AUTOMATIC CREATION OF USER INTERFACES FOR INFORMATION SYSTEM

Alexander Korotkov
Moscow Engineering Physics Institute
Moscow, Russia

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.

KEYWORDS
User interfaces, code generation, information systems, databases, objects inheritance

1. INTRODUCTION

The user interfaces (UI) creation is the one of the tasks when programming from scratch is unreasonable. 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 of 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 doesn’t use a manual code correction by a programmer. All the settings which is needed for UI customization are included into meta-information in this approach. Let’s consider these approaches in details.

2. EXISTING APPROACHES

The code generation based on the meta-information is popular practice. When the code generation is used most part of the actions which developer performs for UI creation is executed automatically but the manual correction of the generated code is needed also (Figure 1). The problem of this approach is a lot of iterations that development process has usually. Used for the code generating the meta-information may change. Then there is question how to transfer these changes to already generated and corrected by a programmer code. It is possible generate the UI code again and manually correct it again (Figure 2). Otherwise the UI code may be manually corrected due to meta-information changes (Figure 3). In any case additional manually correction is needed and this correction has a not minimal volume.

The second approach doesn’t use the manual code correction and includes all information needed for the UI configuration into meta-information. In this approach there is no problem of transferring changes of a previous approach because the manual generated code correction is not used there.

Databases administration tools use the 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 the Symfony PHP Framework which uses the various settings for the UI customization (Symfony Open-Source PHP Web Framework, 2009).
The most serious problem of this approach is the frequently need of reprogramming the UI generator. 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.

![Diagram 1](image1.png)

**Figure 1.** The scheme of the automatic UI code generation

![Diagram 2](image2.png)

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

![Diagram 3](image3.png)

**Figure 3.** The scheme of the automatic UI code generation for meta-information changing (the first alternative)

![Diagram 4](image4.png)

**Figure 4.** The scheme UI generation when all UI customization settings are included into meta-information

3. **PROPOSED APPROACH**

Author proposes the hybrid approach when the generated code and the 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 would be minimal.

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 (Figure 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. Then the minimum vulnerability with respect to changes in base class (this means the minimum vulnerability with respect to changes in meta-information) will be achieved. If such mechanism will be develop then when the meta-information is changed then the base class will be regenerated but derived class do not require any changes or these changes will be minimal and dealing with the manually added features (Figure 6). This
approach is very similar to popular approach in the Object Relation Mapping software where base classes of persistent objects are generated by data model and derived classes can be manually customized (Propel ORM PHP framework, 2009).

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

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

4. 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 UI for information systems currently exist. The first approach 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 UI generation when all UI customization settings are included into meta-information. The most serious problem of this approach is that the meta-information and the generator extension are needed for each specific UI customization. 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 makes possibility for the minimizing of the problems of the surveyed approaches for the automatic UI generation. This approach was demonstrated on the examples.

REFERENCES