CODE GENERATION USING DESIGN PATTERNS

František Hanák

Master Degree Programme (3), FIT BUT

E-mail: xhanak01@stud.fit.vutbr.cz

Supervised by: Peter Jurnečka

E-mail: ijurnecka@fit.vutbr.cz

Abstract: This paper describes code generation using design patterns. It deals with questions of specification of design patterns and their usage in code generation. It follows formal and unformal design patterns specifications and describes algorithm for searching similar structures of patterns in the source code.

Keywords: design patterns, formal definition, SPINE, BPSL, GEBNF, code generation, CodeDom, .NET

1. ÚVOD

Tím jak se mění doba a informační technologie jsou čím dál vyspělejší, stávají se běžnou součástí našeho života. Čím dál větší důraz je kladen na efektivnost tvorby systému a jeho znovu použitelnost, a proto se při návrhu a vývoji hojně využívají návrhové vzory. V současné době se používají neformální specifikace vzorů, které ale mají jistá úskalí, a proto se často doplňují formálními specifikacemi.

Tato práce se zabývá problematikou jednoznačnosti definice vzorů, vysvětluje možnosti formální specifikace návrhových vzorů, porovnává jednotlivé způsoby reprezentace vzorů z hlediska využitelnosti při generování kódu aplikací a popisuje návrh algoritmu pro vyhledání podobných struktur vzoru ve zdrojovém kódu a jejich modifikaci za účelem realizace vzoru.

2. NÁVRHOVÉ VZORY

Návrhové vzory (Design Patterns) jsou doporučené postupy a osvědčená řešení často nebo opakovaně se vyskytujících problémů v objektově orientovaném (OO) návrhu [1]. Tato řešení jsou obecného charakteru, nezávislá na architektuře či programovacím jazyku a zajišťují společně s OO návrhem znovu použitelnost komponent systému.

2.1. NEFORMÁLNÍ SPECIFIKACE NÁVRHOVÝCH VZORŮ

Neformální definice popisují jak strukturu vzoru, tak i chování. Jejich výhodou je, že jsou jednoduché na pochopení a na první pohled srozumitelné. Zásadní nevýhodou je, že neumožnují automatizovanou verifikaci vzoru a mohou být nejednoznačné [3].

2.2. FORMÁLNÍ SPECIFIKACE NÁVRHOVÝCH VZORŮ

Cílem formální specifikace návrhových vzorů je odstranit nevýhody neformálních popisů, zajistit jejich precizní a jednoznačný popis a umožnit jejich automatizovanou verifikaci. Formální specifikace nenahrazují neformální, pouze je doplňují a dělí se do dvou kategorií [3]. Do první kategorie patří jazyky vyvinuté konkrétně pro potřeby formálních zápisů vzorů. Patří sem například GEBNF (Graphic Bacus Naur Form), která umožňuje formálně definovat diagramy tříd, sekvence apod., díky čemuž se stává formálním doplňkem jazyka UML (více viz [2]). Jazyky druhé kategorie pouze vhodným způsobem kombinují a upravují již existující formální systémy. Významným zástupcem

je například jazyk BPSL (Balanced Pattern Specification Language), který používá kombinaci predikátové logiky 1. řádu pro popis struktury a temporální logiku akce pro popis chování (více viz [2]). Výhodou druhé kategorie je, tyto systémy již existují dlouho a díky tomu je i podpora nástrojů pro automatizovanou verifikaci na vysoké úrovni. Zásadní nevýhodou ale je, že neumožňuje úplnou definici vzoru, tj. jak definici chování, tak i struktury. Dalšími často používanými systémy jsou SPINE nebo Prolog.

3. GENEROVÁNÍ KÓDŮ POMOCÍ NÁVRHOVÝCH VZORŮ

Pro generování kódů aplikací pomocí návrhových vzorů jsem navrhl algoritmus, který na základě definice vzoru vyhledá v již existujícím zdrojovém kódu vhodné entity, které modifikuje takovým způsobem, aby výsledný kód realizoval daný vzor.

3.1. DEFINICE VZORU

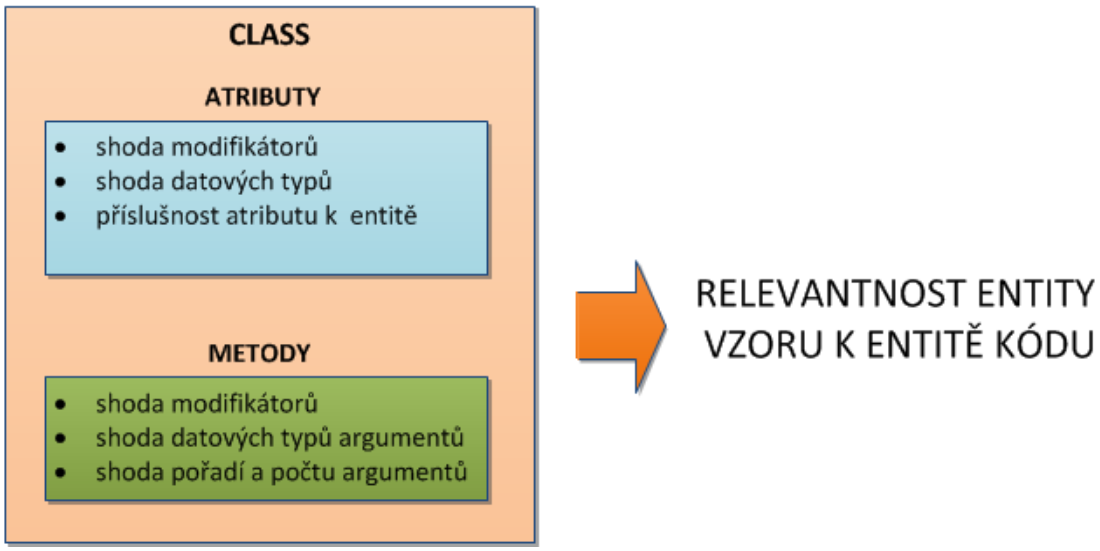
Jednou z podstatných částí algoritmu je i zpracování definice vzoru, kterou jsem navrhl ve formátu XML souboru, který je možné validovat vůči XSD dokumentu. Tím je zajištěna jednoduchá možnost validace definic vzorů a stejně tak rozšiřitelnost podpory aplikace o další vzory. V rámci popisu vzoru je možné definovat jednotlivé entity (rozhraní, třídy), jejich atributy, metody (včetně modifikátorů), ale i relace mezi entitami (dědičnost, asociace, realizace) včetně kardinalit. Zároveň je možné definovat i typ entit (nebo návratový typ metod) v abstraktním pojetí.

3.2. NALEZENÍ VHODNÝCH ENTIT

Aby algoritmus mohl efektivně vyhledávat podobné struktury vzoru ve zdrojovém kódu, reprezentuje popis vzoru i načtený zdrojový kód stejnou datovou strukturou. Nejprve je načtena definice vzoru a provedena její validace. Poté jsou načteny soubory se zdrojovým kódem. Vzor i kód je následně převeden do uniformní datové struktury (oba jsou reprezentovány jako abstraktní syntaktický strom). Algoritmus pak načte jednotlivé entity vzoru a vyhledá k nim relevantní entity zdrojového kódu. Výsledkem tedy je seznam vhodných entit zdrojového kódu ke každé entitě vzoru.

3.3. OHODNOCENÍ NALEZENÝCH ENTIT

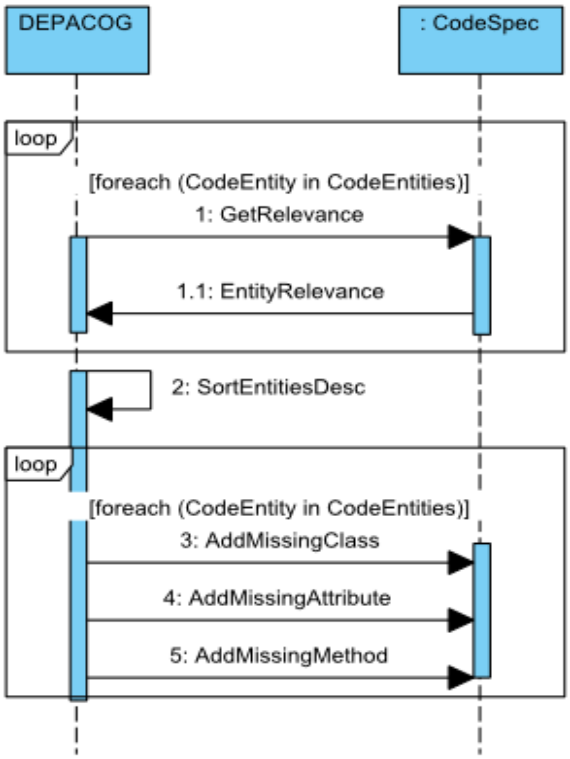
Algoritmus nalezené entity ohodnotí relevantností, aby pro modifikaci zdrojových kódů vybral jen ty nejvhodnější. Ohodnocení je provedeno na základě následujících kritérií, kde každé splnění daného kritéria zvyšuje relevantnost dané entity kódu.



Obrázek 1: Způsob určení relevantnosti entity vzoru.

3.4. NALEZENÍ ROZDÍLU MEZI ENTITAMI

Neméně důležitou částí algoritmu je i nalezení rozdílu mezi entitou vzoru a kódu, aby algoritmus poznal, které položky je třeba dogenerovat. Algoritmus se pokusí "namapovat" jednotlivé atributy a metody entity vzoru na nejvíce relevantní entitu kódu. K tomu využívá ohodnocení atributů a metod (popsáno výše).



Obrázek 2: Sekvenční diagram ohodnocení entit.

4. ZÁVĚR

Tento příspěvek popsal možnosti specifikace návrhových vzorů a jejich srovnání z hlediska využitelnosti při automatizovaném generování kódu, možnost multiplatformní specifikace návrhového vzoru, která usnadňuje validaci, a návrh algoritmu, který na základě definice vzoru vyhledá podobné struktury ve zdrojovém kódu a modifikuje ho takovým způsobem, aby realizoval daný vzor.

Článek vznikl jako součást projektu GA CR 102/09/H042 Matematické a inženýrské metody pro vývoj spolehlivých a bezpečných paralelních a distribuovaných počítačových systémů a FIT-S-11-1 Pokročilé bezpečné, spolehlivé a adaptivní IT.

REFERENCE

[1] Gamma, E., Helm, R., Johnson, R., Vlissides, R.: Design Patterns - Elements of Reusable Object Oriented Software. Addison-Wesley, New York, 2004. ISBN 0201633612

[2] Taibi, T., Ngo, D.C.L.: Formal Specification of Design Patterns – A Balanced Approach [online], Journal of Object Technology, 2003. [cit. 2011-05-11]. Dostupné na URL: http://www.jot.fm/issues/issue_2003_07/article4

[3] Taibi, T.: Design Patterns Formalization Techniques: Formal specification and verification of design patterns. [online], London, 2007. ISBN: 978-1-59904-221-3. [cit. 2011-05-11]. Dostupné na URL: http://www.google.cz/books?id=lQH6Ejk9OwUC&lpg=PA1 &hl=cs&pg=PA1#v=onepage&q&f=true