8. Softwarový návrh

1. Principy návrhu

- nejedná se o knihovnu či úsek zdrijového kódu, nýbrž o způsob popisu řešení, jak daný problém řešit
- můžeme si to představit jako šablonu
- v objektovém paradigmatu se návrhové vzory typycky ukazují na vztazích tříd a objektů

2. Vícevrstvá architektura aplikací

- aplikace, jež využívají vícebrstvou architekturu, jsou diverzifikované a netvoří celistvý program (monolitické programy)
- vícero spolupracujících pater
- příkladem může být internetový obchod, první patro je user interface, druhé patro tvoří vebový server a data, která jsou uložena v databázi, což je vrstva třetí
- nejznámější je třívrstvá
  - prezentační (grafické rozharnní, práce se vstupy a výstupy)
  - aplikační (funkce, práce s daty, jejich zpraování)
  - datová (webová služba, souborový systém)
- **MVC**: neboli Model-View-controller, model je reprezentnat datového patra, view je uživatelské rozhranní, controller je ovladač, hlavní jednotka, MVC je navrhnuto tak, aby modifikace měla minimální vliv na ostatní
- **MVP**: Model-View-Presenter, presenter je middle-man mezi modelem a uživatelskym rozhranním
- další: Model-view-viewmodel

3. Návrhové vzory

- **A) Creational Patterns**:
  - **Abstract factory**: umožňuje vytvořit rodiny příbuzných objektů, aniž bychom specifikovali jejich konkrétní třídu, uživatel používá už jednotlivá rozhranní abstraktních továren, nespecifikujeme konkrétní třídy každého objektu
  - **Builder**: složité objekty vytvářet krok po kroku, jiné reprezentace stejnou posloupností kroků
  - **Singleton**: každá třída má pouze jednu instanci
- **B) Structural Patterns**:
  - **Adapter**: umožňuje, aby si jinak nekompatabilní dvojice objektu a prostředí bylo kompatabilní
  - **Composite**: stromová kompozitní struktura pro rozličné objekty, takže to umožňuje lepší orientaci, jeden interface může být obtížný
  - **Decorator**: pomocí wrapper skriptu můžu měnit chování objektu, přidávám nové funkce, wrapper stack; na sobě závislé wrapper kódy, problém, když chceme nějaký odstranit, tak musíme předělat strukturu, závisí na pořadí wrapper stacku
- **C) Behavioral Patterns**:
  - **Iterator**: umožňuje procházet prvky bez znalosti jejich implementace

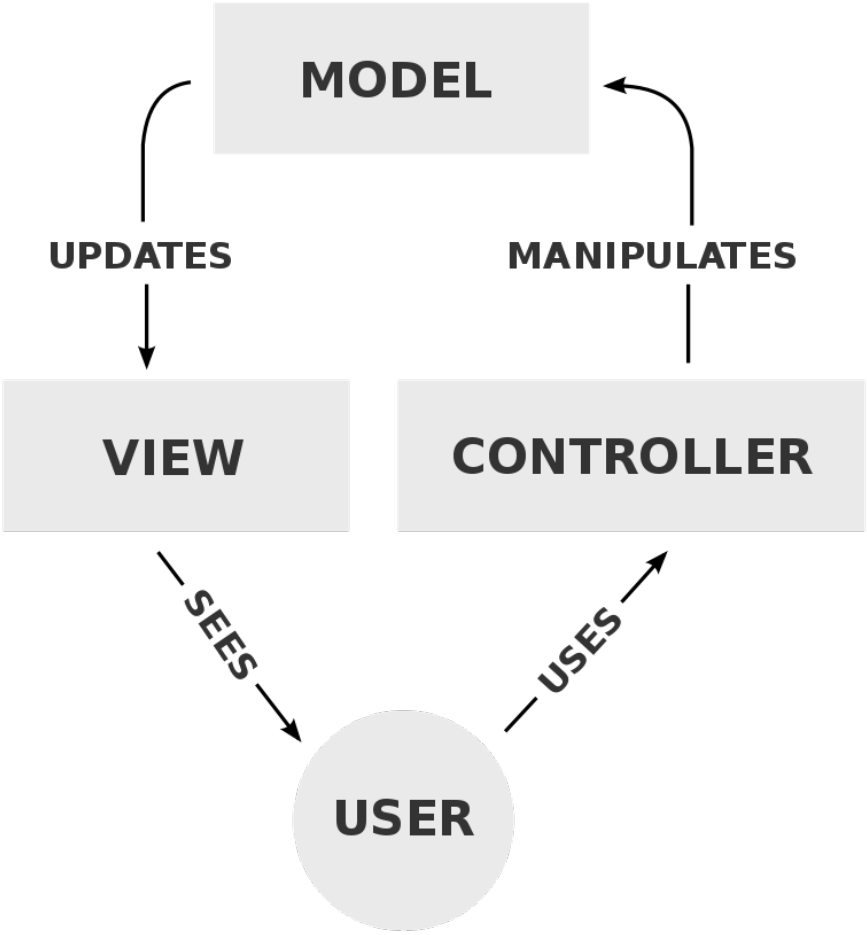

Figure 1: MVC

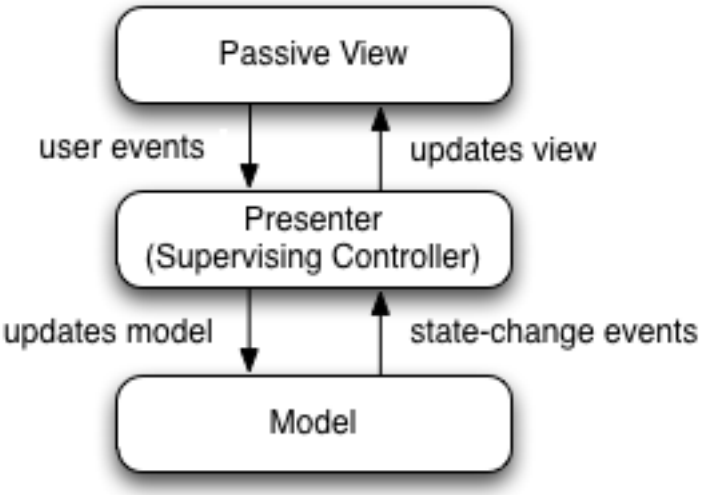

Figure 2: MVP

– **Mediator**: umožňuje zajistit komunikaci mezi dvěma komponentami programu, aniž by byly v přímé interakci, a tím musely přesně znát poskytované metody