MVC vs PAC

MVC (Model-View-Controller)

- Architektonický vzor zabývající se uživatelským rozhraním

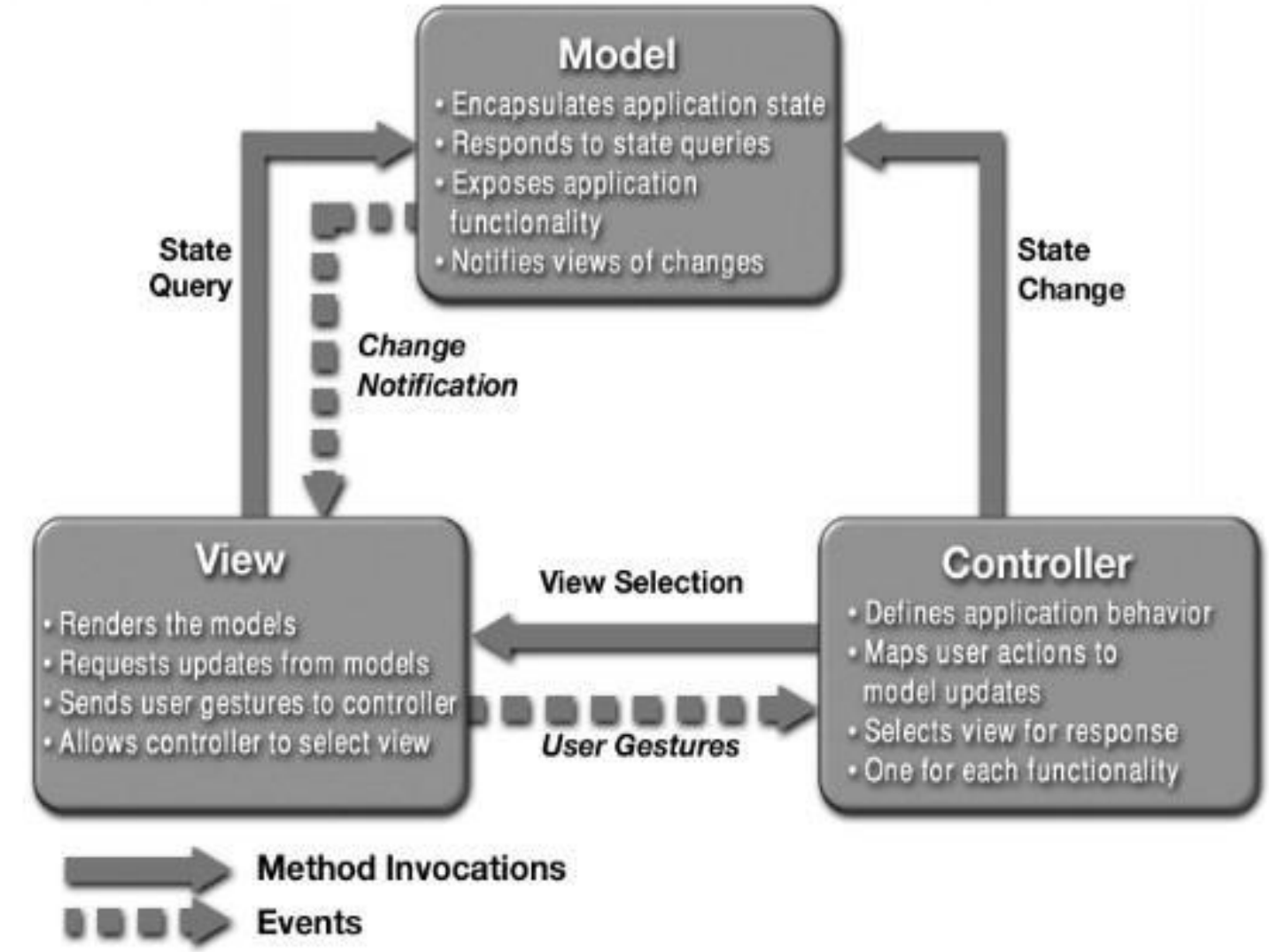
- Odděluje doménovou (bussiness) logiku a uživatelské rozhraní do tří nezávislých komponent:
  - **M**odel
  - **V**iew
  - **C**ontroller

- Velice populární, ale existují různé definice a variace
  - Detaily jsou častým předmětem diskuzí a sporů

MVC - Komponenty

- **M**odel
  - Představuje data a doménovou logiku
  - Zpravidla je pouze jeden na aplikaci
  - Je nezávislý na ostatních komponentách
- **V**iew
  - Představuje uživatelské rozhraní
  - Zobrazuje výstup uživateli
  - Často je více pohledů na jeden Model
- **C**ontroller
  - Aplikační logika
  - Obsluhuje a řídí vstupy uživatele
  - Často úzce provázán s View

MVC - Diagram

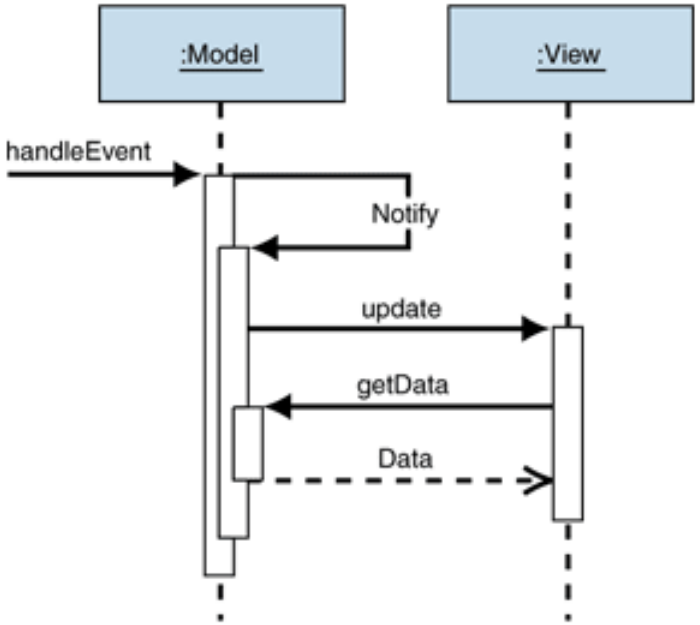

Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

MVC - Motivace

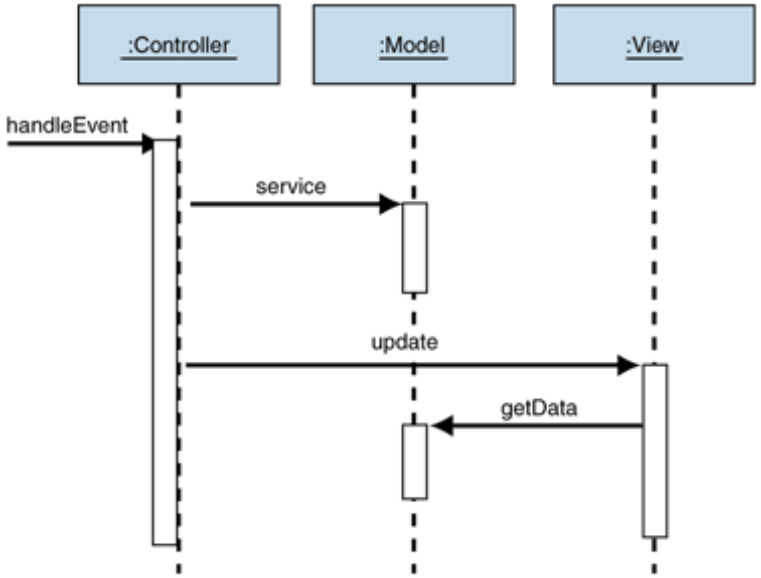
- **udržitelnost**
  - Vyšší přehlednost a pochopitelnost kódu
  - UI lze měnit za chodu aplikace a nezávisle na Modelu
  - Za každou komponentu může zodpovídat jiná osoba (nebo tým)
- **testovatelnost**
  - Každou komponentu lze testovat a debugovat nezávisle na ostatních
- **Zamezení duplikace kódu**
  - Více pohledů na jedna data (jeden Model)

MVC - Verze

- **Aktivní**
  - **Model** upozorňuje příslušné pohledy na svou změnu
  - Realizováno návrhovým vzorem Observer



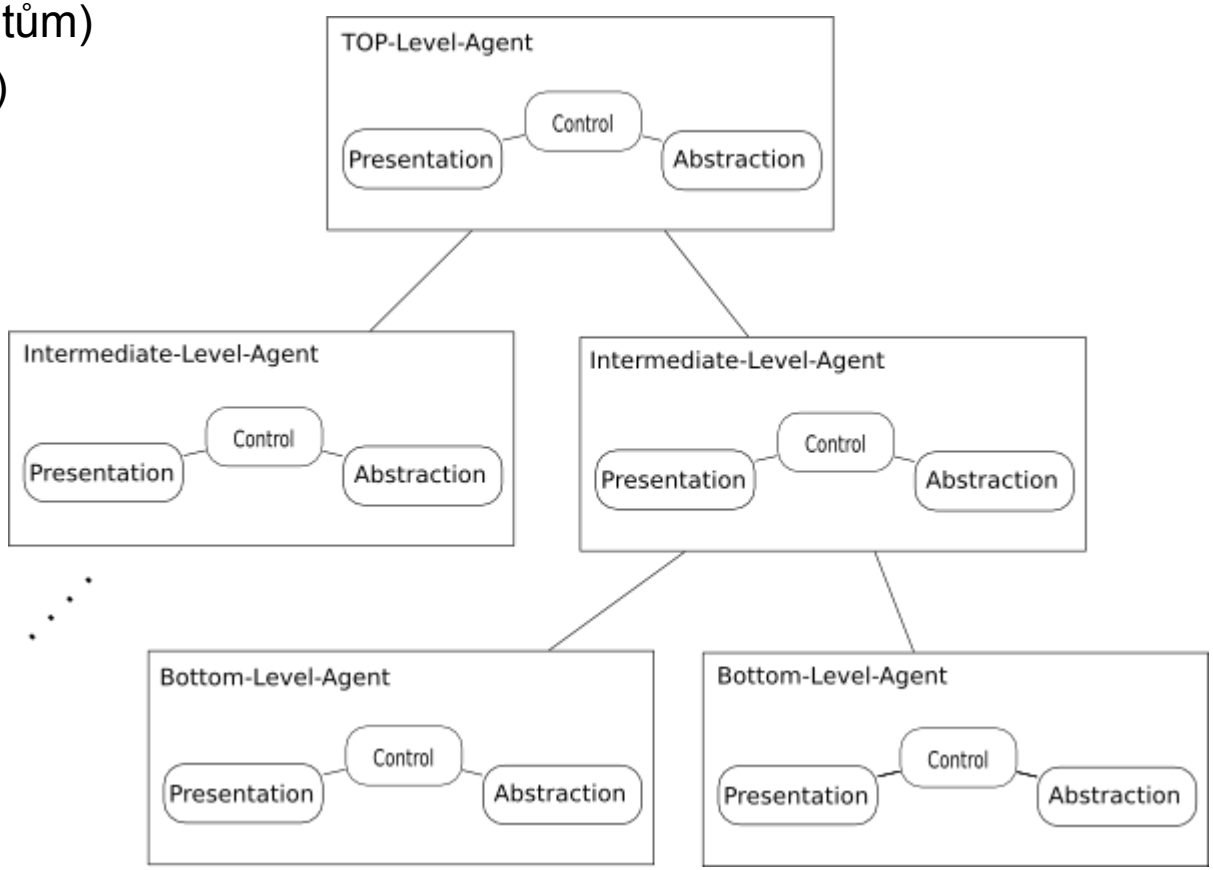
- **Pasivní**
  - **Controller** upozorňuje příslušné pohledy na změny modelu, které vyvolal



PAC (Presentation-Abstraction-Control)

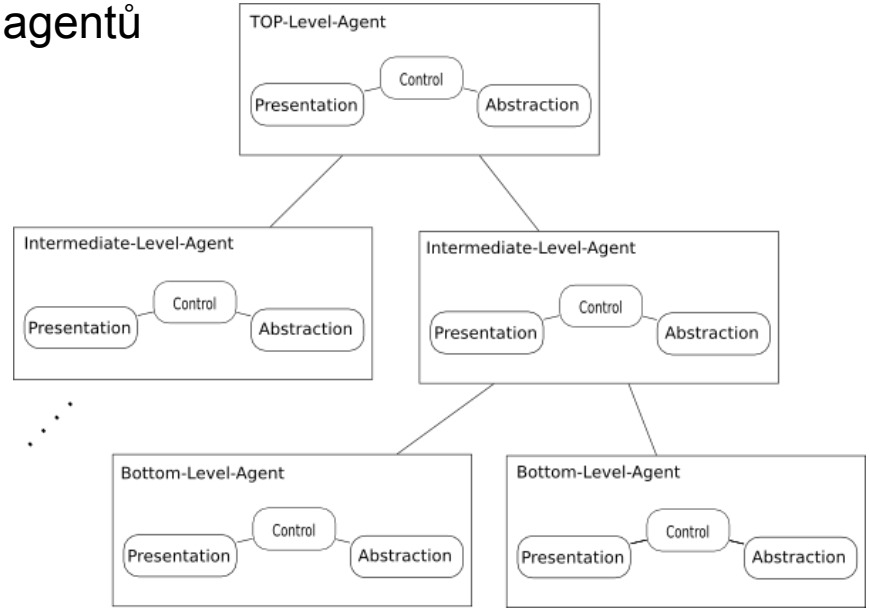
- Hierarchie spolupracujících agentů

- Každý agent má tři komponenty:
  - **P**resentation (view)
  - **A**bstraction (přístup k datům)
  - **C**ontrol (aplikační logika)



PAC – Komponenty agenta

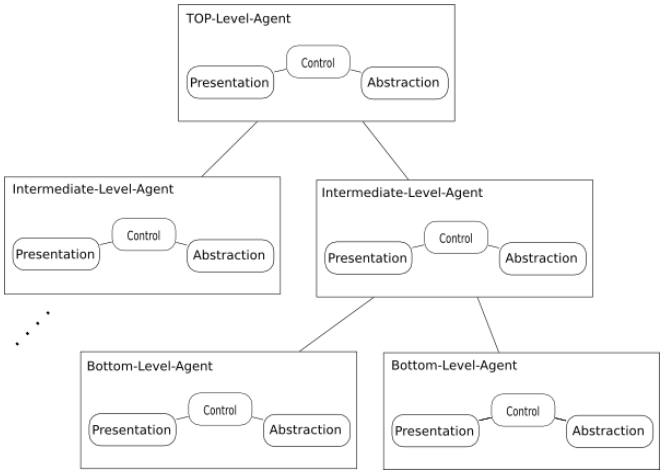
- **P**resentation
  - Viditelné rozhraní
  - Komunikuje pouze s Control
- **A**bstraction
  - Spravuje datový model a přistupuje do něj
  - Komunikuje pouze s Control
- **C**ontrol
  - Komunikuje s Abstraction a Presentation
  - Dále také komunikuje s Control ostatních agentů



PAC – Agenti (1)

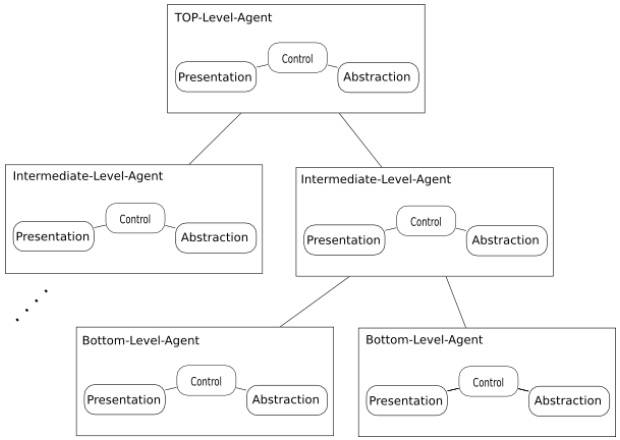
- Každý agent řeší specifický aspekt funkcionality aplikace

- Tři druhy agentů
  - **Top-level**
    - •Jádro systému
    - •Z UI má ty komponenty, které se nedají zařadit do podsystémů – např. dialogy či menu
  - **Bottom-level**
    - •Určitý sémantický koncept aplikace (jedna funkce)
    - •Např. vykreslování grafu, včetně operací (zoom)
  - **Intermediate-level**
    - •Koordinace více pohledů na ta samá data
    - •Umožňuje otevírání a zavírání těchto view
    - •Přeposílá zprávy o změnách na datech



PAC – Agenti (2)

- Agenti nabízí jak **horizontální** (na vrstvy), tak **vertikální** (funkcionalita) **dekompozici** systému
- Agent může být něco o velikosti jednoho objektu, ale také celého systému
- Agenti mají vlastní stav
- Agenti by měli tvořit stromovou strukturu
  - Jeden Top-level agent
  - Více Intermediate-level agentů
  - Ještě více Bottom-level agentů
- Agenti níže jsou závislí na agentech výše až k Top-level



MVC vs PAC

MVC	PAC
Model	Abstraction
View + Controller	Presentation + Control
View obsluhuje výstup Controller obsluhuje vstup	Presentation obsluhuje vstup i výstup
View má určitou logiku (znalost modelu)	Presentation je zcela bez logiky (templatovací systém)
View přijímá data přímo z Modelu	Presenter přijímá data z Abstraction přes Control
	hierarchická, vrstevnatá architektura

MVC vs PAC

MVC výhody	PAC výhody
•Oddělení funkcionality od UI •Menší složitost systému •Výkonnost - menší režie •Oddělené zpracování vstupu a výstupu	•Oddělení funkcionality od UI •Snadno rozšiřitelné (přidání agenta) •Snadno upravitelné (úprava agenta) (Nevyžaduje změny po celém systému) •Podpora multitaskingu •Low coupling •Vhodné pro tvorbu složitých UI - složení z jednotlivých agentů bez porušení zapouzdření
MVC nevýhody	**PAC nevýhody**
•Provázanost Views a Modelu	•Větší složitost systému •Rychlost - komunikace agentů •Riziko rozpadu systému do příliš velkého množství agentů (pro které se navíc musí napsat další, koordinující agenti).

HMVC

- Hierarchical Model View Controller
- Hybrid MVC a PAC
- Hierarchická struktura jako u PAC
- Vnitřní struktura agentů jako u MVC

Zdroje

http://www.garfieldtech.com/blog/mvc-vs-pac
http://msdn.microsoft.com/en-us/library/ff649643.aspx
http://zdrojak.root.cz/clanky/uvod-do-architektury-mvc/
http://icpc.felk.cvut.cz/web/slides/y36ass/Prezentace/2_PresentationAbstractionControll.pdf
http://assets.cambridge.org/97805218/17332/sample/9780521817332ws.pdf
http://moosehead.cis.umassd.edu/cis390/slides/08_PAC.pdf

http://www.cs.ucy.ac.cy/courses/EPL603/Intro4.pdf
http://goo.gl/78KqP
http://kirankawalli.110mb.com/PatternExamples.pdf
http://en.wikipedia.org/wiki/Presentation-abstraction-control
http://en.wikipedia.org/wiki/Model–View–Controller