of Web 2.0 applications are explained in the third section. An example how SCT generator can be employed for the purpose of developing Web 2.0 applications is illustrated in the fourth section. Concluding remarks and future research directions are offered in the last section.

II. Background to Research

The purpose of this section is to provide a brief review of two software development paradigms which constitute the theoretical background to the dynamic frames-based generation of Web 2.0 applications.

Software product line (SPL) denotes a group of software products that have a common set of features which meet stakeholders' needs [14]. Drawing on frame technology (FT), frame based software development (FBSD) is focused on design of generalized and adapted components. FT refers to a language independent textual pre-processor whose aim is development of systems which can be easily modified and consequently reused in a variety of contexts [5]. There are two essential elements which constitute frame technology: code templates structured in a hierarchy of modules known as frames and a specification that consist of particular features specified by the developer. In the context of software engineering, the aforementioned infrastructure represents a sound architecture for deriving SPLs [24]. Grossman and Mah [22] found that the employment of FT results in a decrease of expenses and time to market for large software development projects while in the same time contributes to the increase of reuse levels. These productivity enhancements motivated Jarzabek and Zhang [26] to introduce the meta-programming technique called XML-based Variant Configuration Language (XVCL) that drawing on Basset's frames [25] facilitates management of variability in SPLs. XVCL supports the decomposition of programs into generic and adaptable meta-components known as x-frames which as XML files represent domain knowledge in the form of SPL artefacts. An x-framework is a normalized layered hierarchical structure composed of x-frames that allows handling variants at different granularity levels. A configuration of a particular SPL member is managed by the topmost x-frame which is called the specification frame (SPC). Starting with the SPC call, the XVCL processor goes through an x-framework, interprets XML tags in visited x-frames and by conducting necessary adaptations assembles components of specific SPL members. Taking into account advantages of XVCL with the respect to the reusability improvement, its concepts have been thoroughly evaluated in the context of databases [27], fault tolerant architectures [28], computer aided dispatch domain [29], etc.

The central role in generative software development (GSD) plays domain model which deals with mapping between problem space and solution space [2]. Problem space denotes a set of features of a SPL member that are described by means of the DSL. Implementation-based abstractions that constitute the specification of a SPL member are referred to as solution space. The mapping between the set forth spaces is carried out with the use of generator which calls a specification and eventually result in a corresponding implementation. Apart from XVCL, there are some other techniques that are also used for the purpose of generating software artefacts. One of them is GenVoca [31], a composition methodology meant for generating hierarchical SPL families. Fundamental features related to GenVoca are virtual machines, layers, realms, type equations, and a grammar. Virtual machines represent a set of methods, classes, and their objects that are employed for the implementation of SPL functionalities. An implementation of particular virtual machine is called layer. Realm is a set of layers that implement the same virtual machine. Each layer imports interface of the realm whose parameters it contains and exports the virtual machine of the realm it belongs to. Layer that imports and exports the same virtual machine is labeled as symmetric layer. The objective of layers is to encapsulate transformation that maps objects and operations between virtual machines. The structure composed of layers that are employed for modeling a particular software system is called a type equation. Realms together with their layer specify a grammar in which particular SPL member has a role of a sentence.

Current research related to the practical use of generators can be classified into several groups. The first group is focused on generating code snippets in a variety of programming languages that range from Python [39] and Java [6][8] to PHP [9]. The aim of the second group of generators is design of non-code artefacts such as graphical interface [38], programming assignments [36], text [37], and 3D scenes [41]. The last group is composed of generators which are implemented in scripting programming languages such as Open PROMOL [34] and CodeWorker [35]. While Open PROMOL deals with specifying program modifications of a target language, CodeWorker is meant for both parsing of arbitrary grammars and source code generation. The generator presented in this paper adds to the extant body of knowledge which deals with generation of code artefacts. Details on features of generation architecture that was employed for that purpose in the context of Web 2.0 applications are provided in the following section.

III. Generating the Web 2.0 Application

The SCT generator model is based on dynamic frames [7] and can be used in the generation of a wide variety of applications. The SCT generator model defines the generator of source code from three core elements: Specification (S), Configuration (C) and Templates (T). Specification contains features of the generated application in the form of attribute-value pairs. Templates contain source code in a target programming language together with connections (replacing marks for the insertion of variable code parts). Configuration

defines the connection rules between Specification and Templates. All three model elements together constitute the SCT frame.

A particular SCT frame produces source code that could be either stored in a specific data file or included in another SCT frame. The basic idea of the generation process is shown in Figure 1. The initial SCT frame contains the initial source code template that includes connections. The generator of the source code creates a new SCT frame for each connection. While the source code of SCT frames located deeper in the hierarchy is included as the integral part of its superior SCT frame, the source code of the initial SCT frame is stored in a data file.

Since an average application contains more data files, the SCT model implies the existence of a Handler. It represents a part of the SCT source code generator that aims to make the generator scalable in a way that it can produce more pieces of program code (e.g. program files) from the same set of Specification, Configuration and Templates. The SCT dynamic frames model enables the generation of various program units (e.g. files, classes, functions etc.) from the same Specification. Moreover, it enables the generation of code in a variety of programming languages (e.g. JavaScript, PHP, XML, Python, Java, etc.) and is consequently suitable for the generation of Web 2.0 applications. The generated code can be stored in program files for later execution as well as in variables for immediate execution [32] [32].

Web 2.0 applications are specific since they use different technologies in an integrated manner. The flexibility of the SCT generator enables implementation of several technologies in the same application. This is achieved with cautious design of Specification, Templates and Configuration of the SCT generator.

Model which reflects development process of Web 2.0 application by means of SCT generator is illustrated in Figure 2. The first step is to identify Web 2.0 services and build one or several prototypes for each service. Based on experience obtained during development of prototypes, a set of templates is developed for each service. Those templates are input into SCT generator. Different applications have different set of services which is defined in Application specification. Each application has list of web services and other important data listed in its Application specification. How templates are combined together, based on Application specification, is defined in Application configuration.

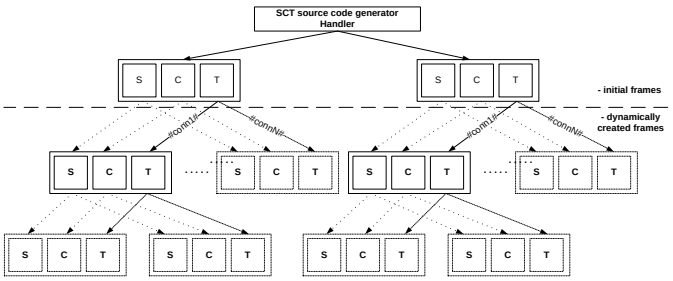

Fig. 1. The generating process

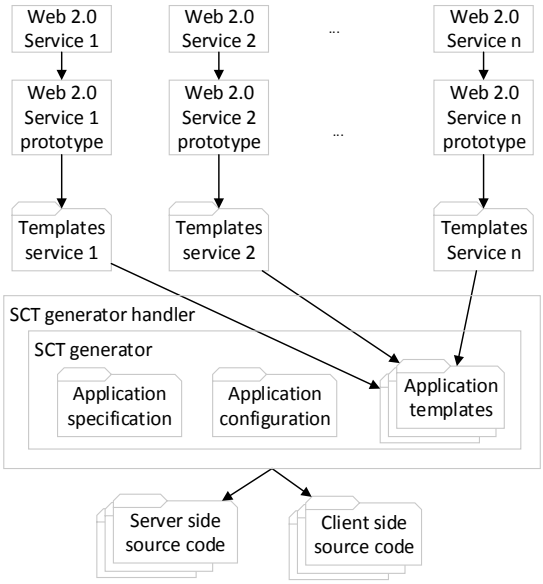

Fig. 2. Model of building Web 2.0 applications using the SCT generator

The SCT Handler generates more data files which contain source code and implement both server and client side of Web 2.0 application. Since Web 2.0 application employs different technologies, it is a challenge to make such templates that can be easily manageable and reusable. In that respect, the SCT generator model offers management of the whole set of code templates via relatively small Configuration.

The process of building new generators begins with application prototype that is decomposed into SCT model elements through several steps. The SCT generator applies these elements in automatic assembling of different application variants. Steps in the design of a generator of Web 2.0 application are as follows [40]:

0. Prerequisite. The prerequisite for building the SCT generator is the application prototype in a form of a source code.

1. Selection of new main templates and output types. The main templates are specified in the initial part of Configuration and define the type of code to be generated, e.g.:

```
#1#,,index.template          - entry HTML page
```

2. Creating of Specification. Specification consists of attributes and their values. The hierarchy of attributes is specified by '+' sign, e.g.:

```
field_combo:id_course
+field_display:Course        - subordinated attribute
```

3. Delineation of variable program parts. Variable program parts depend on Specification, so they will be later replaced with connections.

4. Flexibilization of prototype. Variable program parts are being replaced by connections (in #-es).

5. Adding new rule to Configuration. The configuration rule specifies all three elements of the SCT model: connection, specification attribute and used code template, respectively e.g.:

```
#links#,title,links.template
```

6. Building of code templates that are main constituent artefacts of generated applications.

7. Generating, testing and adjusting in a generative development process.

An outcome of the development process that begins with an application prototype is a generator that can be used in automatic design of different application variants. Obtained SCT model elements (Specification, Configuration and Templates) can be used in the further development of generators as well as applications.

As shown in Figure 3, the development process of particular Web 2.0 application can be illustrated with spiral model that was originally proposed by Boehm [43].

IV. THE EXAMPLE OF GENERATING

The example[1] includes a SCT based generator, implemented in Python, together with generated Web 2.0 application (also in Python; Ajax was used for user interface and PostgreSQL for database implementation).

Specification of the given instance contains three output types:

```
OUTPUT:out1                - used for index page
OUTPUT:output              - CGI scripts (Python)
OUTPUT:output_html         - HTML forms
```

Each output type refers to one or more output files that will be generated. For instance, there are two files that are going to be generated from the following specification group:

```
output:output/students.cgi    - CGI script
output_html:output/students_form.html - HTML form
table:ajax_students           - database table
connection:exams              - link to another DB table
+connection_field:student_id - subordinated attributes
+connection_display:Exams      to link
title:students                - group name
+title_display:Students       - text to be displayed
primary_key:student_id        - DB table primary key
field_number:student_id       - table attribute+type
+field_display:Student id     - text to be displayed
field_text:surname_name
+field_display:Surname and name          . . .
field_number:year_of_enrollment
+field_display:Year of enrollment        . . .
field_number:year_of_study
+field_display:Year of study
```

Configuration contains rules for assembling software from Specification and Templates. The initial part of Configuration specifies the initial code templates that correspond to output types from Specification:

```
#1#,,index.template  - index page
#2#,,script.template - CGI scripts
#3#,,form.template   - HTML forms
```

Other lines of Configuration contain two- or three-element groups, e.g.:

```
#table#,table                       - link, attribute
#title_field#,title,title.template - link, attribute,
                                     template
```

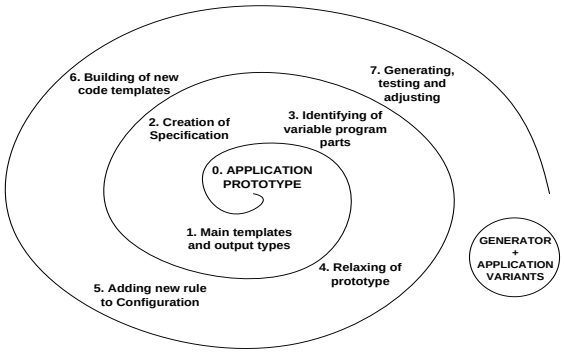

Fig. 3. Spiral generator/application development [40]

The two-element group specifies direct replacement of the position of variable in Templates with value of attribute from Specification. The three-element group specifies that code template has to be used as many times as it occurs in Specification. Each code template employs connections (usually words in '#'-es) in order to specify variable parts that are going to be generated. The following example of a template is used in the generation of input/edit forms:

```
<form id="myForm" action="" method="POST" >
<input type="hidden" name="action" value="!operation!">
        #fields_on_form#
<td><input type='submit' name="Submit"
value='Send'onclick="Perform2('#fields_getelementbyid#'
,'#title#.cgi?action=!operation!','R0')"></td>
<td> </td>
</form>
```

The example utilizes Ajax to route the output of the *CGI* script to a particular *HTML* element marked by *id* (here: 'R0'). As shown in Figure 4, this feature enables editing of particular row in database table, without the need of refreshing the whole web page.

The aim of the SCT generator model is to achieve high reusability of features (attributes with their values) defined in Specification. These features can be distributed through connections on many different places in diverse code templates, as shown in Table 1.

TABLE I. DISTRIBUTION OF SPECIFICATION FEATURES IN THE EXAMPLE APPLICATION

Attribute	Total number of occurrences in Specification	Total number of occurrences in generated code	Number of files where the value occurs
application	1	2	1
table	3	90	3
title	3	52	3
title_display	3	15	3
field_number	6	397	3
field_combo	2	70	1

[1] The example is available at gpml.foi.hr/SCT_Python_Ajax

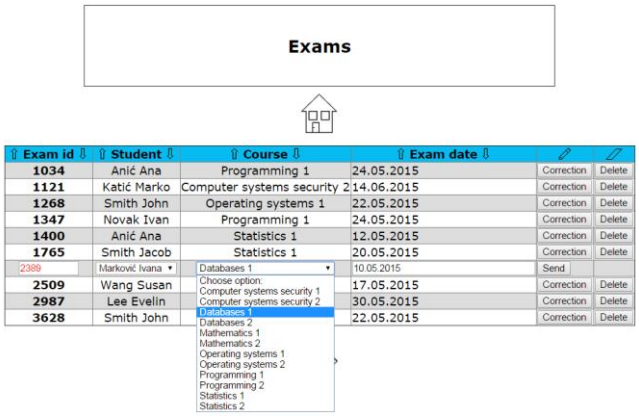

Fig. 4. Editing the particular row in database table

The multi-distribution of specification features could be used in application updating. This can be performed by changing the Specification, which enables new features of applications inside the problem domain proposed by Configuration. Any modification in Templates changes the way Specification attribute values are used, including the programming language. Any update of Configuration changes the way the generator builds the program code, respectively. The introduction of a new line in Configuration could enable the use of a new Specification attribute and a new code template. The purpose of the set forth is to avoid any later modifications of the generated code.

V. Concluding Remarks

This paper illustrated the use of dynamic frames generator model in the development of Web 2.0 applications. There are numerous benefits of the proposed approach. The first one is the improvement of the development process productivity that is the outcome of reusability of program artefacts. The set forth productivity reflects in terms of enhanced efficiency in development of software product lines, as well as facilitated features specification at higher level of abstraction. The second one is simplified application update which results from definition of application in a higher abstraction language used by SCT generator. Inclusion of new features in application is performed by adding new definitions in application specification. The third one is the customization of application to the specific needs of particular user. Considering that network effects, perpetual beta, and lightweight user interfaces, respectively are essential design patterns of Web 2.0 applications, the proposed approach supports the user-centered development of software product line members.

Our future work will be focused on the employment of dynamic frames based generators in the development of some specific types of Web 2.0 applications such as mashups. More specifically, our research efforts will deal with interplay of different generator implementations and novel web technologies.

References

[1] A. Bozzon and S. Comai, “Conceptual Modeling and Code Generation for Rich Internet Applications”, Proceedings of the 6th International Conference on Web Engineering, pp. 13-18, 2006.

[2] K. Czarnecki, “Overview of generative software development”, in Unconventional Programming Paradigms, Lecture Notes in Computer Science, vol. 3566, J.-P. Banâtre, P. Fradet, J.-L. Giavitto and O. Michel, Eds. Le Mont Saint Michel: Springer, 2004, pp. 326-341.

[3] J.M. Hermida, S. Meliá, A. Montoyo and J. Gómez, “Developing Rich Internet Applications as Social Sites on the Semantic Web: A Model-Driven Approach”, International Journal of Systems and Service-Oriented Engineering, vol. 2, no. 4, pp. 21-41, 2011.

[4] M. Linaje, J.C. Preciado, R. Morales-Chaparro, R. Rodríguez-Echeverría and F. Sánchez-Figueroa, “Automatic Generation of RIAs Using RUX-Tool and Webratio”, in Web Engineering, Lecture Notes in Computer Science, vol. 5648, M. Gaedke, M. Grossniklaus and O. Díaz, Eds. San Sebastian: Springer, 2009, pp. 501-504.

[5] N. Loughran, A. Rashid, W. Zhang and S. Jarzabek, “Supporting Product Line Evolution with Framed Aspects”, in Workshop on Aspects, Components and Patterns for Infrastructure Software, 2004, <http://www.comp.lancs.ac.uk/computing/aod/papers/SPL_ACP4IS2004.pdf>.

[6] I. Magdalenić, D. Radošević and Z. Skočir, “Dynamic Generation of Web Services for Data Retrieval Using Ontology”, Informatica, vol. 20, no. 3, pp. 397-416, 2009.

[7] D. Radošević and I. Magdalenić, ”Source Code Generator Based on Dynamic Frames”, Journal of Information and Organizational Sciences, vol. 35, no. 1, pp. 73-91, 2011.

[8] D. Radošević, M. Konecki and T. Orehovački, “Java Applications Development Based on Component and Metacomponent Approach”, Journal of Information and Organizational Sciences, vol. 32, no. 2, pp. 137-147, 2008.

[9] D. Radošević, T. Orehovački and M. Konecki, “PHP Scripts Generator for Remote Database Administration based on C++ Generative Objects”, Proceedings of the 30th MIPRO Jubilee International Convention on Intelligent Systems, pp. 167-172, 2007.

[10] B. Stearn, “XULRunner: A New Approach for Developing Rich Internet Applications”, Internet Computing, vol. 11, no. 3, pp. 67-73, 2007.

[11] M. Urbieta, G. Rossi, J. Ginzburg and D. Schwabe, “Designing the Interface of Rich Internet Applications”, Proceedings of the 5th Latin American Web Conference (LA-WEB), pp. 144-153, 2007.

[12] F. Valverde and O. Pastor, “Applying Interaction Patterns: Towards a Model-Driven Approach for Rich Internet Applications Development”, Proceedings of Workshop on Web-oriented Software Technology (IWWOST), pp. 13-18, 2008.

[13] T. Orehovački, G. Bubaš and A. Kovačić, “Taxonomy of Web 2.0 Applications with Educational Potential”, in Transformation in Teaching: Social Media Strategies in Higher Education, C. Cheal, J. Coughlin abd S. Moore, Eds. Santa Rosa: Informing Science Press, 2012, pp. 43-72.

[14] P. Clements and L. Northrop, Software product lines: Practices and patterns. Boston: Addison-Wesley, 2002.

[15] T. Orehovački and S. Babić, “Predicting Students' Continuance Intention Related to the Use of Collaborative Web 2.0 Applications”, Proceedings of the 23rd International Conference on Information Systems Development (ISD), pp. 112-122, 2014.

[16] T. Orehovački and T. Granollers, “Subjective and Objective Assessment of Mashup Tools”, in Design, User Experience, and Usability - Theories, Methods, and Tools for Designing the User Experience, Lecture Notes in Computer Science, vol. 8517, A. Marcus, Ed. Heraklion: Springer, 2014, pp. 340-351.

[17] T. Orehovački, A. Granić and D. Kermek, “Evaluating the Perceived and Estimated Quality in Use of Web 2.0 Applications”, Journal of Systems and Software, vol. 86, no. 12, pp. 3039-3059, 2013.

[18] T. Orehovački, A. Granić and D. Kermek, “Exploring the Quality in Use of Web 2.0 Applications: The Case of Mind Mapping Services”, in Current Trends in Web Engineering, Lecture Notes in Computer Science, vol. 7059, A. Harth and N. Koch, Eds. Paphos: Springer, 2011, pp. 266-277.

[19] T. Orehovački, "Perceived Quality of Cloud Based Applications for Collaborative Writing", in Information Systems Development – Business Systems and Services: Modeling and Development, J. Pokorny, V. Repa, K. Richta, W. Wojtkowski, H. Linger, C. Barry and M. Lang, Eds. Prague: Springer, 2010, pp. 575-586.

[20] T. Orehovački, S. Babić and M. Jadrić, "Exploring the Validity of an Instrument to Measure the Perceived Quality in Use of Web 2.0 Applications with Educational Potential", in Learning and Collaboration Technologies - Designing and Developing Novel Learning Experiences, Lecture Notes in Computer Science, vol. 8523, P. Zaphiris, and A. Ioannou, Eds. Heraklion: Springer, 2014, pp. 192-203.

[21] T. Orehovački and N. Žajdela Hrustek, "Development and Validation of an Instrument to Measure the Usability of Educational Artifacts Created with Web 2.0 Applications", in Design, User Experience, and Usability - Design Philosophy, Methods, and Tools, Lecture Notes in Computer Science, vol. 8012, A. Marcus, Ed. Las Vegas: Springer, 2013, pp. 369-378.

[22] I. Grossman and M. Mah, "Independent research study of software reuse using frame technology", Technical Report, QSM Associates, 1994.

[23] L. Fuentes, C. Nebrera and P. Sánchez, "Feature-oriented model-driven software product lines: the TENTE approach", Proceedings of the forum of the 21st international conference on advanced information systems (CAiSE), pp. 67-72, 2009.

[24] P.G. Bassett, "The case for frame-based software engineering", IEEE Software, vol. 24, no. 4, pp. 90-99, 2007.

[25] P.G. Bassett and E. Yourdon, Framing software reuse – lessons from real world. Upper Saddle River: Prentice Hall, 1997.

[26] S. Jarzabek and H. Zhang, "XML-based method and tool for handling variant requirements in domain models", Proceedings of the 5th IEEE international symposium on requirements engineering, pp. 166-173, 2001.

[27] S. Guo, L. Tang and W. Xu, "XVCL – an annotative approach to feature-oriented programming", Proceedings of the 2010 international conference on computational intelligence and software engineering, pp. 1-5, 2010.

[28] L. Yuan, J. Song Dong and J. Sun, "Modeling and customization of fault tolerant architecture using object-Z/XVCL", Proceedings of the 13th Asia Pacific software engineering conference, pp. 209-216, 2006.

[29] H. Zhang and S. Jarzabek, "XVCL: a mechanism for handling variants in software product lines", Science of Computer Programming, vol. 53, no. 3, pp. 381–407, 2004.

[30] I. Groher and M. Voelter, "Aspect-oriented model-driven software product line engineering", in Transactions on Aspect-Oriented Software Development VI, Lecture Notes in Computer Science, vol. 5560, S. Katz, H. Ossher, R. France and J-M. Jézéquel, Eds. Heidelberg: Springer, 2009, pp. 111-152.

[31] D. Batory, V. Singhal, J. Thomas, S. Dasari, B. Geraci and M. Sirkin, "The GenVoca model of software-system generators", IEEE Software, vol. 11, no. 5, pp. 89-94, 1994.

[32] R. Fabac, D. Radošević and I. Magdalenić, "Autogenerator-Based Modelling Framework for Development of Strategic Games Simulations: Rational Pigs Game Extended", The Scientific World Journal, 2014, <dx.doi.org/10.1155/2014/158679>.

[33] I. Magdalenić, D. Radošević and T. Orehovački, "Autogenerator: Generation and Execution of Programming Code on Demand", Expert Systems with Applications, vol. 40, no. 8, pp. 2845-2857, 2013.

[34] V. Štuikys and R. Damaševičius, "Scripting language open PROMOL and its processor", Informatica, vol. 11, no. 1, 71–86, 2000.

[35] C. Lemaire, "CodeWorker parsing tool and code generator – user's guide & reference manual, release 4.5.4.", 2010, <http://www.codeworker.org/CodeWorker.pdf>.

[36] D. Radošević, T. Orehovački and Z. Stapić, "Automatic on-line generation of student's exercises in teaching programming", Proceedings of the 21st Central European conference on information and intelligent systems, pp. 87-93, 2010.

[37] J. Müller and U.W. Eisenecker, "The applicability of common generative techniques for textual non-code artifact generation. In Proceedings of the workshop on modularization, composition, and generative techniques for product line engineering, 2008, <http://www.infosun.fim.uni-passau.de/spl/apel/McGPLE2008/papers/Paper8.pdf>.

[38] M. Schlee and J. Vanderdonckt, "Generative programming of graphical user interfaces", Proceedings of the working conference on advanced visual interfaces, pp. 403-406, 2004.

[39] D. Radošević and I. Magdalenić, "Python implementation of source code generator based on dynamic frames", Proceedings of the 34th International Convention on Information and Communication Technology, Electronics and Microelectronics, pp. 369-374, 2011.

[40] D. Radošević, I. Magdalenić and T. Orehovački, "Building Process of SCT Generators", Proceedings of the 36th International Convention on Information and Communication Technology, Electronics and Microelectronics, pp. 1037-1042, 2013.

[41] A. Kvesić, D. Radošević and T. Orehovački, "Using SCT Generator and Unity in Automatic Generation of 3D Scenes and Applications", Proceedings of the 25th Central European Conference on Information and Intelligent Systems, pp. 312-317, 2014.

[42] T. Orehovački, D. Kermek and A. Granić, "Examining the Quality in Use of Web 2.0 Applications: A Three-Dimensional Framework", Communications in Computer and Information Science, vol. 373, pp. 149-153, 2013.

[43] B. W. Boehm. "A Spiral Model of Software Development and Enhancement". Computer, vol. 21, no. 5, pp. 61-72, 1988.