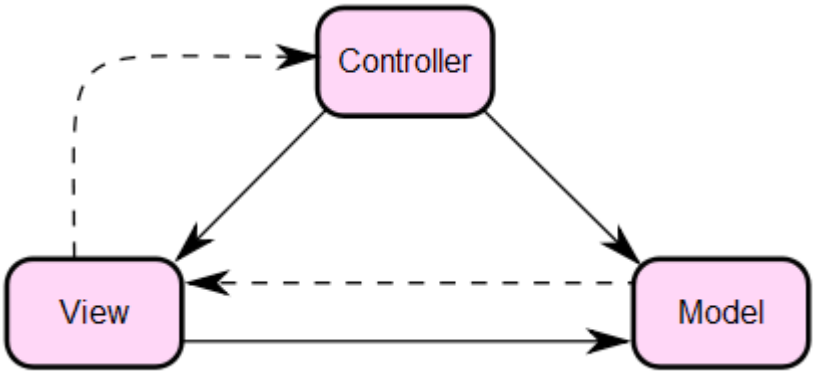
Návrhové vzory

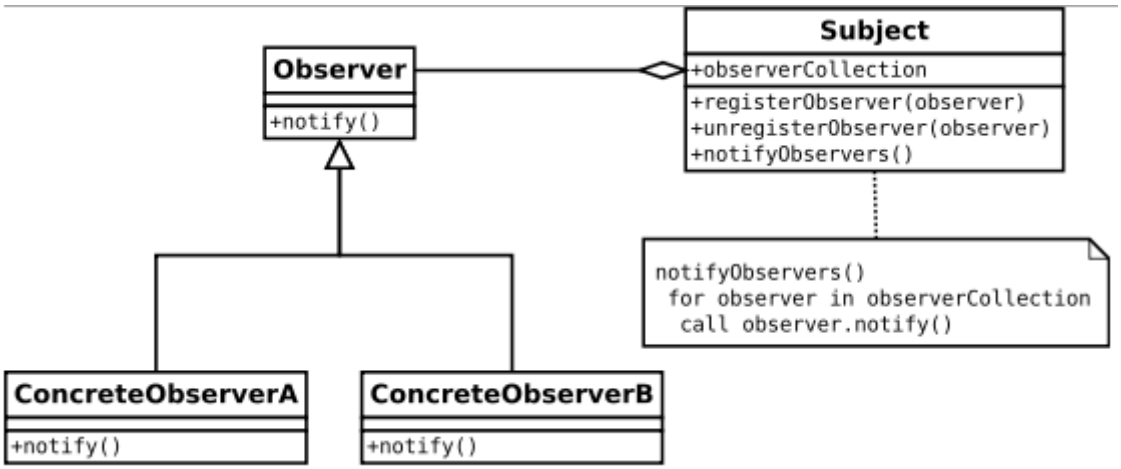
Návrhové vzory tvoria neoddeliteľnú súčasť objektového návrhu software. Predstavujú tzv. „best practise“ alebo časom overené najvhodnejšie riešenie často sa opakujúcich problémov. Naša desktopová aplikácia zameraná na vytváranie meta modelov je celá postavená na koncepte **MVC** (Model View Controller), ktorý jednoznačne rodeľuje celú aplikáciu na 3 časti, ktoré navzájom medzi sebou komunikujú. Každá časť obsahuje objekty, ktoré plnia špecifickú úlohu.



- **Model** – stará sa o aplikačné dáta, v našom prípade spravuje jednotlivé meta modely vytvorené užívateľmi, pridávanie nových tvarov a spojení, ich serializovanie.
- **Controller** – kontroluje komunikáciu medzi modelom a view, prijíma vstupy od view a vykoná príslušné zmeny v modeli, logicky oddeľuje model a view.
- **View** – predstavuje prezentačnú vrstvu aplikácie, v našej aplikácii je to grafické užívateľské prostredie, v ktorom si užívateľ pomocou grafickej reprezentácie modeluje jednotlivé modely.

V metamodelári sme si túto architektúru trochu upravili, exkluzívne sme oddelili časti view a model a umožnili ich vzájomnú komunikáciu jeden cez controller. Táto zmena nám umožňuje kedykoľvek upraviť, prípadne úplne vymeniť model alebo view, keďže sú od seba striktne oddelené.

Toto rozvrstvenie aplikácie na nezávislé vrstvy vyžaduje nejaký mechanizmus na vzájomnú notifikáciu tried o rôznych udalostiach. Jedna možnosť je vytvoriť medzi nimi silné väzby (tight coupling), ale tento prístup znemožňuje neskoršie znovupoužitie existujúcich tried. Ďalším riešením, ktoré sme využili aj my je návrhový vzor **Observer**. Tvorí dôležitú súčasť MVC aplikácií, pretože realizuje vzťahy typu 1..n, kde sa u jedného subjektu zaregistrujú viacerí observeri, ktorý sú automaticky notifikovaný pri vzniku určitej udalosti. Rozlišuje dva druhy objektov – subjekt a observer.



- **Subject** – obsahuje zoznam všetkých zaregistrovaných observerov, umožňuje ich registráciu a odhlásenie.
- **Observer** – definuje interface pre objekty, ktoré majú byť notifikované ak nastane uvažovaná udalosť

A nakoniec jeden veľmi jednoduchý, ale nesmierne užitočný návrhový vzor, ktorý sme použili pri návrhu – **Iterator**. Slúži na jednoduchý prístup k elementom objektu, ktorý agreguje veľké množstvo iných objektov, napr. kolekcia, pri tom neodkrýva jeho vnútornú štruktúru. Povaha našej aplikácie vyžaduje časté prechádzanie rôznych listov a iných podobných štruktúr, ktoré reprezentujú model. Každý iterátor má uniformný interface, ktorý obsahuje základné metódy na prácu s kolekciou – next(), ktorá vracia nasledujúci prvok kolekcie a hasNext(), ktorá vracia informáciu o tom, či existuje ďalší prvok.

Použitá literatúra

GAMMA, Erich, et al. *Design patterns : elements of reusable object-oriented software*. [s.l.] : [s.n.], 1995. 395 s. ISBN 9780201633610.