

# Ročníkový projekt

Štěpán Šindelář

9. dubna 2010

## 1 Stručná specifikace

Základem práce je myšlenka, že by programátor mohl do zdrojového kódu pomocí syntaktických prostředků daného jazyka přidat dokumentaci použitých vzorů. Pojmem vzory jsou zde myšleny klasické návrhové vzory popsané v [1] a vzory popsané v [2], dále rozvinuté v [3] jako součást postupu nazvaného *Domain-driven design*.

Protože této dokumentace by bylo dosaženo pomocí syntaktických prostředků, bylo by snazší ji automatizovaně zpracovat a navíc by se taková dokumentace stala součástí zdrojového kódu, což by mohlo programátory motivovat k jejím změnám v průběhu změn v samotném kódu.

Cílovou technologií bude platforma .NET. Jako syntaktický prostředek se tedy nabízí použít atributy, kterými lze odekorirovat třídy i jejich metody nebo vlastnosti.

Součástí práce by byly následující dvě aplikace.

### Grafický nástroj pro objektové modelování.

V podstatě obdoba UML class diagramu, ale na rozdíl od UML by byly zohledněny i návrhové vzory. Tento nástroj bude umožňovat navrhnout objektový model i použité vzory a pak podle takového diagramu vygenerovat zdrojový kód v jazyce C#.

Protože tento vygenerovaný kód odekorirovuje správnými atributy, bude pak možné z existujícího kódu znovu vyextrahovat zpět grafickou reprezentaci. Uživatel by ji mohl upravit a poté své změny nechat promítnout zpět do zdrojového kódu.

Hlavním přínosem by měla být možnost modelovat i v *Domain-driven design* pojmech (vzorech), a tak umožnit určitou míru abstrakce v situaci, kdy je diagram vygenerovaný ze zdrojových kódů a obsahuje velké množství dat – např. pomocných tříd.

S *Domain-driven design* souvisí i poslední funkcionalita. Tou bude možnost generovat z diagramu nejen zdrojový kód, ale i konfiguraci pro vybrané ORM nástroje.

### Kompilér vynucující správnou implementaci vzorů.

Dokumentaci použitých vzorů lze použít i k automatizované kontrole toho, zda dané třídy opravdu implementují uvedený vzor. Tato aplikace je inspirována projektem Pattern Enforcing Compiler (PEC) For Java [4].

Na rozdíl od PEC by měl tento nástroj podporovat navíc i některé vzory z [2] a [3]. Na druhou stranu nebude podporovat všechny „klasické“ vzory, které PEC podporuje.

V otázce výběru konkrétních vzorů, jejichž správnou implementaci by bylo možné nějakým způsobem automatizovaně zkontrolovat, je třeba ještě provést podrobnější zkoumání.

## 2 Plán práce

- Do konce dubna: podrobnější nastudování uvedených zdrojů.
- Do půlky května:
  - detailní specifikace podporovaných vzorů,
  - návrh systému atributů pro jejich označování v kódu,
  - výběr vzorů vhodných pro automatizovanou kontrolu a popis způsobu této kontroly pro jednotlivé vzory.
- Do konce května:
  - návrh architektury pro společné jádro obou nástrojů,
  - podrobná specifikace funkcí obou nástrojů.
- Do konce června: návrh architektury pro „kompilátor“.
- Do září:
  - implementace „kompilátoru“,
  - návrh architektury grafického modelovacího nástroje,
  - prototyp GUI grafického modelovacího nástroje.
- V průběhu ZS: implementace grafického modelovacího nástroje.

## Reference

- [1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-wesley Reading, MA, 1995.
- [2] M. Fowler, *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2002.
- [3] E. Evans, *Domain-driven design: tackling complexity in the heart of software*. Longman, 2004.
- [4] H. Lovatt, “Pattern enforcing compiler for java.” <https://pec.dev.java.net/>, May 2010.