для платформы UbiquiMobile. В конце статьи приводятся демонстрационные примеры, на которых был апробирован DSL язык. В качестве первого примера приводится приложение, позволяющее пользователю пользоваться расписанием электричек. Во втором приложении пользователь может войти в систему для того, чтобы получить check-in код.

Ключевые слова: предметно-ориентированные языки; мобильная разработка


1. Introduction
A large number of platforms, languages, and methods are used in mobile application development. Existing mobile development tools significantly differ from each other, and the common technology implementation problem is still relevant. There are various ways of the high-level description of mobile application – architectural patterns mvc, pac, microkernel, etc. [1]. All these patterns were borrowed from other software areas, are actively applied during mobile application development, but not quite correspond to their nature. Mobile applications differ from desktop and web programs [2]. Mobile applications are commonly used for short sessions, more focused on specific objectives performance.

Use of a suitable architectural pattern allows to increase considerably application development efficiency, but a bigger result can be achieved by graphic languages usage. DSL is the programming language in terms of the concrete subject domain which is applied to the solution of concrete type tasks [3], [4], [5]. Graphic DSL languages help to represent applications using visual diagrams. The result code will be generated according to these diagrams.

The purpose of this article is to develop an architectural template for mobile applications and to create a DSL based on it. DSL should allow describing the main logical application structure in terms of states, controllers and transition conditions between them.

2. Tools
The Modeling SDK technology is used for the graphic DSL implementation [6]. Modeling SDK is the plugin for Visual Studio intended for visual domain-specific languages development. Visual DSL development happens in the following order. At first, the metamodel (the set of all syntactically correct diagrams) is developed and edited, the implemented classes are generated. Then a DSL package compilation and debugging take place in an experimental instance of Visual Studio.

For metamodel programming, the graphic editor of Modeling SDK is used, but also it is possible to redefine or add new methods to the generated partial classes of the C# language. The T4 language is used for code generation [7]. The Dsl and DslPackage projects are automatically created in the new solution of Visual Studio. In the Dsl project, various metamodel artifacts of the created DSL are stored. DslPackage project contains the user interface settings.

3. Controllers and states model
An application state corresponds to some complete logic fragment. The result of state change is data transfer which is logically finished and clear to other states. It is convenient to group states and transition conditions into controllers by their logical connectivity, data community, UI forms, transition frequency and data transfer between states. Grouping states into controllers gives an opportunity to define more strong transition logic, allowing transitions between states in the controller and forbidding them between conditions of unconnected controllers.

The main application cycle is run by the special mechanism starting and switching controllers of application switches to it. There can be several exits in a state. An application can return to the previous controller, switch to the next controller, etc. Execution logic is implemented in terms of the
finite-state machine in the controller. Each controller has a set of the predefined states (in particular, initial and final states), and it is possible to add new states.

Mobile application implementation by means of controllers and states model allows to centralize its logical basis, the structure of the code becomes evident. When using controllers, the aim of the mobile application developer comes down to describing the necessary logical controllers, state and conditions of transition.

4. Graphic DSL Description
The model of controllers and states was tested on mobile applications of different classes and proved the efficiency. But the best results can be achieved, having taken this model as a basis of graphic DSL for mobile applications (see fig.1)

![Fig. 1. Language implementation in Modelling SDK](image)

Basic elements of the language are controllers and its states. States are placed on the controller, can connect among themselves and also to ports of the controller for the conditions description of an entrance and an exit from it. Each state opens in the separate diagram on which conditions of an entrance, an exit from a state and its internal logic are described. The logic of states includes a display of UI forms, processing of their events, services calls, conditions checking, etc. There is a display of a UI form for each state in the language. To connect the existing screen form with a state the ShowForm element is used.

5. Code generation
The language of T4 templates is used for code generation. The main components of the T4 language are directives, blocks of the text and control units. For a generation of the unchangeable code, text blocks are used, and dynamic parts are implemented by means of control units.

As a result of generation, the controllers’ classes appear. Each controller has several states presented in the transfer type form. Process of work is implemented in terms of the finite-state machine. On links between states, the template of transitions are implemented. Controllers can also have ports. Ports are used for transitions between controllers.

The resulting code is applied to UbiqMobile platform [8]. UbiqMobile platform is aimed to cross-platform mobile development. The main features of the platform are that the business logic of all applications is executed on the server. And mobile devices have only thick clients to represent the result of application work.

6. Samples
The purpose of the first sample is to display the train schedule for the user. The application consists of a single controller and two states. In start state, the user can choose departure and destination stations (see fig.2). After clicking on the button, the application will switch the current state from the first state to the second one (see fig.3).

![Fig. 2. Schedule application scheme](image)

![Fig. 3. Schedule application UI forms](image)
The second sample allows the user to log in and receive the code which then can be used later (see fig. 4). There are two controllers in the application: LoginController and MainController. There is also a switching between controllers implemented by means of ports. In LoginController there is only one state. At MainController there are two states: a state with option selection and a state where a user can receive the necessary code. The UbiqMobile UI forms, corresponding to states of the application are given below (see fig. 5).

7. Conclusion

Within this work, the following results were achieved. The graphic DSL for mobile application development is implemented. The code generation for UbiqMobile platform feature is added. Demonstration samples are represented.

References