**Модель-вид-контролер** (або **Модель-вигляд-контролер**,  **Модель-представлення-контролер** [англ.](http://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Model-view-controller*, **MVC**) — [архітектурний шаблон](http://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%BD%D1%96_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), який використовується під час проектування та розробки [програмного забезпечення](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F).



Рис 1.1 Концепція Model-View-Controller

Цей шаблон поділяє систему на три частини: [модель даних](http://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), вигляд даних та [керування](http://uk.wikipedia.org/wiki/%D0%9A%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F). Застосовується для відокремлення даних (модель) від [інтерфейсу користувача](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0)(вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

Дана схема проектування часто використовується для побудови архітектурного каркасу, коли переходять від теорії до реалізації в конкретній предметній області.

Історія

Концепція MVC була описана у 1979 році [Трюгве Реенскауг](http://ru.wikipedia.org/w/index.php?title=%D0%A2%D1%80%D1%8E%D0%B3%D0%B2%D0%B5_%D0%A0%D0%B5%D0%B5%D0%BD%D1%81%D0%BA%D0%B0%D1%83%D0%B3&action=edit&redlink=1) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) [*Trygve Reenskaug*](http://en.wikipedia.org/wiki/Trygve_Reenskaug)), який тоді працював над мовою програмування [Smalltalk](http://ru.wikipedia.org/wiki/Smalltalk) в [Xerox PARC](http://ru.wikipedia.org/wiki/Xerox_PARC). Оригінальна реалізація описана в статті «Applications Programming in Smalltalk-80: How to use Model-View-Controller». Потім Джим Алтоф з командою розробників реалізували версію MVC для бібліотеки класів [Smalltalk](http://ru.wikipedia.org/wiki/Smalltalk)-80.

В оригінальній концепції була описана сама ідея і роль кожного із елементів: моделі, представлення і контролера. Та зв’язки між ними були описані без конкретизації. Крім того, розрізняли дві основні модифікації:

1. Пасивна модель – модель не має ніяких способів впливу на представлення чи контролер і використовується ними в якості джерела даних для відображення. Всі зміни моделі відстежуються контролером і він же відповідає за перемальовування представлення, якщо це необхідно. Така модель частіше всього використовується в структурному програмуванні, оскільки у цьому випадку модель являє собою просто структуру даних, без методів їх обробки.
2. Активна модель – модель сповіщає представлення про те, що в ній виникли зміни, а представлення, які зацікавленні в сповіщенні, підписуються на ці сповіщення. Це дозволяє зберегти незалежність моделі як від контролера, так і від представлення.

Класичною реалізацією концепції MVC вважають версію з активною моделлю.

Із розвитком Об’єктно-орієнтованого програмування і поняття про шаблони проектування був створений ряд модифікацій концепції MVC, які при реалізації у різних авторів можуть відрізнятися від оригінальної. Так, наприклад, Еріан Вермі в 2004 році писав приклад узагальненого MVC.

Призначення

Основна мета використання цієї концепції у розділенні бізнес-логіки(моделі) від її візуалізації(представлення, вигляду). За рахунок такого розмежування підвищується можливість повторного використання. Найбільш корисне застосування даної теорії у тих випадках, коли користувач повинен бачити ті ж самі дані одночасно у різних контекстах та/або з різних точок зору. Виконуються наступні задачі:

1. До однієї моделі можна приєднати декілька видів, при цьому на чіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми.
2. Не чіпаючи реалізацію вигляду, можна змінити реакції на дії користувача(натиснення мишкою на копку, введення даних), для цього достатньо використовувати інший контролер.
3. Ряд розробників спеціалізуються лише в одній із областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку. Тому можливо досягнути того, що програмісти, які займаються розробкою бізнес-логіки(моделі), взагалі не будуть обізнані в тому, яке представлення використовуватиметься.

Концепція

Концепція MVC дозволяє розділити дані, представлення і обробку дій користувача на три окремих компоненти:

* Модель([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) *Model*) Модель представляє інформацію: дані і методи роботи з цими даними, реагує на запити, змінюючи свій стан. Не містить інформації, про те, як ці знання можна візуалізувати.
* Представлення, вигляд ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) *View*). Відповідає за відображення інформації(візуалізацію). Часто в якості представлення виступає форма(вікно) з графічними елементами.
* Контролер ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) *Controller*). Забезпечує зв’язок між користувачем і системою: контролює введення даних користувачем і використовує модель і представлення для реалізації необхідної реакції, керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель.

Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

Важливо зазначити, що як представлення, так і контролер залежать від моделі. Однак модель не залежить ні від представлення, ні від контролера. Тим самим досягається призначення такого розмежування: воно дозволяє будувати модель незалежно від візуального представлення, а також створювати декілька різних представлень для однієї моделі

Для реалізації схеми Model-View-Controller використовується достатньо велике число шаблонів проектування(в залежності від складності архітектурного рішення), основні з яких стратегія, композит, спостерігач.

Найбільш типова реалізація відділяє вигляд від моделі шляхом встановлення між ними протоколу взаємодії, використовуючи апарат подій(підписка/сповіщення). При кожній зміні внутрішніх даних в моделі вона сповіщає всі залежні від неї представлення і представлення обновлюється. Для цього використовують шаблон спостерігач. При обробці реакції користувача вигляд обирає, в залежності від потрібної реакції, потрібний контролер, який забезпечує той чи інший зв’язок з моделлю. Для цього використовується шаблон стратегія, або замість цього може бути модифікація із використанням шаблону команда. А для можливості однотипної роботи з підоб’єктами складно-скомпонованого ієрархічного виду може використовуватись шаблон композит. Крім того, можуть використовуватись і інші шаблони проектування, наприклад, фабричний метод, який дозволить задати за замовчуванням тип контролера для відповідного виду.

**Найчастіші помилки**

Програмісти-початківці (особливо в веб-програмуванні, де абревіатура MVC стала популярною) дуже часто трактують архітектурну модель MVC як пасивну модель MVC. У цьому випадку модель виступає виключно сукупністю функцій для доступу до даних, а контролер містить бізнес логіку. В результаті код моделі по факту являється засобом для отримання даних із СУБД, а контролер являє собою типовий модуль, наповнений бізнес-логікою, або скрипт у термінології веб-програміста. В результаті такого розуміння MVC розробники стали писати код, який Pádraic Brady, відомий у колах спільноти Zend Framework, охарактеризував як ТТПК – «Товсті тупі потворні контролери» (Fat Stupid Ugly Controllers)

Середньостатистичний ТТПК отримував дані із БД (використоруючи рівень абстракції бази даних, враховуючи, що ще модель) або маніпулював, перевіряв, записував, а також передавав дані у вигляд. Такий підхід став дуже популярним тому, що використання таких контролерів схоже на класичну практику використання окремого php-файлу для кожної сторінки додатку.

Але в об’єктно-орієнтованому програмуванні використовується активна модель MVC, де модель – це не тільки сукупність коду доступу до даних і СУБД, але і вся бізнес логіка. Варто зазначити про можливості моделі інкапсулювати в собі інші моделі. В свою чергу контролери являють собою лише елементи системи, в безпосередні обов’язки яких входить прийом даних із запиту і передача їх іншим елементам системи. Тільки у цьому випадку контролер стає «тонким» і виконує виключно функції зв’язкової ланки(*glue layer*) між певними компонентами системи.

**Реалізації**

Концепція MVC вперше застосувалася при проектуванні мови програмування [Smalltalk](http://uk.wikipedia.org/wiki/Smalltalk) як модель для [інтерфейсу користувача](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0) . Також в область застосування концепції входить реалізація каркаса Документ-Вид (Document-View) в рамках бібліотеки [MFC](http://uk.wikipedia.org/wiki/MFC) для мови Visual C++ . У сучасних технологіях концепція MVC представлена схемою JSP Model 1/2 для динамічної обробки Web-змісту на основі [Java Server Pages](http://uk.wikipedia.org/wiki/Java_Server_Pages) (JSP).

**Java**

У мові програмування [Java](http://uk.wikipedia.org/wiki/Java) концепція MVC підтримується на рівні стандартних класів-бібліотек. В результаті використання парадигми MVC програміст отримує в своє розпорядження могутню структуру об'єктів-компонентів, функції яких чітко розмежовані, що гарантує надійність і розширюваність системи, що розробляється.

* Веб-каркаси: [Struts](http://uk.wikipedia.org/wiki/Struts), [WebWork](http://uk.wikipedia.org/w/index.php?title=WebWork&action=edit&redlink=1" \o "WebWork (ще не написана))/Struts 2, [Spring](http://uk.wikipedia.org/w/index.php?title=Spring&action=edit&redlink=1) MVC, [JavaServer Faces](http://uk.wikipedia.org/wiki/JavaServer_Faces" \o "JavaServer Faces) (JSF), [Tapestry](http://uk.wikipedia.org/w/index.php?title=Tapestry&action=edit&redlink=1)
* Десктоп-каркаси: [Swing](http://uk.wikipedia.org/wiki/Swing_(Java)), [JFace](http://uk.wikipedia.org/wiki/JFace" \o "JFace)