Zitate:

An easy way to understand MVC: the model is the data, the view is the window on the screen, and the controller is the glue between the two. -- [ConnellyBarnes](http://c2.com/cgi/wiki?ConnellyBarnes)

<http://c2.com/cgi/wiki?ModelViewController>

Merksätze

 "**We need SMART Models, THIN Controllers, and DUMB Views**"

<http://c2.com/cgi/wiki?ModelViewController>

The Controller does not oversee the operation of the Views and Models - it's not a [GodClass](http://c2.com/cgi/wiki?GodClass). The controller mediates communication and unifies validation, using either direct calls or the [ObserverPattern](http://c2.com/cgi/wiki?ObserverPattern).

Definitionen

Smalltalk-80 System:

Eine der zentralen Ideen: Model-View-Controller

Gut definierter Controller mit gut definierter, abgegrenzter Bedeutung.

Eine der ersten Diskussion "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80", by Glenn Krasner and Stephen Pope, August/September 1988 [JournalOfObjectOrientedProgramming](http://c2.com/cgi/wiki?JournalOfObjectOrientedProgramming) (JOOP).

Dieser Artikel hat MVC wie folgt definiert:

"[ModelViewController](http://c2.com/cgi/wiki?ModelViewController) (MVC) programming is the application of this three-way factoring whereby objects of different classes take over the operations related to the application domain (the model), the display of the application's state (the view), and the user interaction with the model and the view (the controller)."

Later, the article more closely defines these terms:

"Models -- The model of an application is the domain-specific software simulation or implementation of the application's central structure."

"Views -- In this metaphor, views deal with everything graphical: they request data from their model and display the data."

"Controllers -- Controllers contain the interface between their associated models and views and the input devices (e.g., keyboard, pointing device, time)."

In dieser Interpretation werden Controller als schlichte und gut abgegrenzte Klassen beschrieben. Diese kümmern sich um die verarbeitung von Event loops für eine bestimmte View.

Für Smalltalk-80- und VisualWorks-Programmierer sind Controller eine Unterklasse von „Controller“. Mehr nicht.

Reinterpretation von IvarJacobson 1992 in „Object-Oriented Software Engineering: A Use Case Driven Approach“

Hauptfokus: Entwicklung mehrerer Modelle in einem großen System welches auf Use Cases basiert.

Nachdem ein Anforderungsmodell definiert wurde, wird ein Analysemodell definiert. Seine eigenen Worte dabei: „Dieses Modell zielt darauf ab, das System unabhängig von der eigentlichen Entwicklungsumgebung zu strukturieren.“

A major focus of Jacobson's book was in developing several "Models" of a large system based around [UseCases](http://c2.com/cgi/wiki?UseCases). After defining a Requirements model, his method moves on to define an [AnalysisModel](http://c2.com/cgi/wiki?AnalysisModel). In his own words: "This model aims to structure the system independently of the actual implementation environment".

Later, he describes the overall structure of the [AnalysisModel](http://c2.com/cgi/wiki?AnalysisModel). In his words, "In the information space of this model, our aim is to capture information, behavior and presentation. The [AnalysisModel](http://c2.com/cgi/wiki?AnalysisModel) is built by specifying objects in this information space." Finally, he states "Many object-oriented analysis methods choose to have only one object type, which can be placed anywhere within this space. We have chosen to use three object types. The object types used in the[AnalysisModel](http://c2.com/cgi/wiki?AnalysisModel) are entity objects, interface objects and control objects".

Ivar goes on to describe his entity objects as being close to what MVC has described as one type of Model objects - that is to say those objects found in the analysis of the problem domain. This leaves the question of his control objects - what are they? Jacobson himself states: "...in more complex use cases, there often remains behavior that is not naturally placed in either of these two object types [meaning entity and interface objects]. Such behavior is placed in control objects. The control objects typically act as glue which unites other objects so that they form one use case".

So, here we see a completely different meaning for "control" or "controller" from what was defined for the "controller" in Smalltalk-80. What happened is that Jacobson's definition (which comes closer to the traditional English meaning of the word "control") has become adopted more and more frequently, to the detriment of the other meaning.

Now, Ivar was not the only person to redefine the term, or provide a similar meaning. As far back as 1987, a paper by Joelle Coutaz referenced in[PatternOrientedSoftwareArchitectureOne](http://c2.com/cgi/wiki?PatternOrientedSoftwareArchitectureOne) proposed a system called [PresentationAbstractionControl](http://c2.com/cgi/wiki?PresentationAbstractionControl) (PAC), which maps (roughly) to the notions of View/Controller, Model, and Mediator. This was written in pattern form in POSA in 1995.

Also, in 1992, Greg Hendley and Eric Smith wrote a series of articles in the Smalltalk Report describing what they referred to as the "Interface-Control-Model (ICM)" architecture. This was again, somewhat similar to Jacobson's concept and nearly identical to the PAC architecture.

So, why do we care about this fiddling over meaning? The answer is that in Java BOTH meanings are bandied about. For instance, Java Swing is discussed as being derived from a pure MVC architecture. The major difference being that in Swing, the View and Controller (using terms adopted from Smalltalk-80) are folded together into a single component. In the words of the Getting Started with Swing document on Sun's web site "traditional MVC architecture makes it very hard to create a generic controller that doesn't know at design time what kind of view will eventually be used to display it." This sentence shows they understood the Smalltalk-80 meanings of a view and a controller and chose to NOT incorporate them into their design.

However, MVC is often used as a synonym for something closer to PAC or ICM. I myself have been guilty of this, as I was in my recent (June 1999) Java Report article called "Using Server-Side Java Successfully".

Kyle Brown, IBM Corporation

<http://c2.com/cgi/wiki?WhatsaControllerAnyway>

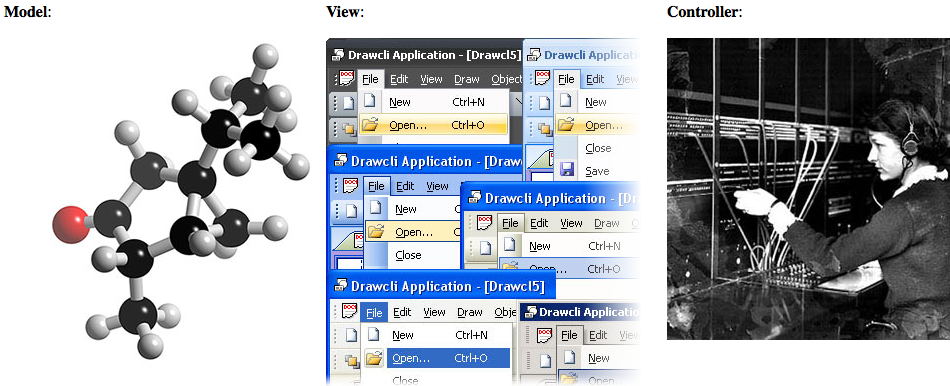
Geschichte

Model-View-Controller is the concept introduced by Smalltalk's inventors ([TrygveReenskaug](http://c2.com/cgi/wiki?TrygveReenskaug) and others) of encapsulating some data together with its processing (the model) and isolate it from the manipulation (the controller) and presentation (the view) part that has to be done on a [UserInterface](http://c2.com/cgi/wiki?UserInterface).

<http://c2.com/cgi/wiki?ModelViewController>

Konzept

* A **model** is an object representing data or even activity, e.g. a database table or even some plant-floor production-machine process.
* A **view** is some form of visualization of the state of the model.
* A **controller** offers facilities to change the state of the model.

Smalltalk provides mechanisms to link models, views and controllers in some standard way in order for a model to communicate state changes to every attached view (there can be several, as you can see). Model state changes happen either because a controller issued some command or for some internal reason.

Note that the term Controller has adopted two radically different meanings

<http://c2.com/cgi/wiki?ModelViewController>