Information system user interfaces automatic creation

Alexander Korotkov
Moscow Engineering Physics Institute
Moscow, Russia
email: aekorotkov@gmail.com

***Abstract*—Very actual task for the information system is automatically user interface creating. Solution of this task greatly decreases the information system development time. There are various approaches for automatic creation of user interfaces. In this article most popular approaches are surveyed and their problems are described. Author suggests the hybrid approach which allows minimizing the problems of existing approaches.**

***Index Terms*—user interfaces, code generation, information systems, databases, objects inheritance**

I. Introduction

The programming of some kinds of applications from scratch is not reasonable. The continuous extension of the libraries is widely used. The libraries are designed to solve tasks that are isolated from general applications logic. That's why the approaches like solution of general tasks with complex configuration and code generation is also popular.

The user interfaces (UI) creation is the one of the tasks when programming from scratch is unreasonable. The rich UI libraries solve only part of task. Some meta-information that may be helpful to create UI already exists as the rule. It may be database structure for example. On the one hand, UI very frequently needs some project specific features. On other hand, many UI parts are evidently reasonable for automatic creation. The problem is to develop approach which allows to automatically creating UI which can be extended by project specific features.

The various approaches of the automatic creation UI for information systems currently exist. Two approaches are the most popular. The first approach uses the program code generation which can be modified by programmer in a future. The second approach uses runtime generation UI which is based on the meta-information and the user settings probably. Let's consider these approaches in details.

II. Existing approaches

The code generation based on the meta-information is popular practice. This approach is automation of the process which programmers are making manually in other case. When programmer has the task to develop UI for some object he should already has encountered with all parts of this task in some degree. He will copy the parts of a code from his old projects, correct identifiers and do some other correcting to make it working together. Then first version will be done. When we use the code generation most part of these actions are executed automatically (Fig. 1). Automatic code generation eliminates the nasty errors from lack of attention. The problem of this approach is a lot of iterations that development process has usually. These iterations may contain meta-information correction. Used for the code generating the meta-information may changes also. Then there is question how to transfer these changes to already generated and corrected by a programmer code. We can generate the UI code again and manually correct it again (Fig. 2). Otherwise we can manually correct UI code due to meta-information changes (Fig. 3). In any case manually correction is needed and this correction has a not minimal volume.

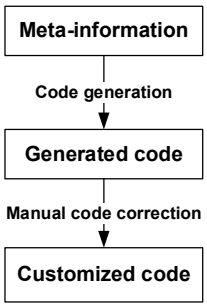

Fig. 1. The scheme of the automatic UI code generation

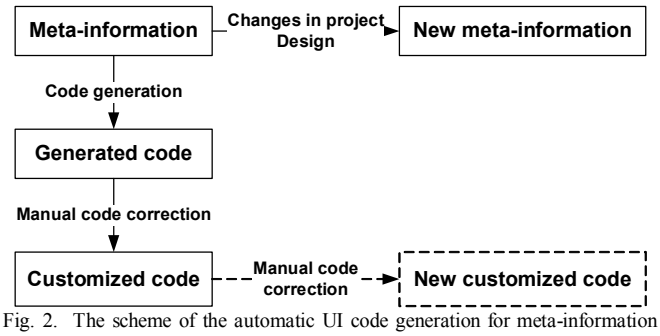

Fig. 2. The scheme of the automatic UI code generation for meta-information changing (the first alternative)

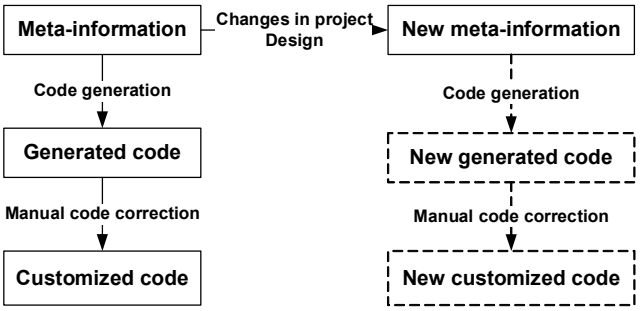

Fig. 3. The scheme of the automatic UI code generation for meta-information changing (the second alternative)

The second approach doesn't use the manual code correction and includes all information needed to configure UI into meta-information. In this approach we have not problem of transferring changes from previous approach because manual generated code correction is not used there.

Databases administration tools use a simple implementation of this approach. These tools provide a viewing and editing interface for each table of a database. The more complex example is the administration generator of Symfony PHP Framework [1].

In this approach any specific UI customizations must be covered by the meta-information and generator possibilities. So if some new sort of UI customization is needed then meta-information and generator must be extended to cover such customization.

The most serious problem of this approach is that reprogramming a UI generator is frequently needed. It may produce the new generator functions to be appropriate to specific project needs but doesn't solve the general problems. In this case the generator design may suffer.

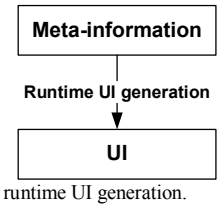

Fig. 4. The scheme of the runtime UI generation.

III. Proposed approach

Author proposes the hybrid approach when generated code and manual corrections are logically separated. When the meta-information is changed than the generated code will be generated again but existing manual corrections do not require any changes or these changes will be minimal. To implement this approach the two tasks should be solved:

1) Code generator should be flexible enough to generate the application which has a skeleton which doesn't require the changes during the customization. Because it would be problematical to implement logically separation of manual corrections which include the application restructure.
2) The mechanism of separation of generated application and manual changes should be found.

For the separation of the generated application and the manual changes the inheritance mechanism is proposed. In this approach for each automatically created UI component the two classes are generated (Fig. 5). The first class is automatically generated UI component. The second class inherits the first class and it is empty initially. Programmer can manually fulfill the second class for UI component customization. When meta-information is changed than the first class will be regenerated but second class do not require any changes or these changes will be minimal and dealing with the manually added features (Fig. 6). This approach is very similar to popular approach in the Object Relation Mapping (ORM) [2] software where base classes of persistent objects are generated and derived class can be manually customized. Let's consider how this approach can be applied to very usual UI components of information system such as the grid and form.

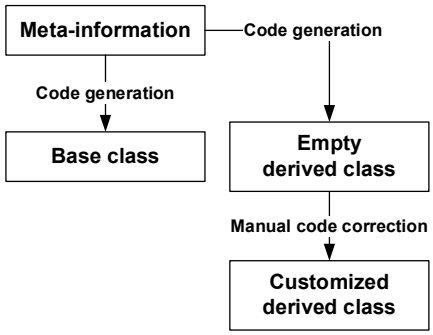

Fig. 5. The scheme of the hybrid approach of a UI generation

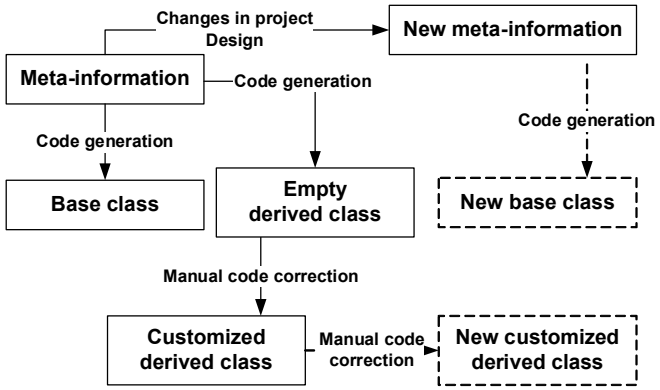

Fig. 6. The scheme of the hybrid approach of a UI for meta-information changing

IV. Implementation

To implement this approach the Web-interface which uses the Javascrit-framework ExtJS [3] is generated. The generated class for gird is derived class from the Ext.Grid. This class contains configuration of columns, render methods of columns, appearance configurations methods of columns and UI objects for cell editing. The derived class can redefine these configurations, methods and objects. Objects in base class are defined by the configurations without the explicit constructor call. It helps to avoid the excess of the objects creation when they are redefined in the derived class. The

generated class for form inherits Ext.FormPanel. This class contains UI objects using for editing of the form data. The derived class can redefine these objects. Similar to the grid base class the objects are defined by their configurations. Derived classes can contain additional methods and properties which implements extended inner logic.

V. Conclusion

The one of problems of automatic UI creation is the developing approach which allows automatically creating UI which may be extended by the project specific features. Various approaches for automatic creation of user interfaces (UI) for information systems currently exist. The one of these approaches uses the program code generation. There is a problem in this approach when meta-information is changed. Then manually changes are required. These changes may concern the one more UI component customization or the manually transfer of the meta-information changes to program code. Other approach is runtime UI generation. The most serious problem of this approach is that interface is strictly limited by laying therein customization possibilities. Reprogramming some feature of such UI is difficult. Author proposes the mixed approach when inheritance is used to separate generated application and manual changes. The two classes are generated for each UI component. The first class is automatically generated UI component. The second class inherits the first class and it is initially empty. Programmer can manually customize the second class. When meta-information is changed the base class is regenerated only and the manual changes in derived class are minimal. This mixed approach make possibility for the minimizing of the problems of the surveyed approaches for the automatic UI generation. This approach was proved to be functional.

References

[1] Symfony Open-Source PHP Web Framework, http://www.symfony-project.org/

[2] Propel ORM PHP framework, http://propel.phpdb.org/trac/

[3] Symfony Admin Generator, http://www.symfony-project.org/book/1_2/14-Generators#Administration