AUTOMATIZACE TRANSFORMACÍ OBJEKTOVĚ ORIENTOVANÝCH MODELŮ POMOCÍ APARÁTU SBAT

Oldřich Nouza
ČVUT v Praze, Fakulta jaderná a fyzikálně inženýrská
Břehová 7, 115 19 Praha 1
nouza@km1.fjfi.cvut.cz

ABSTRAKT:
This paper deals with automated transformations of object-oriented models using the SBAT transformation engine. The object-oriented model is formally described as a state space by the first-order predicate logic calculus, where the initial state represents the source model of the transformation, and the goal state the target one. The SBAT engine is designed to support automation of refactoring, design pattern application and object normalization.

KLÍČOVÁ SLOVA:
automatizace, modelové transformace, objektový model, refaktoring, SBAT, STRIPS

ÚVOD

Transformace je stěžejním tématem v informatice a softwarovém inženýrství. Ačkoliv existují uspokojivá řešení transformací modelů na text, nelze totéž prohlásit o transformacích modelů na jiné modely. V posledních letech bylo navrženo mnoho nových přístupů, avšak relativně málo zkušeností neumožňuje posouzení jejich efektivity v praktickém použití. Tato oblast je tedy stále ve fázi výzkumu [1].
Modelové transformace podporované dnešními CASE nástroji jsou založeny na aplikaci příslušného transformačního pravidla s odpovídajícími parametry [2], [3], [5], [12]. Tento přístup má ovšem jednu podstatnou nevýhodu, a sice malou opětovnou použitelnost [3]. Pokud požadovaná transformace dosud nemá definováno transformační pravidlo, je potřeba jej složit z pravidel jiných, popřípadě dodefinovat. Z tohoto důvodu byl navržen nový transformační aparát SBAT (STRIPS Based Automated Transformations = automatické transformace založené na STRIPS), který tyto kroky nevyžaduje.

MODELOVÉ TRANSFORMACE

Existuje několik definic pojmu *modelová transformace*. Publikace [7] s odvoláním na [4] jej definuje následovně:

> Transformace *je automatické generování cílového modelu ze zdrojového modelu podle definice transformace.* Definice transformace *je množina transformačních pravidel, která popisují, jak může být model ve zdrojovém jazyce transformován na model v cílovém jazyce.* Transformační pravidlo *je popis, jak jedna či více konstrukcí ve zdrojovém jazyce může být transformována na jednu čí více konstrukcí v cílovém jazyce.*

V publikaci [7] jsou modelové transformace klasifikovány podle dvou kritérií:

1. Jazyky použité pro vyjádření modelů
    - *Endogenní transformace* – Zdrojový a cílový model jsou vyjádřeny ve stejném jazyce. Příklady tohoto typu transformace jsou optimalizace, refaktoring, zjednodušení či normalizace.

- *Exogenní transformace* – Zdrojový a cílový model jsou vyjádřeny v různých jazycích. Příklady tohoto typu transformace jsou syntéza (transformace modelu vyšší úrovně abstrakce na model nižší úrovně abstrakce), reverse engineering (inverze syntézy), či migrace mezi modely v různých jazycích, avšak na stejné úrovni abstrakce.

2. Úroveň abstrakce modelů
   - *Horizontální transformace* – Zdrojový a cílový model mají shodnou úroveň abstrakce. Typickým příkladem je refaktoring.
   - *Vertikální transformace* – Zdrojový a cílový model mají různé úrovně abstrakce. Typickým příkladem je refinement, kde specifikace postupnými kroky, které ji zpřesňují, přechází v implementaci.

REFAKTORING

Refaktoring lze chápat jako proces vylepšení struktury softwarového systému beze změny jeho chování [6]. Jinými slovy to znamená, že refaktorovaný software pro stejný vstup vrací stejný výstup jako software původní
Existuje několik způsobů, jak refaktoring popsat. Jedna z možností je vyjádřit refaktoring jako kompozici primitivních, čili dále nedělitelných refaktoringových operací. Tato myšlenka byla použita v [10] pro formální popis refaktoringu zdrojových kódů jazyka C++ a později v [11] k ukázce refaktoringu modelů v jazyce UML. Z tohoto přístupu rovněž vychází návrh transformačního aparátu SBAT.

PLÁNOVACÍ SYSTÉM STRIPS

Technická zpráva [8] vydaná výzkumným institutem Standford Research Institute uvádí:

> *STRIPS (STanford Research Institute Problem Solver) patří mezi systémy pro řešení problému, které prohledávají prostor modelů světa se záměrem najít takový, který odpovídá danému cíli. Předpokládáme, že pro každý model světa existuje množina operátorů, z nichž každý tento model světa transformuje na jiný. Úloha spočívá v nalezení posloupnosti operátorů, které postupně transformují počáteční model světa na takový, jenž vyhovuje předepsané cílové podmínce.*

Formálně lze úlohu STRIPS definovat následujícími definicemi.

Definice 1. Úloha STRIPS je uspořádaná trojice (I, O, G), kde I je počáteční stav, O je množina operátorů a G je podmínka cílového stavu.

Definice 2. Operátor o(x) je definován jako uspořádaná trojice (P, A, D), kde $P(\bar{x})$ je podmínka aplikovatelnosti operátoru, $A = [A_1(\bar{x}), \ldots, A_n(\bar{x})]$ je množina formulí, které se po aplikaci operátoru stanou pravdivými, $D = [D_1(\bar{x}), \ldots, D_n(\bar{x})]$ je množina formulí, které po aplikaci operátoru přestanou být pravdivými, $\bar{x} = (x_1, \ldots, x_n)$ jsou volné proměnné vyskytující se ve formulích $P, A_1, \ldots, A_n, D_1, \ldots D_n$. Prvky množiny A se nazývají add-efekty, prvky množiny D delete-efekty.

Definice 3. Nechť je dán operátor $o1(x_1, \ldots, x_n) = (P, A, D) \in O$. Přechodová funkce $o' : (x_1 \times \ldots \times x_n \times S) \to S$, kde S je množina stavů, je definována následovně:

$$o'(x_1,\ldots,x_n,s)=\begin{array}{ll}(s\cup A)-D & \varnothing\\ P\ \textit{je splněna v}\ s & \end{array}$$

Jinými slovy, platí-li ve stavu s podmínka P, hodnota přechodové funkce bude stav obsahující formule, které se vyskytují ve stavu s nebo množině A, ale nevyskytují se v množině D.
Úkolem je najít takový seznam aplikací operátorů, které způsobí přechod z počátečního stavu do stavu, který splňuje podmínku stavu cílového. Formálně řečeno:

Definice 4. Řekneme, že stav s_m je řešením úlohy (I,O,G), jestliže existuje seznam aplikací operátorů $o_1(\overline{h}_1),\ldots,o_m(\overline{h}_m)$, kde $\overline{h}_i$ jsou vektory konstant a $m\in\mathbf{N}$, a platí:

- $s_0 = I$
- $(\forall i\in[1,\ldots,m])s_i = o'_{i-1}(\overline{h}_{i-1},s_{i-1})$
- G je splněna ve stavu s_m

Jestliže I je splněna v G, pak řešení, které je v tomto případě I, nazýváme triviálním řešením.

TRANSFORMAČNÍ APARÁT SBAT

Na základě současného stavu technik automatické modelové transformace zmíněného v úvodu jsme vytvořili návrh nového transformační aparátu splňujícího následující požadavky:

- Aparát bude podporovat následující typy transformací: refaktoring, použití návrhového vzoru, objektová normalizace.
- Vstupem bude zdrojový model a podmínky, které musí splňovat cílový model.
- Výstupem bude cílový model, který vyhovuje podmínkám zadaným na vstupu, popřípadě informace, že žádný takový model se nepodařilo nalézt.
- Zdrojový model bude s cílovým modelem konzistentní z hlediska chování modelovaného systému.
- Proces transformace zdrojového modelu na cílový bude automatizovaný, bez nutnosti specifikace transformačních pravidel na vstupu.
- Aparát bude univerzální, čili použitelný pro širokou skupinu objektových modelů.

Formální definice objektového modelu pro účely aparátu SBAT vychází z metamodelu tříd jazyka UML, popsaného podrobně např. v [9]. Vzhledem k požadované obecnosti se nebudeme zaměřovat na implementační detaily jako parametry a těla metod, viditelnost složek tříd apod.
Při formalizaci modelových transformací vyjdeme z výše zmíněné myšlenky skládání refaktoringu z konečného počtu primitivních refaktoringů. Model definujeme jako stavový prostor, kde množina stavů představuje model všechny možné stavy modelu a množina přechodových funkcí množinu všech primitivních refaktoringů.

Definice 5. Nechť C je univerzum tříd, A univerzum atributů a F univerzum metod. Nechť existují třídy $Client, Object\in C$, kde $(\forall x\in C-[Object])(Object\prec x)$, čili třída $Object$ je předkem všech ostatních tříd, a $Client$ představuje klientskou třídu, která posílá zprávy všem objektům v modelu. Nechť ε představuje prázdnou hodnotu. Stav modelu je uspořádaná pětice $(C_s, in_s, super_s, types_s, send_s)$, kde

- $C_s \subset C - [Object, Client]$ je konečná množina tříd modelu ve stavu s,
- $in_s \subseteq (A \cup F) \times C_s$ je binární relace nazývaná „je ve třídě“ definovaná předpisem $in_s = \{(x, y) | x \in (Attr(y) \cup Meth(y)) \wedge y \in C_s\}$, kde $Attr(y)$ je množina atributů třídy y a $Meth(y)$ je množina metod třídy y,
- $super_s \subseteq (C_s \cup [Object]) \times C_s$ je binární relace nazývaná „je nadtřídou“ definovaná předpisem $super_s = \{(x, y) | x = super(y) \wedge x, y \in C_s\}$, kde $super(y)$ znamená „nadtřída třídy y“,
- $type_s \subseteq (A \times C_s) \cup (F \times C_s)$ je binární relace nazývaná „je typu“ definována předpisem $type_s = \{(x, y) | y = type(x) \wedge y \in C_s\}$, kde $type(x)$ znamená „typ atributu x“ nebo „typ návratových hodnot metody x“,
- $send_s \subseteq (F \times (C_s \cup [Client]) \times (A \cup F) \times C_s)$ je 4-ární relace nazývaná „poslání zprávy“ definována předpisem $send_s = \{(x, y, u, v) | (x, y) \in in_s \wedge (\exists w)((u, w) \in in_s \wedge (u \prec w \vee u = w)) \wedge \langle x, \Lambda \rangle \in Meth(y)\}$, kde lambda-výraz Λ obsahuje poslání zprávy $o \triangleleft u$, kde o je instance třídy w.

Dále musí být splněny následující podmínky:

- každý atribut se v hierarchii tříd může vyskytovat nejvýše jednou, čili $(\forall x \in A)(\forall y, z \in C)(in_s(x, y) \wedge in_s(x, z) \rightarrow \neg(y \prec z) \wedge \neg(z \prec y))$,
- každý atribut je nějakého typu, čili $(\forall x \in A)(\exists y \in C_s)(type_s(x, y))$,
- každý atribut je nejvýše jednoho typu a každá metoda vrací nejvýše jeden typ návratové hodnoty, čili $(\forall x \in (A \cup F))(\forall y, z \in C_s)(type_s(x, y) \wedge type_s(x, z) \rightarrow y = z)$.

Definice 6. Model je stavový prostor (M, Φ), kde M je konečná množina stavů a $\Phi = \bigcup_{i=1}^{n} [\varphi_i : M \times E^{k_i} \rightarrow M]$ je množina transformačních pravidel.

Předpokladem každé modelové transformace jsou odpovědi na následující otázky:

1. Co je třeba transformovat?
2. Jaký bude výsledek transformace?

K nalezení odpovědí je potřeba formulovat transformační úlohu a určit princip řešení. Jedním z možných prostředků k dosažení tohoto cíle je aplikace plánovacího systému STRIPS. Předpokládejme konečnou podmnožinu B univerza objektů, která obsahuje elementy všech možných stavů modelu. K formulaci úlohy STRIPS (I, O, G) pro modelové transformace je potřeba pospat stavy modelu a transformační pravidla pomocí kalkulu predikátové logiky prvního řádu. Pro tento účel definujeme predikáty popsané v tabulce.

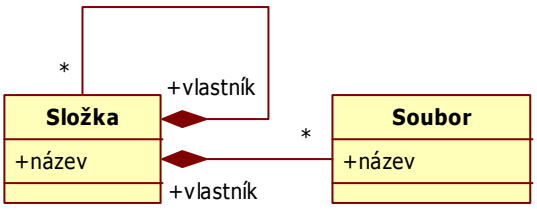
Tabulka 1. – Predikáty pro formulaci úlohy STRIPS

Predikát	**Deklarace**	**Definice**
třída c je v modelu	$inModel(c)$	$c \in C_s$
třída c není v modelu	$outOfModel(c)$	$c \in (B - C_s)$
atribut a je ve třídě c	$attrInClass(a, c)$	$(a, c) \in in_s$

atribut a není ve třídě c	$attrOutOfClass(a,c)$	$\neg((a,c)\in in_s)\wedge a\in(B\cap A)$
metoda μ je ve třídě c	$methInClass(a,c)$	$(\mu,c)\in in_s$
metoda μ není ve třídě c	$methOutOfClass(a,c)$	$\neg((\mu,c)\in in_s)\wedge \mu\in(B\cap F)$
atribut a je typu t	$hasType(a,t)$	$(a,t)\in type_s$
metoda μ vrací hodnoty typu t	$has\,\mathrm{Re}tType(\mu,t)$	$(\mu,t)\in type_s$
třída c je nadtřída třídy d	$superClass(c,d)$	$(c,d)\in super_s$
třída c je předkem třídy d	$parentClass(c,d)$	$c\prec d$
metoda μ třídy c posílá zprávu η objektům třídy d	$sending(\mu,c,\eta,d)$	$(\mu,c,\eta,d)\in sending_s$

Definovaná množina predikátů lze použít pro popis libovolného stavu modelu $m_s=(C_s,in_s,super_s,types_s,send_s)$, počátečního stavu I, podmínky koncového stavu G a množiny operátorů O, které představují primitivní refaktoringy. Vzhledem k jejímu velkému rozsahu ji zde uvádět nebudeme, nicméně lze dokázat, že existuje a je konečná, úplná a minimální.

Příklad 1. Fungování aparátu SBAT ukážeme na příkladě. Uvažujme model tříd systému souborů (viz obr. 1).



Obr. 1. – Diagram tříd systému souborů

Cílem je transformovat tento model do stavu, který vyhovuje návrhovému vzoru *Composite*.

Řešení. Formální popis zdrojového modelu $m_s=(C_s,in_s,super_s,types_s,send_s)$ je následující:

- $Cs=[Složka,Soubor,String]$,
- $ins=[(název,Složka),(vlastník,Složka),(název,Soubor),(vlastník,Soubor)]$,
- $super_s=[(Object,Složka),(Object,Soubor),(Object,String)]$
- $type_s=[(návev,String),(vlastník,Složka)]$
- $send_s=\{(Client,main,x,y)|(x,y\in in_s)\}$

Tomu odpovídá počáteční stav úlohy STRIPS:

$$I = [inModel(Složka), inModel(Soubor), inModel(String),$$
$$outOfModel(Element),$$
$$attrInClass(název, Složka), attrInClass(vlastník, Složka),$$
$$attrInClass(název, Soubor), attrInClass(vlastník, Soubor),$$
$$attrOutOfClass(název, Element), attrOutOfClass(vlastník, Element),$$
$$superClass(Object, Soubor), superClass(Object, Složka),$$
$$hasType(vlastník, Složka), hasType(název, String),$$
$$sending(Client, main, Složka, název), sending(Client, main, Složka, vlastník),$$
$$sending(Client, main, Soubor, název), sending(Client, main, Soubor, vlastník)]$$

Formulace podmínky cílového stavu je následující:

$$G = inModel(Element) \wedge attrInClass(název, Element) \wedge attrInClass(vlastník, Element) \wedge$$
$$\sup erClass(Element, Soubor), \sup erClass(Element, Složka)$$

Plánovací systém STRIPS najde posloupnost aplikací operátorů, které transformují počáteční stav I do stavu, který vyhovuje podmínce G. Seznam těchto aplikací operátorů obsahuje tabulka 2.

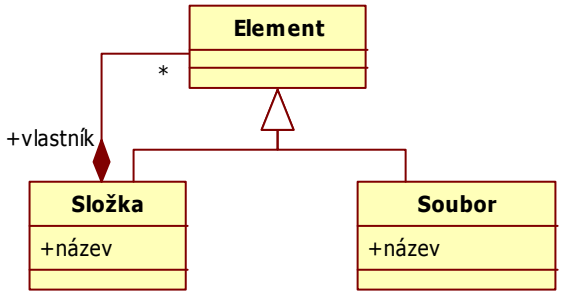
Tabulka 2. – Seznam aplikací operátorů k dosažení cílového stavu

Aplikace operátoru	**Popis**
$addClass(Element)$	přidání třídy *Element* do modelu
$changeSup(Soubor, Objekt, Element)$	změna nadtřídy třídy *Soubor* z *Objekt* na *Element*
$changeSup(Složka, Objekt, Element)$	změna nadtřídy třídy *Složka* z *Objekt* na *Element*
$attrUp(název, Element, Soubor, Složka)$	přesun atributu *název* ze tříd *Soubor* a *Složka* do společného předka *Element*
$attrUp(vlastník, Element, Soubor, Složka)$	přesun atributu *vlastník* ze tříd *Soubor* a *Složka* do společného předka *Element*

Nalezený cílový stav odpovídající podmínce G je následující:

$$Goal = [inModel(Složka), inModel(Soubor), inModel(String), inModel(Element),$$
$$attrInClass(název, Element), attrInClass(vlastník, Element),$$
$$attrOutOfClass(název, Složka), attrOutOfClass(vlastník, Složka),$$
$$attrOutOfClass(název, Soubor), attrOutOfClass(vlastník, Soubor),$$
$$superClass(Element, Soubor), superClass(Element, Složka),$$
$$superClass(Object, Element),$$
$$hasType(vlastník, Složka), hasType(název, String),$$
$$sending(Client, main, Složka, název), sending(Client, main, Složka, vlastník),$$
$$sending(Client, main, Soubor, název), sending(Client, main, Soubor, vlastník)]$$

Grafickou podobu cílového stavu modelu v notaci UML ukazuje obr. 2.



Obr. 2. – Model sytému souborů podle návrhového vzoru *Composite*

ZÁVĚR

V tomto článku jsme představili automatizovaný transformační aparát SBAT, založený na plánovacím systému STRIPS. Na příkladě jsme ukázali jeho použití pro aplikaci návrhového vzoru *Composite*.
Za hlavní přínos aparátu SBAT pro praxi považujeme podporu automatizace transformací objektově orientovaných modelů, což by mělo vést k ušetření lidských zdrojů v projektových týmech a jejich následnému využití pro ladění, testování či jiné činnosti nezbytné pro kvalitu vyvíjeného softwaru. Dalším přínosem by mělo být teoretické zázemí pro výzkumné aktivity v oblasti modelových transformací.

PODĚKOVÁNÍ

Tento příspěvek byl vytvořen za podpory grantu MŠMT LA08015.

LITERATURA

1. CZARNECKI, Krzysztof – HELSEN, Simon. Feature-based survey of model transformation approaches. *IBM Systems Journal*. 2006, vol. 45, no. 3, s. 621-645.
2. GRAY, Jeff – LIN, Yuehua – ZHANG, Jing. Automating Change Evolution in Model-Driven Engineering. *Computer. 2006*, vol. 31, no. 2, s. 51.
3. JÉZEQUEL, Jean-Marc. *Model Transformation Techniques* [online]. [2005] [cit. 2010-01-18]. Dostupný z WWW: <http://modelware.inria.fr>.
4. KLEPPE, Anneke G. – WARMER, Jos – BAST, Wim. MDA Explained: *The Model Driven Architecture: Practice and Promise.* Boston (MA, USA): Addison-Wesley Longman Publishing Co., Inc., 2003. 170 s. ISBN:032119442X.
5. LIN, Yuehua – GRAY, Jeff. A model transformation approach to automatic model construction and evolution. In *Proceedings of the 20th IEEE/ACM international Conference on Automated Software Engineering.* [s.l.]: [s.n.], 2005. s. 448-451.
6. MARKOVIC, Slavisa. Composition of UML Described Refactoring Rules. In OCL and Model Driven Engineering, UML 2004 Conference Workshop. [s.l.]: [s.n.], 2004. s. 45-59.
7. MENS, Tom – CZARNECKI, Krzysztof – GORP, Pieter Van. A Taxonomy of Model Transformations. In *Language Engineering for Model-Driven Software Development, ser.*

Dagstuhl Seminar Proceedings. Dagstuhl (Germany): Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), 2005.

8. NILSON, Nils J. – FIKES, Richard E. *STRIPS: A new approach to the application of theroem proving to problem solving*. Menlo Park (California): Standford Research Institute, 1970. 34 s.
9. Object Management Group (OMG). *OMG Unifed Modeling Language (OMG UML), Infrastructure: Version 2.2.* [s.l.]: [s.n.], 2009. 226 s. Dostupný z WWW: <www.omg.org>.
10. OPDYKE, William. *Refactoring Object-Oriented Frameworks.* Champaign, (IL, USA), 1992. 197 s. University of Illinois at Urbana-Champaign. Disertační práce.
11. SUNYÉ, Gerson, et al. Refactoring UML Models. In *Proceedings of the 4th International Conference on The Unifed Modeling Language, Modeling Languages, Concepts, and Tools.* London (UK): Springer-Verlag, 2001. s. 134-148. ISBN 3-540-42667-1.
12. ZHANG, Jing – LIN, Yuehua – GRAY, Jeff. Generic and Domain-Specifc Model Refactoring using a Model Transformation Engine. In *Volume II of Research and Practice in Software Engineering.* [s.l.]: [s.n.], 2005. s. 199-218.