T. Boiko, Bachelor student
I. Sugonyak, PhD in Tech. Sciences., As. Prof., research advisor
S. Kobzar, Senior Lecturer, language advisor
Zhytomyr State Technological University

ARCHITECTURAL PATTERNS IN SOFTWARE DEVELOPMENT

The modern word is inundated with various programming tools and technologies. Considering this diversity, programmers are able to develop great, complicated and highly functional software. Hence the necessity of structuring big projects has occurred. In order to solve this problem software architectural patterns were created.

An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context. There is a wide range of architectural patterns, which are classified into many groups. However, there are three of these patterns, which are considered the most popular: MVC, MVP, MVVM. The patterns mentioned above are commonly used in web, mobile and desktop applications. They will be discussed in more detail.

MVC, MVP and MVVM are layered style patterns. Their main idea is to decouple project structure into three main components: business logic of the application, user interface (UI) layer and the application layer, which holds the interaction between the app's logic and the UI.

The implementation of these patterns gives such benefits as:

- Ability to develop the interface and logic simultaneously;
- Increased testability;
- Code's reusability;
- Manageability. The UI might be changed without influencing the app's logic.

Although these patterns share the basic idea, each of them has its specific peculiarity.

MVC stands for Model-View-Controller, this pattern may be called the classic architectural pattern, since it is the most widely used. The components of MVC perform their individual duties.

Model. The Model component represents entities of project subject area. It also defines rules for transactions with database or any other external storage.

View. The View itself represents the set of user interface controls on the screen. It is the only layer, which interacts with the user directly. MVC concept implies View as a component, which is able to connect Model straightforwardly, request any Model's data and transform it to interface.

Controller. The Controller is responsible for processing the user's request. It normally serves as a mediator between Model and View. This component gets user's data via View, send it to Model and post new data back to the UI layer. One Controller can interact with several Views and define the appearance of the Views.

MVC is considered to be complex, thus developers need to spend enough time to understand the whole idea. Although the pattern's concept describes decoupling, it does not allow developers of the Model ignore the structure of the Views. It may be found difficult to debug MVC architecture application in order to event-driven nature of the UI code.

The MVC pattern is native architectural decision for Android and IOS mobile development. It is also frequently implemented in web applications; therefore, there are many MVC web frameworks such as Angular.js, ASP.NET MVC, and Ruby on Rails.

The other pattern is MVP (Model-View-Presenter). It is the descendant of MVC design. The Model handles the same responsibilities as in MVC.

Here the View also describes the interface and displays the Model's data; however, unlike the MVC's View component, it interacts nothing but the presenter.

The presenter can be imagined as a transformed Controller. It creates the liaison between View and Model, redirect user's input from View to Model and pass processed data back to the View. The View and the presenter are completely separated and communicate to each other by an interface.

The disadvantages of MVP are similar to the disadvantages of MVC pattern.

Although the MVP pattern performs decoupling more efficiently than MVC, it is not commonly used in the web development. This pattern has implementation for Windows Forms, Silverlight and ASP.NET Web Forms in .NET environment and in different Java, PHP frameworks.

MVVM is the newest pattern among considered above. It divides an application into three components: Model-View-ViewModel. It supports two-way data binding between View and ViewModel, which allows automatic propagation of changes inside the state of ViewModel to the View.

Typically, the Model performs the same duties as Models in MVC and MVP do. The View receives data through the data binding system and display it to the user by the UI controls.

The ViewModel is responsible for defining methods and functions that assist in maintaining the View's state, triggering events in the View and manipulating the Model as a respond to changing of the View.

Thanks to data binding system, MVVM provides automatic connection between the View and the ViewModel, but pattern's creator John Gossman himself points out possible problems of data binding. He states that the system in large applications can result in considerable memory consumption.

The MVVM pattern is used by Windows Presentation Foundation, Silverlight and implemented in Java frameworks for Android. There are also frameworks for web development like KnockoutJS.

Choosing the architecture of the future application, developers ought to set tasks clearly and estimate the given capabilities of used technologies. Traditionally, web developers use MVC pattern, desktop applications provide MVP and MVVM solutions and mobile developers implement different patterns, though MVC is preferable and the classic one for mobile applications.

The architectural patterns are designed to help programmers create pure, cohesive and maintainable solutions but it is still up to developers whether to use provided advantages or not.

REFERENCES

1. Architectural Patterns and Styles [Електронний ресурс] –Режим доступу: https://msdn.microsoft.com/en-us/library/ee658117.aspx?f=255&MSPPError=-2147217396

2. Osmani A. Learning JavaScript Design Patterns /O'Reilly Media, Inc, 2012. – 830 p.