Model–view–controller

From Wikipedia, the free encyclopedia

Model–view–controller (**MVC**) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.[1][2]

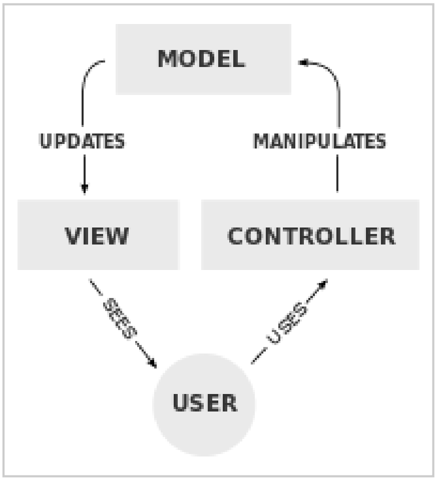
Contents

Overview

As with other software patterns, MVC expresses the "core of the solution" to a problem while allowing it to be adapted for each system.[3] Particular MVC architectures can vary significantly from the traditional description here.[4]

Components

The central component of MVC, the *model*, captures the behavior of the application in terms of its problem domain, independent of the user interface.[5] The model 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, the *controller*, accepts input and converts it to commands for the model or view.[6]



A typical collaboration of the MVC components

Interactions

In addition to dividing the application into three kinds of components, the model–view–controller design defines the interactions between them.[7]

- 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.,

by scrolling through a document).
- A **model** stores data that is retrieved to the controller and displayed in the view. Whenever there is a change to the data it is updated by the controller.
- A **view** requests information from the model that it uses to generate an output representation to the user.

Use in web applications

Although originally developed for desktop computing, model–view–controller has been widely adopted as an architecture for World Wide Web applications in major programming languages. Several commercial and noncommercial web application frameworks have been created that enforce the pattern. These frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server.[8]

Early web MVC frameworks took a thin client approach that placed almost the entire model, view and controller logic on the server. In this approach, the client sends either hyperlink requests or form input 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.[8] As client technologies have matured, frameworks such as AngularJS, Ember.js, JavaScriptMVC and Backbone have been created that allow the MVC components to execute partly on the client (also see Ajax).

History

MVC was one of the seminal insights in the early development of graphical user interfaces, and 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 Xerox Parc[10][11] in the 1970s. In the 1980s, Jim Althoff and others implemented a version of MVC for the Smalltalk-80 class library. It was only later, in a 1988 article in The Journal of Object Technology, that MVC was expressed as a general concept.[12]

The MVC pattern has subsequently evolved,[13] giving rise to variants such as HMVC, MVA, MVP, MVVM, and others that adapted Model View Controller to different contexts.

See also

- Hierarchical model–view–controller
- Model–view–adapter
- Model–view–presenter
- Model View ViewModel
- Observer pattern
- Presentation–abstraction–control
- Three-tier architecture

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. Gamma, Erich et al. (1994) *Design Patterns*
4. 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."
5. Burbeck, Steve (1992) Applications Programming in Smalltalk-80(TM):How to use Model–View–Controller (MVC) (http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html)
6. 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)
7. Buschmann, Frank (1996) *Pattern-Oriented Software Architecture*.
8. 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.
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 JOT* (SIGS Publications). Also published as "A Description of the Model–View–Controller User Interface Paradigm in the Smalltalk-80 System (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.

External links

- What Are The Benefits of MVC? (http://blog.iandavis.com/2008/12/09/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 (http://martinfowler.com/eaaDev/uiArchs.html)
- Understanding MVC architecture – a quick explanation (https://www.youtube.com/watch?v=eTdVkgF_Slo) on YouTube
- 1. MVC and Introduction to Objective-C (September 27, 2011). Stanford University introductory lecture on the MVC pattern. (https://www.youtube.com/watch?v=6EcjhVwH0Dw) on YouTube
- Cocoa Core Competencies: (https://developer.apple.com/library/mac/documentation/general/conceptual/devpedia-cocoacore/MVC.html) Overview of the MVC pattern.

Wikibooks has a book on the topic of: ***Computer Science Design Patterns/Model–view–controller***

Retrieved from "https://en.wikipedia.org/w/index.php?title=Model–view–controller&oldid=668289942"

Categories: Software design patterns | Architectural pattern (computer science)