Definition

MVC behandelt drei Rollen:

Model

* Implementiert die zentrale Struktur der Anwendung. (Datenhaltung)

View

* Repräsentiert die Anzeige des Models in dem User Interface

Controller

* Verwaltet Benutzereingaben, manipuliert das Model und aktualisiert die View

Es gibt keine Standarddefinition für MVC. Viele Frameworks benutzen unterschiedliche Versionen.

Schlüsselaspekte von MVC sind:

Die Präsentation/Ansicht von dem Modell trennen

* Ermöglicht es mehrere verschiedene User Interfaces zu implementieren und die Module besser zu testen

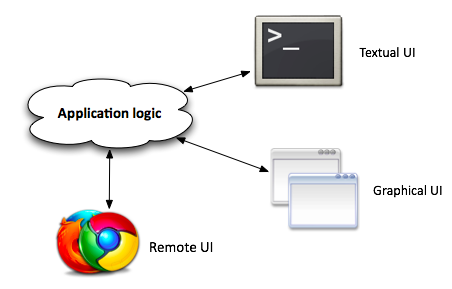
Den Controller von der Präsentation trennen.

* Besonders nützlich mit Web Interfaces. (bei GUI frameworkds eher weniger)

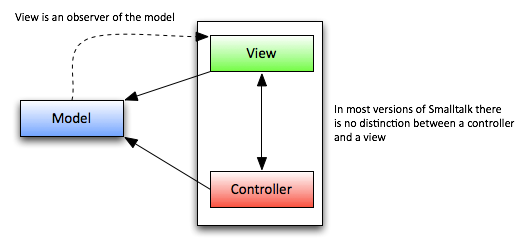
Das wichtigste jedoch sei die Trennung von Model und View.

Martin Fowler: „The irony ist hat almost every version of smalltalk didn’t actually make a view/controller separation“

Smalltalk:



Things to note about Smalltalk's dependency structure:



* The model does not depend on the view or the controller
* [Observer pattern](http://en.wikipedia.org/wiki/Observer_pattern) was used to update views when models changed
* Smalltalk-80 MVC did not make a view/controller separation

<http://amix.dk/blog/post/19615>

Geschichte

MVC wurde von Trygve Reenskaug im Jahre 1979 das erste Mal beschrieben.

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)."

Die einzelnen Module werden wie folgt beschrieben:

"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.“

Die Struktur des AnalyseModells wird so beschrieben:

„Das Ziel ist es im Informationsraum des Modells, Informationen, Verhalten und Präsentation einzufangen. Das Analysemodell besteht aus spezifischen Objekten dieses Informationsraumes.“

„Viele Objekt-Orientierte Analysemethoden haben nur einen Objekttyp, welcher überall in diesem Informationsraum passt. Die Objekttypen des Analysemodells sind Entitätsobjekte, Interfaceobjekte und Kontrollobjekte.“

Ivar beschreibt seine Entitätsobjekte ähnlich wie ein Modellobjekt.

Kontrollobjekt: Bei komplexeren UseCases bleiben oft Anwendungen übrig welche weder als Modell noch als View bezeichnet werden können. Diese sind Kontrollobjekte. Sie verhalten sich wie Kleber welche andere Objekte zusammenhalten und somit einen Anwendungsfall beschreiben.

Ivar Jacobsons definition wurde nach und nach immer mehr angenommen (zum Nachteil der anderen Definition von Smalltalk-80).

Weitere Neudefinition [PresentationAbstractionControl](http://c2.com/cgi/wiki?PresentationAbstractionControl) (PAC), in einem Paper von Joelle Coutaz im Jahr 1987 aus „[Pattern Oriented Software Architecture One](http://c2.com/cgi/wiki?PatternOrientedSoftwareArchitectureOne)“. Dieses beschreibt grob View/Controller Model and Mediator. Dieses wurde in Pattern form in POSA 1995 beschrieben.

In 1992 haben Greg Hendley und Eric Smith mehrere Artikel im Smalltalk Report verfasst welche sie als „Interface-Control-Model (ICM)“ Architektur beschrieben. Dieses war sehr ähnlich zu Jacobsons concept und sogar fast identisch zu Presentation Abstraction Control (PAC).

Wieso das Hin und Her über die Bedeutung? Beide (PAC und ICM) werden in Java immer wieder genannt. Java Swing soll direkt von der puren MVC Architektur abgeleitet sein.

In Swing sind View und Controller zu einer Komponente zusammengefasst. Im Dokument „Getting started with Swing“ von Sun’s Webseite steht „traditionelle MVC Architekturen machen es sehr schwer generische Controller zu erstellen welche die Art der View zu Laufzeit erkennen die genutzt wird“. Sun hat also die Bedeutung von MVC von Smalltalk-80 absichtlich nicht in ihr Design übernommen.

MVC wird oft als synonym zu PAC oder ICM genutzt. Kyle Brown, IBM Corporation zb. In „Using Server-Side Java Successfully“