Web form generators design model

Daniel Strmečki, Danijel Radošević, Ivan Magdalenić
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
{daniel.strmecki, danijel.radosevic, ivan.magdalenic}@foi.hr

Abstract. *This paper presents a design model for the development of Web form generators. The proposed model is highly modular and tends to make use of generative programming techniques and the available Web components to enable code generation (both GUI and GPL code). The design model is presented from the user's angle (frontend) and the developer's angle (backend). An UML class diagram of the design solution is presented for the better understanding of the model and its implementation. The design model was implemented and revised on a Java Web framework for the development of Web business applications (a set of Web forms). Two Form specific generators were successfully implemented with very little resource consumption. Both Form specific generators generate fully functional Web forms using only a database specification and a configuration (feature) specification, both in XML format. The implemented generators use code templates to avoid code duplication and to make generators easy to understand and maintain.*

Keywords: generator, generative, Web, form,

1 Introduction

The basic goal of generative programming is to build generative models for system families and to generate concrete systems from those models [1]. Generative programming is used with other disciplines like Domain engineering and Software product lines. The goal of Domain engineering is to develop a common architecture for a system family and to devise a production plan for the family members. Software product lines tends to make use of system's common features to increase the productivity and quality, and of course to reduce the development time, costs and complexity [3]. Generative programming uses feature modelling techniques, in addition to the commonly used UML class diagrams, use cases and other. Feature modelling is a creative activity of modelling the common and the variable properties of concepts and their interdependencies [4]. Unlike generic programming whose focus is on parameterization, generative programming deploys metaprogramming. Metaprogramming refers to developing programs designed to read, generate, analyse or transform other programs, and even modify itself while running. Template metaprogramming is a metaprogramming technique in which code templates are used to generate the source code. Czarnecki and Eisenecker [3] defined a generator as a program that take a higher-lever specification of a piece of software and produces its implementation. The development of generators is expensive and can become quite complex. If we are dealing with complex generators, we need to modularize their design. That means that we should implement complex generators as a set of cooperating, smaller generators [3].

The basic motivation for the development of generators is to avoid repetitive and tedious programming tasks [7]. When working on the development of Web based business software solutions we encounter a set of very similar forms on different projects. Codebooks are the most basic example of such forms. Costumers like to have many codebooks in their applications. Codebooks are used on other forms in a form of a select or lookup input fields. They make software products more configurable and costumers are more generally happy with adaptable software solutions. Codebooks are just one example of Web forms with a lot of repeatable components that can be easily generated. If we develop a large number of such forms, the development of a form generator will

pay off in a certain period of time. Of course, it is necessary to make an assessment on the concrete numbers for every company. This article presents a design model for the development of Web form generators. The model is based on the Web interface, modular generators design and code templates. XML is used for the specification of database tables and features (metadata used by generators, e.g. labels for input fields). The new form feature selection is done on the *Configuration form*. That is a Web form used for the feature selection and generation of other forms. It uses generators to implement the GUI and the GPL source code of the new, generated forms. The result of the generation process is a fully functional Web form or a set of Web forms whose GUI (HTML / JS components) and GPL source code can be extended. That means that other components can be added to forms after their generation. The goal is to reduce the Web forms development time, facilitate maintenance and improve software quality using generative programming techniques.

2 Background of the research

In generative programming, various generators solutions and design models were presented in the beginning of 21st century. The solution proposed in this paper is based on the Web interface, modular design and metaprogramming. It is suited for the development of Web generators for a certain group of forms. For example: codebooks, classic view and edit forms, etc. Each company should identify and analyse a group of forms for which a code generator implementation would pay off in the appropriate period of time.

Jarzabek's XML-based Variant Configuration Language (XVCL) is a script language for configuring and managing variants in programs and other kinds of documents [9]. It uses x-frames as building blocks of program code to be generated. These x-frames are organised in a tree structure, where specification x-frames (SPC-s) are on the top of the hierarchy, describing the process of the system assembling.

Variants can be described as differences between system requests that belong to the same domain. Automating variant management reduces the risk of errors in applications which makes development of domain specific applications cheaper and less time consuming.

GenVoca is a composition model that uses component to define scalable hierarchy system families [1]. The hierarchy of software system is defined with the series of abstract virtual machines implemented by appropriate components while realm is a components cluster. The grammar of software system is defined by realms and their components. A set of sentences defines language and set of compositions defines system family [1].

CodeWorker [6] is a parsing and code generation tool. It uses specification that can be written in arbitrary syntax, but it's required to write scripts to parse specification and populate parse tree with specified data. Those scripts are called parse scripts and CodeWorker enables using of Extended Backus-Naur Form (EBNF) to define their syntax.

Specification-Configuration-Templates (or shortly SCT) is generator model that is based on dynamic frames [8]. Each frame consists of program specification that defines the specific difference of generated application inside its problem domain, configuration with program assembly rules and code template as a building block of a generated program. Specification consists of attributes in a hierarchic order and their values. SCT Generator generates source code by merging features from specification with code templates. Besides defining features of generated applications, specification also defines program files that have to be generated. In a case of a special implementation of the SCT, named as Autogenerator [5], the generated files are virtual, just marking the names of program pieces to be generated. Configuration defines the connections between specification and code templates. Each configuration rule is made of connection, source and (optionally) code template. Code templates are the fragments of a code that contains connections. The process of source code generation starts with the initial SCT frame that contains the complete Specification and Configuration and only one template from the set of all Templates. Other SCT frames are produced dynamically for each connection in the template, forming a generation tree [5].

3 Web form generators design model

The goal is to make generators design as modular as possible to avoid complexity and difficulties in maintaining. That means that instead of developing one complex generator, we should develop a set of smaller, cooperating generators [3]. The modular generators design will make them easier to understand and most importantly easier to maintain. The presented design model of Web form generators uses a database table specification in a XML format.

Figure 1 displays the usage of form generators. The user needs to write the database specification manually or using a form named *DB Designer*. The *DB Designer* is a Web form that simplifies the creation of an XML database table specification. XML database specification contains database tables and its columns definitions. It can also contain some specific metadata used by generators. Once the user has created the database table specification he uses the *DB Creator* form to generate a *DB Record* for the table. *DB Creator* will create a table in the database (using SQL) and generate a *DB Record* in a specific GPL (using ORM). Once the user has created the *DB Record* he can open the *Configuration form* and select the wanted table (*DB Record*) on it.

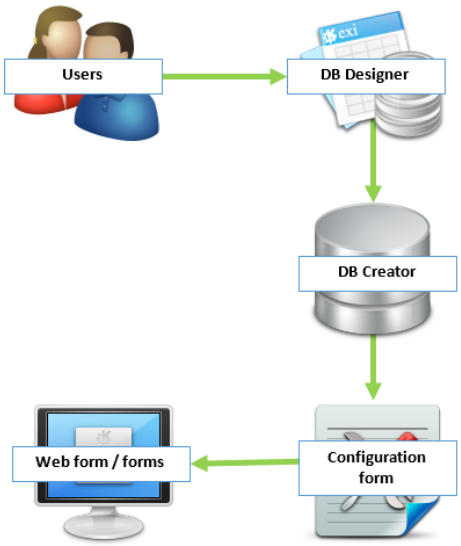

Fig. 1. Frontend of Web form generators model

The *Configuration form* will list all of the table columns and the user can choose a set of provided features. For example:

- columns to show on the screen,
- columns used for searching,
- columns used for sorting,
- document management feature,
- workflow tasks feature,
- etc.

Once the user selects all of the wanted features for his new form / forms, he can generate them with a single command (mouse click). Fully functional Web forms are generated including the GUI (HTML / JS components) and forms GPL code. Figure 2 displays the backend part of the model. It answers the question: how do code generators actually work? The *Configuration form* uses a *Base generator*. That is a generator that uses a set of smaller generators (*Form specific generators* or *Feature specific generators*).

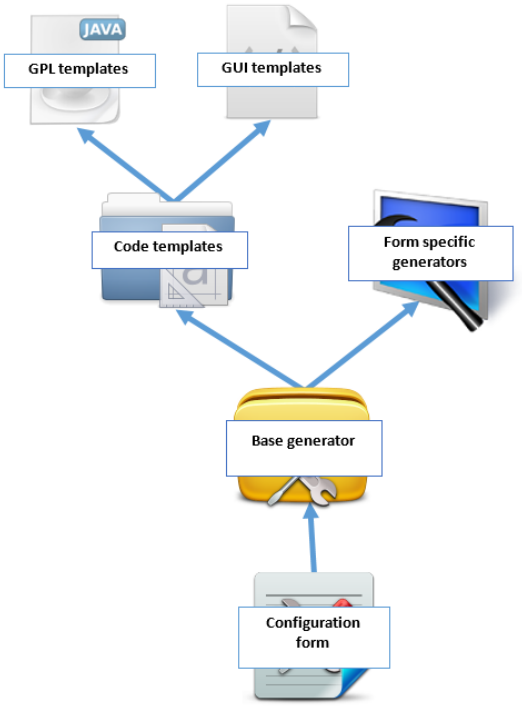

Fig. 2. Backend of Web form generators model

In order to facilitate generators maintenance, a metaprogramming technique has been applied. Both the *Base generator* and the *Form specific generators* use code templates. The model provides two types of templates: GPL templates and GUI templates. The GPL templates contain the GPL code used by (or intended to be used by) multiple generators. The main goal of this approach is to avoid code duplication in multiple code generators. The GUI code templates contain the GUI components used by generators. GUI

code templates can contain HTML / JS code or some other format like XML to specify GUI components. GUI components specified in XML can be transformed (generated) to HTML / JS using GPL / framework specific GUI generators. The proposed model for the form generators design enables the creation of small and modular generators. Code templates are used to avoid code duplication and to facilitate maintenance.

The UML class diagram of the Web form generators design model is shown in figure 3. The diagram specifies that every *Configuration screen* is a *Web form* that uses a *Base generator*. *Base generator* uses *Form specific generators* through the implementation of a *Generator interface*. Each form generated using this approach is also a *Web form.*

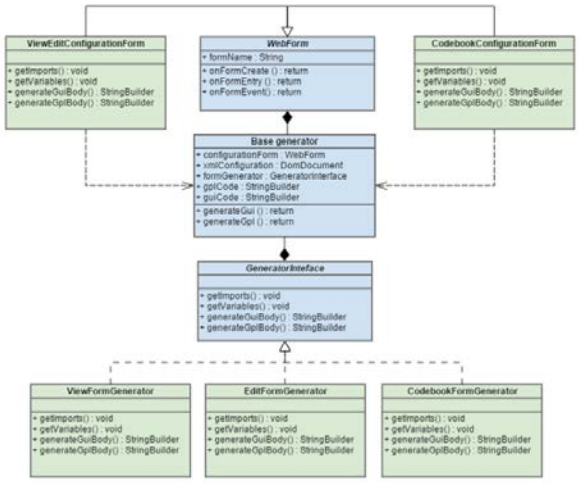

Figure 3: Class diagram

Generators created using this model are used for generating Web forms that work with database tables and use a set of Web GUI components. Those components should be modular and prepared in advance so generators can use them. Most companies now engaged in the development of Web business applications already possess such components, or they can be easily acquired from the Open source community (for example jQuery UI or Kendo UI plugins).

4 Web form generators example

The Web form generators design model has been implemented in Evolution Framework[1]. Evolution Framework is a Java Web framework with a set of tools and components used for creating business Web applications. It is a framework with over a 100 Web components (controls) that can be used by form generators. In this implementation, Evolution Framework is supplemented with two *Form specific generators*: *View screen generator* and *Edit screen generator*. Both *Form specific generators* implement a *Generator interface* shown in figure 4. Using the displayed interface, the *Form specific generators* define their specific imports, class variables, XML (GUI) body and Java (GPL) body. This interface is then used by a *Base generator* to perform form specific code generation as shown on the class diagram.

```
public interface GeneratorInterface {

    public Vector<String> getImports(BaseGenerator baseGenerator, WebZahtjev wz);

    public Vector<String> getVariables(BaseGenerator baseGenerator, WebZahtjev wz);

    public void generateXmlBody(BaseGenerator baseGenerator, WebZahtjev wz);

    public void generateJavaBody(BaseGenerator baseGenerator, WebZahtjev wz);

}
```

Fig. 4. Form specific generators interface

Form specific generators use code templates in their XML and Java body definition. In Evolution Framework, Web component / control definitions are saved in a XML format. A sample of an XML template for an Evolution Framework Web component / control is shown on figure 5. It is the definition for a Grid component (GUI control) where all of the parameters are listed in the first line of the template. Evolution Framework already has the support for the generation of the required HTML and JS code from such XML component definition (a feature called Visual Editor).

```
${name}, ${colnumber}, ${showcheckboxes}, ${showpager}, ${nodatamessage}, ${body}
<grid colnumber="${colnumber}" showcheckboxes="${showcheckboxes}"
      showpager="${showpager}" name="${name}" nodatamessage="${nodatamessage}">
    ${body}
</grid>
```

Fig. 5. Grid component XML template

A sample of a Java code template is shown in figure 6. It is a template of Java code used to execute SQL SELECT queries in Evolution Framework. Same as in the XML template example, all the parameters are listed in the first line of the template.

```
${query_var}, ${query_str}, ${where_var}, ${rs}, ${set_search}, ${foreach_row}
Where ${where_var} = new Where(wz.sql);
${set_search}
String ${query_var} = "${query_str}" + w1;
ResultSet ${rs} = wz.sql.executeQuery(${query_var});
while (${rs}.next()) {
    ${foreach_row}
}
wz.sql.close(${rs});
```

Fig. 6. Evolution Framework Java template for executing select queries

Using code templates and modular design of generators, the implementation of *View screen*

[1] http://www.evolution-framework.com/

generator has less than 330 lines of Java code, and the implementation of *Edit screen generator* has less than 240 lines of Java code. Both generators are used on a single *Configuration form* where the user chooses a set of features for his new generated forms. *Configuration form* for the generation of view and edit forms in Evolution Framework has less than 380 lines of Java code. For comparison, a single generation process, generating a view and edit forms from the simple XML specification displayed in figure 7, generates more than 400 lines of Java code. This sample specification was created manually, but *DB Designer* can also be used to create it.

```
<?xml version="1.0" encoding="windows-1250"?>
<database name="db_fom_demo" java_package="hr.evolva.modules.fom.test.db"
    source_folder="WEB-INF/src/src_m_fom" prefix="fom">
  <table name="fom_documents" class="DB_fom_document">
    <col name="id" type="int(10) not null" label="ID"/>
    <col name="client" type="varchar(250)" label="Client"/>
    <col name="adress" type="varchar(250)" label="Adress"/>
    <col name="city" type="varchar(250)" label="City"/>
    <col name="postal_number" type="int(10)" label="Postal number"/>
    <col name="state" type="varchar(250)" label="State"/>
    <col name="creation_date" type="date" label="Creation date"/>
    <col name="receipt_date" type="date" label="Receipt date"/>
    <col name="title" type="varchar(100)" label="Title"/>
    <col name="description" type="varchar(250)" label="Description"/>
    <col name="note" type="varchar(250)" label="Note"/>
    <col name="amount" type="numeric(10,2)" label="Amount"/>
    <col name="currency" type="varchar(100)" label="Currency"/>
    <pk cols="id"/>
  </table>
</database>
```

Fig. 7. Sample Database specification in XML

DB Creator was then used to actually create the table in the database using SQL CREATE TABLE. The *DB Creator* form also creates a *DB Record* for the sample table using Evolution Framework's custom ORM. User can now open the *Configuration form* and select the XML file with the database specification, and the sample table (or *DB Record*). The *Configuration form* will list all of the table columns and the user can choose which columns he wants to show on the view, and on the edit screen. User also chooses which columns he wants to use for searching (base and advanced search) and mark some columns as invisible on the view form, or invisible / disabled on the edit form. This implementation of *Configuration form* and *Form specific generators* makes use of Evolution Framework Web components / controls so it also enables some additional features like record document management and record tasks (workflow). Figure 8 displays the generated view form. The view form consists of a grid (with support for pagination, columns sorting, rows selections), basic search input field, advanced search popup with a number of different types of input fields, and action buttons (search, new record, delete selected). The edit form consists of a different types of input fields, action buttons (close form, delete record, save record), document management component, workflow tasks component and workflow panel component. From a specification containing 19 lines of XML, more than 400 lines of Java code were generated in this sample as we created two fully functional Web forms in Evolution Framework, using the generators whose design followed the proposed model. A single programmer and 18 work hours were spent on this implementation of two *Form specific generators* in Evolution Framework using the proposed design model.

ViewNewForm

Search: Search Advanced search New record... Delete selected

Retci 1 .. 10 od 10 25 10

#	Client	Adress	City	Postal number	State	Creation date	Receipt date	Title	Description	Note	Amount	Record files	Currency	Record tasks
1	Daniel Strmečki	Matije Gupca 20	Varaždin	42000	Croatia	07.05.2015	08.05.2015	Document 10 Title	Document 10 Description	Document 10 Note	350,00		HRK	Evolva Problem solved Delivered: 05.05.2015 21:03:40 Performed: 05.05.2015 21:03:53
2	Daniel Strmečki	Matije Gupca 20	Čeminac	31325	Croatia	01.05.2015	02.05.2015	Document 09 Title	Document 09 Description	Document 09 Note	300,00	ZIRIZ-06.pdf	EUR	Evolva Problem solved Delivered: 03.05.2015 11:24:57 Performed: 03.05.2015 11:25:05
3	Daniel Strmečki	Zagrebačka 30	Varaždin	42000	Croatia	07.05.2015	08.05.2015	Document 08 Title	Document 08 Description	Document 08 Note	300,00	AspectJ.docx	EUR	Evolva Problem solved Delivered: 02.05.2015 14:27:49 Performed: 02.05.2015 14:27:56
4	Jurica Martinčević	Zagrebačka 25	Varaždin	10000	Croatia	30.04.2015	06.05.2015	Document 07 Title	Document 07 Description	Document 07 Note	550,00	ZIRIZ-03.pdf ZIRIZ-02.pdf	HRK	Evolva Problem solved Delivered: 02.05.2015 12:41:57 Performed: 02.05.2015 12:42:07
5	Julija Peras	Međimurska	Varaždin	42000	Croatia	27.04.2015	02.05.2015	Document 06 Title	Document 06 Description	Document 06 Note	120,00		HRK	Ivo Ivić Delivered: 02.05.2015 11:21:30
6	Nikolina Tomašković	Zagrebačka 77	Osijek	42000	Croatia	28.04.2015	29.04.2015	Document 05 Title	Document 05 Description	Document 05 Note	380,00	AspectJ.docx	EUR	Evolva Problem solved Delivered: 02.05.2015 11:54:04 Performed: 02.05.2015 11:54:16 Evolva Delivered: 05.05.2015 16:47:49
7	Matija Tomašković	Zagrebačka 58	Varaždin	42000	Croatia	01.05.2015	02.05.2015	Document 04 Title	Document 04 Description	Document 04 Note	250,00		EUR	
8	Marko Marić	Vukovarska 50	Zagreb	10000	Croatia	09.04.2015	16.04.2015	Document 03 Title	Document 03 Description	Document 03 Note	350,00	Base Idea.docx	EUR	Evolva Problem solved Delivered: 01.05.2015 11:12:28 Performed: 01.05.2015 11:12:32
9	Maja Majić	Vukovarska 40	Rovinj	10000	Croatia	11.05.2006	12.05.2006	Document 02 Title	Document 02 Description	Document 02 Note	500,00	ZIRIZ-01.pdf	EUR	
10	Pero Perić	Zagrebačka 22	Varaždin	42000	Croatia	11.05.2005	12.05.2006	Document 01 Title	Document 01 Description	Document 01 Note	1.000,00	Raspored.doc	HRK	Evolva Delivered: 02.05.2015 11:11:41

Figure 8: Generated view form

5 Conclusions

Within this paper a design model for Web form generators was proposed. The model was implemented and revised on Evolution Framework form generators implementation. The design model is highly modular and tends to make use of the available components (developed in house or Open source) commonly used in Web forms. The application of the proposed model and the implementation of Web form generators has a number of advantages:

- form development speed,
- less possibility for errors as generation process is well tested,
- code templates reusability,
- the same code for the same functionalities on different forms / projects facilitates maintenance,
- ability to define well commented and easy to understand generated code.

The model was intended for usage in the development of business Web solutions, but it can also be used in the development of educational Web software. It is limited to Web forms working with database tables. Not all of the Web forms that meet this requirement should be generated. It depends on the each company to assess and decide whether a form generator development would pay off for them in a specific case. If the answer is positive, the proposed design model provides a solution to make the form generators modular, easy to understand and maintain.

In our future work, we plan to implement a few more *Specific form generators* in Evolution Framework using the proposed design model. We also plan to test and implement the proposed design model on other solutions for creating Web forms (other GPLs and frameworks).

Further research will be oriented to enabling the form regeneration without the removal of code added after the last generation. The lines of code (functionalities) added manually by developers after form generation shouldn't be deleted upon the form regeneration. This would enable a higher level of extensibility for the proposed design model.

References

[1] Blair, J., Batory, D. A Comparison of Generative Approaches: XVCL and GenVoca. Technical report, The University of Texas at Austin, Department of Computer Sciences, December 2004.

[2] Czarnecki K.: Generative Programming - Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment Based Component Models, PhD dissertation, 1998.

[3] Czarnecki K., U. Eisenecker: Generative Programming: Methods, Tools, and Applications, Paperback, 2000.

[4] Kang K., Lee J., Donohoe P.: Feature-Oriented Product Line Engineering, IEEE Software, 2002.

[5] Magdalenić I., Radoševic D., Orehovački T.: Autogenerator: Generation and execution of programming code on demand, Expert Systems with Applications, 2013.

[6] Lemaire, C.: CODEWORKER Parsing tool and Code generator - User's guide & Reference manual,

http://codeworker.free.fr/CodeWorker.pdf, 2010.

[7] Mernik M.: When and how to develop domain-specific languages, ACM Computing Surveys, 2005.

[8] Radošević D., Magdalenić I: Source Code Generator Based on Dynamic Frames, Journal of Information and Organizational Sciences, 2011.

[9] Zhang H., Jarzabek S.: XVCL: A Mechanism for Handling Variants in Software Product Lines, Science of Computer Programming, 2004.