WIKIPEDIA

Model–view–controller

Model–view–controller (**MVC**) is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to, and accepted from, the user.[1][2] The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

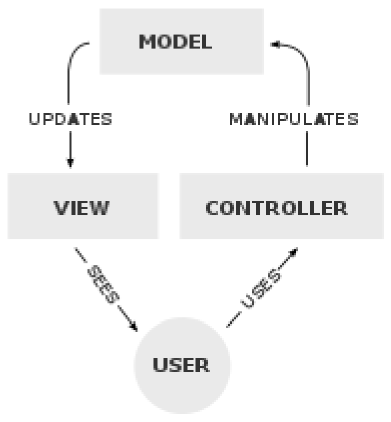

Diagram of interactions within the MVC pattern.

Traditionally used for desktop graphical user interfaces (GUIs), this architecture has become popular for designing web applications and even mobile, desktop and other clients.[3] Popular programming languages like Java, C#, Ruby, PHP and others have popular MVC frameworks that are currently being used in web application development straight out of the box.

Contents

Descriptions

As with other software architectures, MVC expresses the "core of the solution" to a problem while allowing it to be adapted for each system.[4] Particular MVC architectures can vary significantly from the traditional description here.[5]

Components

- The *model* is the central component of the pattern. It expresses the application's behavior in terms of the problem domain, independent of the user interface.[6] It directly manages the data, logic and rules of the application.
- A *view* can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- The third part or section, the *controller*, accepts input and converts it to commands for the model or view.[7]

Interactions

In addition to dividing the application into three kinds of components, the model–view–controller design defines the interactions between them.[8]

- A *model* stores data that is retrieved according to commands from the controller and displayed in the view.
- A *view* generates new output to the user based on changes in the model.
- A *controller* can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g., scrolling through a document, movement of document)

History

One of the seminal insights in the early development of graphical user interfaces, MVC became one of the first approaches to describe and implement software constructs in terms of their responsibilities.[9]

Trygve Reenskaug introduced MVC into Smalltalk-76 while visiting the Xerox Palo Alto Research Center (PARC)[10][11] in the 1970s. In the 1980s, Jim Althoff and others implemented a version of MVC for the Smalltalk-80 class library. Only later did a 1988 article in *The Journal of Object Technology* (JOT) express MVC as a general concept.[12]

The MVC pattern has subsequently evolved,[13] giving rise to variants such as hierarchical model–view–controller (HMVC), model–view–adapter (MVA), model–view–presenter (MVP), model–view–viewmodel (MVVM), and others that adapted MVC to different contexts.

The use of the MVC pattern in web applications exploded in popularity after the introduction of NeXT's WebObjects in 1996, which was originally written in Objective-C (that borrowed heavily from Smalltalk) and helped enforce MVC principles. Later, the MVC pattern became popular with Java developers when WebObjects was ported to Java. Later frameworks for Java, such as Spring (released in October 2002), continued the strong bond between Java and MVC. The introduction of the frameworks Django (July 2005, for Python) and Rails (December 2005, for Ruby), both of which had a strong emphasis on rapid deployment, increased MVC's popularity outside the traditional enterprise environment in which it has long been popular. MVC web frameworks now hold large market-shares relative to non-MVC web toolkits.[14]

Use in web applications

Although originally developed for desktop computing, MVC has been widely adopted as an architecture for World Wide Web applications in major programming languages. Several web frameworks have been created that enforce the pattern. These software frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server.[15]

Some web MVC frameworks take a thin client approach that places almost the entire model, view and controller logic on the server. This is reflected in frameworks such as Django, Rails and ASP.NET MVC. In this approach, the client sends either hyperlink requests or form submissions to the controller and then receives a complete and updated web page (or other document) from the view; the model exists entirely on the server.[15] Other frameworks such as AngularJS, EmberJS, JavaScriptMVC and Backbone allow the MVC components to execute partly on the client (also see Ajax).

Goals of MVC

Simultaneous development

Because MVC decouples the various components of an application, developers are able to work in parallel on different components without impacting or blocking one another. For example, a team might divide their developers between the front-end and the back-end. The back-end developers can design the structure of the data and how the user interacts with it without requiring the user interface to be completed. Conversely, the front-end developers are able to design and test the layout of the application prior to the data structure being available.

Code reuse

By creating components that are independent of one another, developers are able to reuse components quickly and easily in other applications. The same (or similar) view for one application can be refactored for another application with different data because the view is simply handling how the data is being displayed to the user

Advantages & disadvantages

Advantages

- *Simultaneous development* – Multiple developers can work simultaneously on the model, controller and views.
- *High cohesion* – MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.
- *Low coupling* – The very nature of the MVC framework is such that there is low coupling among models, views or controllers
- *Ease of modification* – Because of the separation of responsibilities, future development or modification is easier
- *Multiple views for a model* – Models can have multiple views

Disadvantages

- *Code navigability* – The framework navigation can be complex because it introduces new layers of abstraction and requires users to adapt to the decomposition criteria of MVC.
- *Multi-artifact consistency* – Decomposing a feature into three artifacts causes scattering. Thus, requiring developers to maintain the consistency of multiple representations at once.
- *Pronounced learning curve* – Knowledge on multiple technologies becomes the norm. Developers using MVC need to be skilled in multiple technologies.

See also

- Hierarchical model–view–controller
- Model–view–adapter
- Model–view–presenter
- Model–view–viewmodel
- Presentation–abstraction–control
- Action–domain–responder
- Observer pattern
- Strategy pattern
- Naked objects

References

1. "More deeply, the framework exists to separate the representation of information from user interaction." The DCI Architecture: A New Vision of Object-Oriented Programming (http://www.artima.com/articles/dci_vision.html) – Trygve Reenskaug and James Coplien – March 20, 2009.
2. Burbeck (1992): "... the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object."
3. Davis, Ian. "What Are The Benefits of MVC?" (http://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/) *Internet Alchemy*. Retrieved 2016-11-29.
4. Gamma, Erich et al. (1994) *Design Patterns*

5. Moore, Dana et al. (2007) *Professional Rich Internet Applications: Ajax and Beyond*: "Since the origin of MVC, there have been many interpretations of the pattern. The concept has been adapted and applied in very different ways to a wide variety of systems and architectures."
6. Burbeck, Steve (1992) Applications Programming in Smalltalk-80:How to use Model–View–Controller (MVC) (https://web.archive.org/web/20120729161926/http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html)
7. Simple Example of MVC (Model–View–Controller) Design Pattern for Abstraction (http://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design)
8. Buschmann, Frank (1996) *Pattern-Oriented Software Architecture*.
9. Model–View–Controller History (http://c2.com/cgi/wiki?ModelViewControllerHistory). C2.com (2012-05-11). Retrieved on 2013-12-09.
10. Notes and Historical documents (http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html) from Trygve Reenskaug, inventor of MVC.
11. "A note on DynaBook requirements", Trygve Reenskaug, 22 March 1979, SysReq.pdf (http://folk.uio.no/trygver/1979/sysreq/SysReq.pdf).
12. Krasner, Glenn E.; Pope, Stephen T. (Aug–Sep 1988). "A cookbook for using the model–view controller user interface paradigm in Smalltalk-80" (http://dl.acm.org/citation.cfm?id=50757.50759). *The Journal of Object Technology*. SIGS Publications. Also published as "A Description of the Model–View–Controller User Interface Paradigm in the Smalltalk-80 System (https://web.archive.org/web/20100921030808/http://www.itu.dk/courses/VOP/E2005/VOP2005E/8_mvc_krasner_and_pope.pdf)" (Report), ParcPlace Systems; Retrieved 2012-06-05.
13. The evolution of MVC and other UI architectures (http://martinfowler.com/eaaDev/uiArchs.html) from Martin Fowler.
14. Hot Frameworks (http://hotframeworks.com)
15. Leff, Avraham; Rayfield, James T. (September 2001). *Web-Application Development Using the Model/View/Controller Design Pattern*. IEEE Enterprise Distributed Object Computing Conference. pp. 118–127.

Bibliography

External links

- What Are The Benefits of MVC? – quotes at length from the Gang of Four
- Martin Fowler on the history of UI Architectures and the evolution of MVC
- Understanding MVC architecture – a quick explanation on YouTube
- 1. MVC and Introduction to Objective-C (September 27, 2011). Stanford University introductory lecture on the MVC pattern. on YouTube
- Cocoa Core Competencies: Overview of the MVC pattern.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Model–view–controller&oldid=811602075"

This page was last edited on 22 November 2017, at 18:21.

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.