Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Vedecká rada Fakulty informatiky
a informačných technológií STU

Ing. Valentino Vranić

Autoreferát dizertačnej práce

# Multi-Paradigm Design with Feature Modeling

(Multiparadigmový návrh s modelovaním vlastností)

na získanie
vedecko-akademickej hodnosti philosophiae doctor
v odbore doktorandského štúdia:
25-31-9 Programové a informačné systémy

Apríl 2004

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia a po uplynutí času určeného na štúdium v súlade s § 12 ods. 1 písm. b) Vyhlášky č. 131/1997 Z.z. na Fakulte informatiky a informačných technológií Slovenskej technickej univerzity v Bratislave.

Predkladateľ: Ing. Valentino Vranić
Ústav informatiky a softvérového inžinierstva
Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave
Ilkovičova 3, 842 16 Bratislava

Školiteľ: doc. Ing. Mária Bieliková, PhD.
Ústav informatiky a softvérového inžinierstva
Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave

Oponenti: prof. Ing. Milan Krokavec, CSc.
Katedra počítačov a informatiky
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach

doc. Ing. Karel Richta, CSc.
Katedra počítačů
Fakulta elektrotechniky
České vysoké učení technické v Praze

izr. prof. dr. Tatjana Welzer Družovec
Inštitut za informatiko
Fakulteta za elektrotehniko, računalništvo in informatiko
Univerza v Mariboru

Autoreferát bol rozoslaný 15. júla 2004. Obhajoba dizertačnej práce sa koná 3. septembra 2004 o 11.00h pred komisiou pre obhajobu dizertačnej práce v odbore doktorandského štúdia vymenovanou predsedom spoločnej odborovej komisie 15. júla 2004 v odbore 25-31-9 Programové a informačné systémy na Fakulte informatiky a informačných technológií Slovenskej technickej univerzity, Ilkovičova 3, 842 16 Bratislava.

Predseda spoločnej odborovej komisie:

prof. Ing. Pavol Návrat, PhD.
Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave
Ilkovičova 3, 842 16 Bratislava

# Obsah

**1 Úvod** **1**

**2 Ciele dizertačnej práce** **1**

**3 Pojem paradigmy vo vývoji softvéru** **2**

**4 Multiparadigmové prístupy k vývoju softvéru** **3**

**5 Modelovanie vlastností pre multiparadigmový návrh** **4**

**6 Multiparadigmový návrh s modelovaním vlastností** **6**

**7 Vyhodnotenie metódy** **9**

**8 Prínosy dizertačnej práce** **10**

**Literatúra** **12**

**Publikácie autora** **15**

**Resume** **17**

**Summary of Contributions** **18**

# 1 Úvod

Štvrťstoročie po prednáške Roberta W. Floyda o paradigmách programovania [Flo79] nedosiahla sa zhoda v chápaní významu pojmu *paradigma* v oblasti vývoja softvéru. Tento pojem sa používa na označenie hocijakého charakteristického prístupu k vývoju softvéru. Keďže softvér nakoniec musí byť vyjadrený programom, neprekvapuje, že pojem paradigma sa spomína najmä v kontexte programovacích jazykov.

Programovacie jazyky sú často klasifikované na základe paradigiem, ktoré podporujú, a zvlášť v zmysle široko akceptovaných paradigiem: procedurálnej, funkcionálnej, logickej a objektovo-orientovanej.

Jestvovanie viacerých paradigiem, z ktorých každá má svoje prednosti, viedlo k idei integrácie alebo kombinácie viacerých programovacích jazykov, z ktorých každý podporuje určitú paradigmu, do jedného, multiparadigmového jazyka. Takýto jazyk však nepomáha pri multiparadigmovom návrhu, v ktorom treba rozhodnúť ktorá paradigma je vhodná pre riešený problém.

Autoreferát pokračuje uvedením cieľov dizertačnej práce v časti 2. Časť 2 uvádza ciele dizertačnej práce. Časť 3 poskytuje opis základných pojmov, a časť 4 stručný prehľad skúmanej oblasti multiparadigmového vývoja softvéru. Ďalej sa stručne uvádzajú hlavné prínosy dizertačnej práce: časť 5 opisuje modelovanie vlastností pre multiparadigmový návrh, kým časť 6 opisuje zavedenú metódu multiparadigmového návrhu s modelovaním vlastností. Následne sú v časti 7 zhrnuté poznatky získané vyhodnotením metódy. Časť 8 predstavuje zhrnutie prínosov dizertačnej práce.

Na konci autoreferátu je uvedený zoznam použitej literatúry a zoznam publikácií autora,[1] ktoré súvisia so skúmanou problematikou, ako aj zhrnutie a zhrnutie prínosov dizertačnej práce v anglickom jazyku.

# 2 Ciele dizertačnej práce

Dizertačná práca sa venuje vylepšeniu multiparadigmového návrhu uplatnením techniky konceptuálneho modelovania známej ako modelovanie vlastností (angl. *feature modeling*). Predpokladom tohoto je analýza pojmu paradigmy ako takého, a zvlášť v kontexte používania viacerých paradigiem súčasne v dnešných prístupoch k vývoju softvéru.

Základné ciele dizertačnej práce boli nasledujúce:

[1] Publikácie sú dostupné na `http://www.fiit.stuba.sk/~vranic`.

- Návrh vhodnej reprezentácie paradigmy na základe vyššie spomenutej analýzy. Za týmto účelom má byť rozšírené a prispôsobené modelovanie vlastností.
- Návrh metódy multiparadigmového návrhu založeného na modelovaní vlastností.
- Vyhodnotenie navrhnutej metódy jej aplikáciou na programovací jazyk AspectJ, čím sa získa jeho model paradigiem, a následnou aplikáciou tohoto modelu pri transformačnej analýze domény samotného modelovania vlastností.

# 3 Pojem paradigmy vo vývoji softvéru

Analýza pojmu paradigma, ktorý je vo vede silne spätý s Thomasom Kuhnom [Kuh70], v zmysle jeho použitia v kontexte vývoja softvéru, ale aj jeho všeobecného významu [Mer] viedla k poznaniu, že paradigmu vo vývoji softvéru možno chápať dvomi spôsobmi: ako paradigmu vo veľkom a paradigmu v malom.

Paradigma vo veľkom (angl. *large-scale paradigm*) [Cop99a] označuje podstatu určitého procesu vývoja softvéru. V tomto význame sa namiesto termínu paradigma používajú aj termíny paradigma programovania alebo len programovanie. Názov paradigmy vyjadruje jej najvýznamnejšiu črtu, a tak je často odvodený od centrálnej abstrakcie paradigmy (napr. funkcia pri funkcionálnej paradigme).

Pojem paradigmy vo veľkom sprevádzajú problémy ohľadom nejednoznačností pri pomenúvaní paradigiem (napr. v [Mey97] sa pod funkcionálnou paradigmou myslí procedurálna) alebo pri identifikácii jej základných čŕt (napr. v [Bud95] sa dedenie nepokladá za základnú črtu objektovo-orientovanej paradigmy), ako aj nerozlišovania medzi paradigmou vývoja softvéru ako takou a prostriedkov na jej realizáciu (napr. v [Bud95] sa hovorí o vizuálnej paradigme).

Paradigma v malom (angl. *small-scale paradigm*) predstavuje iný spôsob chápania paradigmy: na úrovni programovacieho jazyka. Tento prístup bol použitý v [Cop99b], kde sa paradigma definuje ako konfigurácia spoločného a premenlivého (angl. *commonality and variability*). V závislosti od programovacieho jazyka možno hovoriť o paradigmách ako sú procedúra, preťaženie, dedenie apod. Prístup je formálne založený na tzv. analýze rozsahu, spoločného a premenlivého (angl. *scope, commonality and variability analysis*) [CHW98].

Zrejme, je ťažké nájsť programovací jazyk, ktorý by bol jednoparadigmový v kontexte paradigiem v malom. Čím viac paradigiem poskytuje

programovací jazyk, zreteľnejší je problém ich výberu pre dané štruktúry riešenej domény, a tým aj potreba metódy na podporu procesu ich výberu. Takúto metódu možno označiť za *metaparadigmu* vo vzťahu k paradigmám v malom.

# 4 Multiparadigmové prístupy k vývoju softvéru

Hoci objektovo-orientovaná paradigma (OOP) ako taká do istej miery dokáže poskytnúť architektonický rámec pre ďalšie paradigmy [Boo94], o čom svedčí multiparadigmový programovací jazyk Leda [Bud95], má isté nedostatky týkajúce sa znovupoužitia, prispôsobivosti, riadenia zložitosti a výkonnosti [CE00], ktoré podnietili vznik ďalších prístupov k vývoju softvéru.

Jedným z týchto prístupov je *aspektovo-orientované programovanie* (angl. *aspect-oriented programming, AOP*), ktoré umožňuje modularizáciu pretínajúcich záležitostí [KLM+97]. V súčasnosti je oblasť AOP predmetom intenzívneho výskumu [AOS, Fil03]). V dizertačnej práci boli porovnané a zhodnotené štyri prístupy ktoré tvoria základ AOP: aspektovo-orientované programovanie vyvinuté v PARC [KLM+97] (v rámci ktorého bol vytvorený programovací jazyk AspectJ [Ecl]), adaptívne programovanie (angl. *adaptive programming*) [Lie] kompozičné filtre (angl. *compositional filters*) [AT98] a subjektovo-orientované programovanie (angl. *subject-oriented programming*) [IBM].

Ďalším multiparadigmovým prístupom je *generatívne programovanie* (angl. *generative programming*) [CE00], paradigma vývoja rodín softvérových systémov pomocou modelovania vlastností. Generatívne programovanie spája metódy objektovo-orientovanej analýzy a návrhu s metódami doménového inžinierstva. Má blízko k generickému programovaniu, doménovo-špecifickým jazykom a aspektovo-orientovanému programovaniu.

*Multiparadigmové programovanie v jazyku Leda* (angl. *multiparadigm programming in Leda*) [Bud95] je založené na pojmu paradigmy vo veľkom. Leda podporuje všetky štyri široko akceptované paradigmy: procedurálnu, funkcionálnu, logickú a objektovo-orientovanú. V súvislosti s jazykom Leda bola vytvorená empirická metóda návrhu zhora nadol [KBV00], ktorá pomáha pri výbere paradigmy pre riešený problém.

*Multiparadigmový návrh* (angl. *multi-paradigm design*) [Cop99b] je metaparadigma určená na vývoj rodín systémov. Zaoberá sa výberom vhodnej paradigmy pre danú vlastnosť, ktorá sa má navrhnúť a imple-

mentovať. Multiparadigmový návrh je založený na analýze rozsahu, spoločného, premenlivého a vzťahov (angl. *scope, commonality, variability, and relationships analysis; SCVR*), ktorá predstavuje rozšírenie analýzy rozsahu, spoločného a premenlivého rozšírenú o vzťahy [CHW98]. V multiparadigmovom návrhu sa rozlišuje medzi doménou aplikácie (angl. *application domain*), na ktorú sa aplikuje proces vývoja softvéru (t.j. ktorá sa rieši), a doménou riešenia (angl. *solution domain*), v ktorej sa riešenie vyjadruje (zvyčajne programovací jazyk). Multiparadigmový návrh pozostáva zo štyroch hlavných krokov: analýze SCVR domény aplikácie, analýze SCVR domény riešenia, transformačnej analýze a návrhu kódu. Hlavné výsledky analýz obidvoch domén sa prezentujú v tabuľkách, a množstvo dôležitých informácii sa uvedádza v tvare voľného textu, čo nie je postačujúce pre transformačnú analýzu, a ani následný návrh kódu.

Posledným multiparadigmovým prístupom prezentovaným v dizertačnej práci je *intenčné programovanie* (angl. *intentional programming*) [Sim99, Röd99], ktoré predstavuje pokus o zbavenie sa obmedzení programovacích jazykov s pevnou syntaxou. V tomto prístupe vyvinutom v Microsofte (výskum v súčasnosti pozastavený) sa program reprezentuje pomocou tzv. intenčného stromu, na ktorého tvorbu a úpravu je potrebný špeciálny editor.

Z analýzy uvedených multiparadigmových prístupov vyplynulo, že základná idea multiparadigmového návrhu [Cop99b] je vhodná pre multiparadigmový vývoj softvéru.

# 5 Modelovanie vlastností pre multiparadigmový návrh

Modelovanie vlastností je technika konceptuálneho modelovania domén, v ktorej sú koncepty vyjadrené svojimi vlastnosťami zohľadňujúc ich vzájomné závislosti a premenlivé za účelom zachytenia konfigrovateľnosti konceptov (upravená definícia z [CE00]). Doména sa chápe ako oblasť záujmu [Cop99b]. Na základe roly, ktorú doména má vo vývoji softvéru, možno rozlišovať medzi doménou aplikácie, na ktorú sa aplikuje proces vývoja softvéru (t.j. ktorá sa rieši), a doménou riešenia, v ktorej sa riešenie vyjadruje (zvyčajne programovací jazyk).

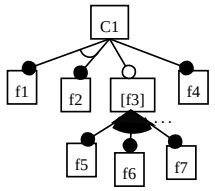
Modelovanie vlastností, pochádzajúce z [KCH+90], je súčasťou rôznych metód [KKL+98, Sim95, GFd98, CE00, Gey00], v tvare prispôsobenom ich potrebám. Modelovanie vlastností v notácii Czarnecki--Eisenecker [CE00] je dostatočne všeobecné a abstraktné, aby mohlo byť

prispôsobené na použitie v multiparadigmovom návrhu.

Dizertačná práca podrobne opisuje modelovanie vlastností postavené na upresnených definíciách základných pojmov modelovania vlastností v notácii Czarnecki-Eisenecker: konceptu, inštancie konceptu, vlastnosti — spoločnej a premenlivej (angl. *common and variable feature*), modelu vlastností a diagramu vlastností (angl. *feature diagram*).

Kľúčovou súčasťou modelov vlastností sú diagramy vlastností. Stručne povedané, diagram vlastností je reprezentovaný orientovaným stromom s dekoráciou hrán. Koreň tohoto stromu predstavuje koncept. Ostatné uzly sú vlastnosti, pričom sa každá takáto vlastnosť vzťahuje na svojho rodiča. Dva typy hrán, s ukončením v tvare plného alebo prázdneho kruhu, sa používajú na rozlíšenie medzi povinnými, resp. voliteľnými vlastnosťami (angl. *mandatory, optional features*). Dva typy dekorácií hrán, v tvare plného alebo prázdneho oblúku, sa používajú na vyčlenenie disjunktných podmnožín hrán vychádzajúcich z jedného uzla, čím možno definovať rozdelenie na alternatívne (angl. *alternative feature*), resp. alebo-vlastnosti (angl. *or-feature*).

Príklad diagramu vlastností je uvedený na obr. 1. Vlastnosti *f1* a *f2* sú povinné alternatívne vlastnosti. Vlastnosť *f3* je voliteľná vlastnosť. Táto vlastnosť je otvorená, t.j. očakáva sa, že bude mať ďalšie vlastnosti (tri bodky naznačujú, že v skupine alebo-vlastností). Vlastnosti *f5*, *f6* a *f7* sú povinné alebo-vlastnosti.



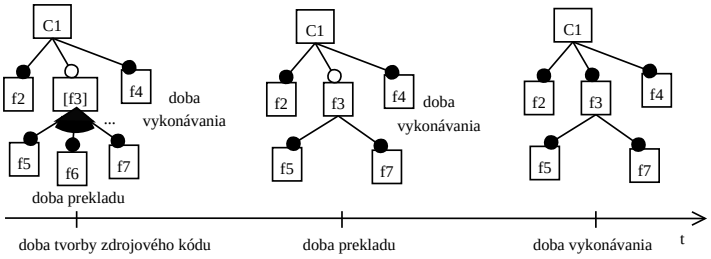

Obrázok 1: Príklad diagramu vlastností.

V dizertačnej práci sú definované ďalšie záležitosti modelovania vlastností: referencie konceptov (angl. *concept references*), informácie pridružené konceptom a vlastnostiam, obmedzenia a pravidlá preddefinovaných závislostí (angl. *constraints and default dependency rules*), parametrizácia v modeloch vlastností a tvorba a reprezentácia inštancií konceptov.

Z vymenovaného práve tvorba a reprezentácia inštancií konceptov, navrhnutá tak, aby zohľadňovala dobu inštanciácie, má osobitný význam pre transformačnú analýzu v multiparadigmovom návrhu. Súčasťou informácií pridružených premenlivým vlastnostiam je preto aj doba viazania alebo mód viazania (angl. *binding time/mode*). Doba viaza-

nia určuje kedy sa má daná vlastnosť vybrať. Doba viazania sa určuje v zmysle postupnosti dôb viazaní v doméne riešenia. Namiesto doby viazania možno použiť mód viazania, ktorý určuje či daná vlastnosť je viazaná dynamicky alebo staticky, čo je vhodné v prípade keď doména riešenia nie je známa.

Inštancie konceptov sú reprezentované diagramami vlastností a pokladajú sa tiež za koncepty, čo znamená, že môžu byť ďalej inštanciované, ako je pre ukázané na obr. 2. Inštanciácia konceptov môže prebiehať zhora-nadol, čo je obvyklý spôsob, ale aj zdola-nahor, kedy možno hovoriť o úspešnosti inštanciácie, pričom inštanciácia je úspešná v prípade zahrnutia samotného uzla konceptu.

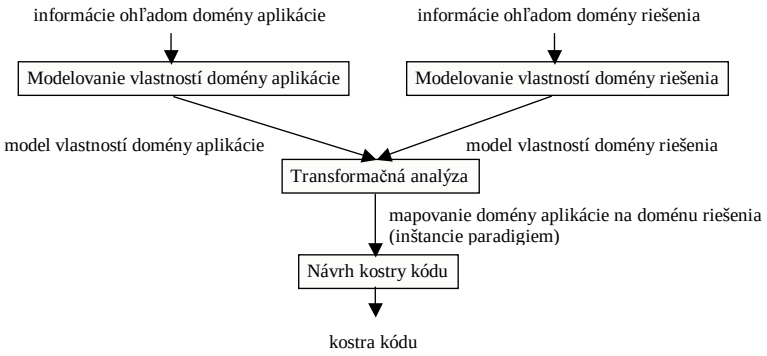



Obrázok 2: Príklad inštanciácie konceptu z obr. 1 zohľadňujúcej dobu inštanciácie.

V dizertačnej práci je definovaný postup aplikácie modelovania vlastností, ktorý je predvedený na doméne vyrovnávacej pamäti pre editory textu. Navrhnuté rozšírenia modelovania vlastností sú porovnané s inými prístupmi týkajúcimi sa rozšírenia modelovania vlastností vzhľadom na UML [RBSP02, Cla01], jeho formalizácie [JG02], potreby klasifikácie vlastností [KKL+98, KCH+90, Rie03] a reprezentácie obmedzení [SRP03].

# 6 Multiparadigmový návrh s modelovaním vlastností

Hlavným prínosom dizertačnej práce je nová metóda multiparadigmového vývoja softvéru, *multiparadigmový návrh s modelovaním vlastností* ($\text{MPD}_{\text{FM}}$), ktorá je založená na pojme paradigmy v malom a modelovaní vlastností. Ako ukazuje obr. 3, $\text{MPD}_{\text{FM}}$ pozostáva zo štyroch činností. Obr. 4 ukazuje pozíciu $\text{MPD}_{\text{FM}}$ v celkovom procese vývoja softvéru.



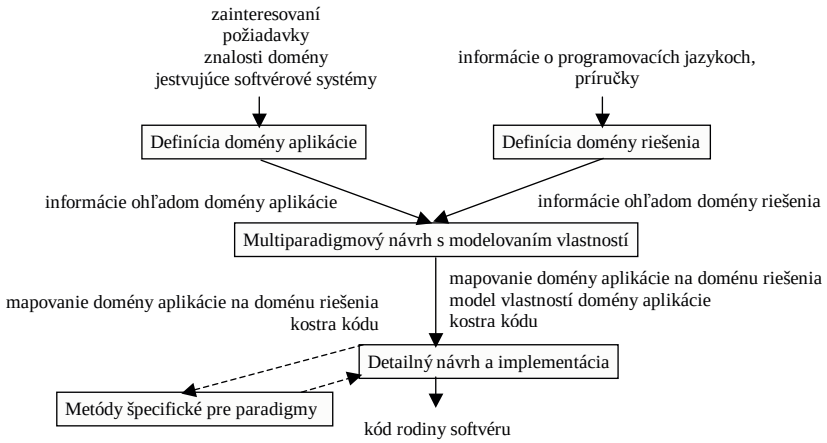

Obrázok 3: Multiparadigmový návrh s modelovaním vlastností.

Hlavným výstupom $\text{MPD}_{\text{FM}}$ je kostra kódu. Modely vlastností domény aplikácie a domény riešenia tiež predstavujú užitočné medzivýstupy, ktoré je možné znovupoužiť v transformačnej analýze. Raz vytvorený model vlastností domény riešenia je možné znovupoužiť v transformačnej analýze všetkých domén aplikácie, ktoré sa budú v tejto doméne riešenia implementovať. Model vlastností domény aplikácie je možné znovupoužiť v transformačnej analýze za účelom jeho implementácie v inej doméne riešenia.

Modelovanie vlastností domény aplikácie sa riadi všeobecným postupom aplikácie modelovania vlastností. V modelovaní vlastností domény riešenia sa modeluje koncept riešenia a paradigmy ako koncepty domény riešenia. Preto sa modelovanie vlastností domény riešenia v ďalšom výklade označuje aj ako modelovanie paradigiem, a model vlastností domény riešenia ako model paradigiem. Paradigmy predstavujú podkoncepty konceptu riešenia, čo je vyjadrené v modeli vlastností konceptu riešenia, ktorého vlastnosti sú referencie paradigiem. V ďalšom výklade sa bude predpokladať, že doména riešenia je programovací jazyk, čo je najčastejší prípad.

Jednotlivé paradigmy programovacieho jazyka sa modelujú podobne ako koncepty vo všeobecnosti. Pre každú premenlivú vlastnosť sa musí určiť jej doba viazania (nie je možné použiť mód viazania). Postupnosť dôb viazania v modelovanej doméne riešenia treba predtým stanoviť z pohľadu životného cyklu programu v tejto doméne začínajúc tvorbou zdrojového kódu (doba tvorby zdrojového kódu), cez ďalšie charakteristické body (doba prekladu, doba spájania, doba načítavania apod.), až po vykonávanie programu (doba vykonávania).

Väčšina paradigiem zodpovedá hlavným konštrukciám, t.j. štruktú-




Obrázok 4: Multiparadigmový návrh s modelovaním vlastností a ostatné činnosti počas vývoja softvéru.

ram, programovacieho jazyka. V transformačnej analýze uzol v modeli vlastností môže zodpovedať koreňu takejto štrukturálnej paradigme. Štrukturálne paradigmy jazyka AspectJ sú: trieda, interfejs, metóda a aspekt (angl. *class, interface, method, aspect*).

Okrem štrukturálnych paradigiem, sú aj paradigmy vzťahu medzi štruktúrami programovacieho jazyka. V transformačnej analýze nijaký uzol v modeli vlastností nebude zodpovedať koreňu takejto vzťahovej paradigme. Vzťahové paradigmy jazyka AspectJ sú: dedenie, preťaženie, medzitypová deklarácia, rada a prierez (angl. inheritance, overloading, inter-type declaration, advice pointcut).

Kľúčovou činnosťou v multiparadigmovom návrhu je transformačná analýza, proces hľadania korešpondencie a mapovania medzi konceptmi domény aplikácie a riešenia. Transformačná analýza v $\mathrm{MPD_{FM}}$ je definovaná plne v zmysle modelovania vlastností ako *inštanciácia paradigiem zdola nahor nad konceptmi domény aplikácie v dobe tvorby zdrojového kódu*. Jej vstupom sú dva modely vlastností: domény aplikácie a domény riešenia. Jej výstupom je množina inštancií paradigiem s anotáciami o zodpovedajúcich konceptoch a vlastnostiach modelu vlastností domény aplikácie.

Dizertačná práca definuje jednotlivé kroky procesu transformačnej analýzy. Výsledkom úspešnej transformačnej analýzy je len jedno z možných riešení. Iné rozhodnutia pri výbere zodpovedajúcej paradigmy môžu viesť k inému riešeniu. Porovnávanie týchto riešení a hľadanie

najlepšieho nie je však predmetom tejto metódy.

Posledná činnosť v $\text{MPD}_{\text{FM}}$, návrh kostry kódu sa uskutočňuje prechádzaním jednotlivými inštanciami paradigiem a ručným zaznamenávaním zdrojového kódu. Pritom treba najprv transformovať štrukturálne paradigmy, a až potom vzťahové paradigmy, ktoré na nich stavajú.

Navrhnutá metóda bola porovnaná s príbuznými prístupmi: multiparadigmovým naávrhom [Cop99b], multiparadigmovým programovaním v jazyku Leda [KBV00] a generatívnym programovaním [CE00].

# 7 Vyhodnotenie metódy

Navrhnutá metóda multiparadigmového návrhu s modelovaním vlastností bola vyhodnotená na základe jej aplikácie na doménu aplikácie a riešenia nezanedbateľnej veľkosti. Pri vyhodnocovaní sa sledovala škálovateľnosť metódy a znovupoužiteľnosť jej výstupov, ako aj čitateľnosť a zrozumiteľnosť artefaktov vznikajúcich jej použitím.

Modelovanie domény aplikácie v $\text{MPD}_{\text{FM}}$ bolo aplikované na samotnú doménu modelovania vlastností (jej model vlastností je uvedený v prílohe k dizertačnej práci). V tejto doméne bolo identifikovaných dvanásť konceptov. Maximálna výška diagramov vlastností je dva, pričom najväčší diagram vlastností pozostáva z 23 uzlov. Referencie konceptov umožňujú vyhnuť sa vytváraniu veľkých diagramov vlastností. Znovupoužitie modelu vlastností domény aplikácie je možné v ďalšej transformačnej analýze, čo môže vyžadovať korekciu dôb viazania, alebo aj použitie v inej metóde založenej na modelovaní vlastností, kedy možno očakávať potrebu rozsiahlejších úprav.

Modelovanie domény riešenia v $\text{MPD}_{\text{FM}}$ bolo aplikované na programovací jazyk AspectJ (verzia 1.1.1), čím bol získaný jeho model paradigiem (uvedený v prílohe k dizertačnej práci). Model paradigiem danej domény riešenia možno považovať za špecializáciu $\text{MPD}_{\text{FM}}$ pre túto doménu. Model paradigiem jazyka AspectJ teda definuje $\text{MPD}_{\text{FM}}$ pre AspectJ.

Model paradigiem jazyka AspectJ pozostáva z desiatich paradigiem, čo spolu so štyrmi pomocnými konceptmi identifikovanými v doméne tohoto jazyka predstavuje štrnásť diagramov vlastností v modeli. Maximálna výška diagramov vlastností je tri, pričom najväčší diagram vlastností pozostáva z 25 uzlov. Tieto hodnoty sú podobné hodnotám získaným v modelovaní domény aplikácie, čo naznačuje, že aj v iných doménach možno očakávať diagramy vlastností porovnateľnej veľkosti, vcelku prijateľne ohľadom čitateľnosti a zrozumiteľnosti. Ako sa dalo očakávať, modely paradigiem možno znovupoužiť nezávisle od modelov vlastností

domén aplikácie. Model paradigiem jazyka AspectJ bol znovupoužitý v dizertačnej práci: bol použitý v transformačnej analýzy domény vyrovnávacej pamäti pre editory textu a domény modelovania vlastností.

Transformačná analýza domény modelovania vlastností s použitím navrhnutého modelu vlastností jazyka AspectJ bola úspešne vykonaná (jej výstup je uvedený v prílohe k dizertačnej práci). Výška niektorých inštancií paradigiem vytvorených v tejto transformačnej analýze dosahuje šesť, kým počet uzlov presahuje päťdesiat. Toto je vyvolané rozpísaním referencií konceptov v paradigmách počas ich inštanciácie. Tento problém možno vyriešiť uvádzaním každej inštancie referencovaného konceptu v osobitnom diagrame vlastností a jeho referencovaním v hlavnom diagrame. Na druhej strane, treba dať prednosť spoločným inštanciám paradigiem kedykoľvek je to možné za účelom poskytnutia lepšieho prehľadu príbuzných inštancií paradigiem. Inštanciácia paradigiem nezávisí od počtu konceptov domény aplikácie alebo riešenia, lebo vždy pracuje len s niekoľkými konceptmi domény aplikácie a jedinou paradigmou (nepočítajúc paradigmy, ktoré táto paradigma referencuje).

Ako bolo v dizertačnej práci preukázané, kostra kódu sa pomerne ľahko odvodí z inštancií paradigiem (kód je v prílohe k dizertačnej práci). Tento proces sa vykonáva sekvenčne, a tak je jeho zložitosť nezávislá od počtu inštancií paradigiem.

# 8 Prínosy dizertačnej práce

Hlavné prínosy dizertačnej práce sú nasledujúce:

1. Multiparadigmový návrh s modelovaním vlastností, nová metóda multiparadigmového návrhu softvéru založená na modelovaní vlastností, ktorá vylepšuje proces výberu paradigiem.

2. Multiparadigmový návrh s modelovaním vlastností pre AspectJ, špeciálizácia metódy pre jazyk AspectJ, definovaná predloženým modelom paradigiem jazyka AspectJ.

3. Vylepšenia modelovania vlastností:

   (a) Inštanciácia konceptov zohľadňujúca dobu inštanciácie s inštanciami konceptov reprezentovanými pomocou diagramov vlastností.

   (b) Parametrizácia modelov vlastností.

   (c) Reprezentácia obmedzení a pravidiel preddefinovaných závislostí pomocou logických výrazov.

   (d) Referencie konceptov, ktoré umožňujú vysporiadanie sa so zložitými modelmi vlastností.

   (e) Bodková konvencia umožňujúca jednoznačné odvolávanie sa na koncepty a vlastností.

   (f) Parametrizovaný koncept na reprezentáciu kardinality v modelovaní vlastností.

4. Model vlastností samotnej domény modelovania vlastností, ktorý poskytuje základ pre ďalšie uvažovanie o tejto technike modelovania.

5. Aplikácia multiparadigmového návrh s modelovaním vlastností pre AspectJ na doménu modelovania vlastností, ktorá poskytuje základ pre vývoj systémov na podporu modelovania vlastností.

Okrem uvedeného, dizertačná práca prispieva aj k chápaniu pojmu paradigmy vo vývoji softvéru, poskytuje prehľad vybraných multiparadigmových prístupov k vývoju softvéru, ako aj prístupov k modelovaniu vlastností.

# Literatúra

[AOS] AOSD steering committee. Aspect-Oriented Software Development home page. `http://aosd.net`. Last accessed in March 2004.

[AT98] Mehmet Aksit and Bedir Tekinerdogan. Solving the modeling problems of object-oriented languages by composing multiple aspects using composition filters. In *Proc. of the Aspect-Oriented Programming Workshop at ECOOP'98*, Brussels, Belgium, 1998.

[Boo94] Grady Booch. *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, second edition, 1994.

[Bud95] Timothy A. Budd. *Multiparadigm Programming in Leda*. Addison-Wesley, 1995.

[CE00] Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative Programing: Principles, Techniques, and Tools*. Addison-Wesley, 2000.

[CHW98] James Coplien, Daniel Hoffman, and David Weiss. Commonality and variability in software engineering. *IEEE Software*, 15(6), November 1998.

[Cla01] Matthias Clau$\beta$. Modeling variability with UML. In *Proc. of Net.ObjectDays 2001, Young Researchers Workshop on Generative and Component-Based Software Engineering*, pages 226–230, Erfurt, Germany, September 2001. tranSIT.

[Cop99a] James O. Coplien. Multi-paradigm design and implementation in C++. Slides and notes of the tutorial given at *1st International Conference on Generative and Component-Based Software Engineering (GCSE'99)*, Erfurt, Germany, September 1999.

[Cop99b] James O. Coplien. *Multi-Paradigm Design for C++*. Addison-Wesley, 1999.

[Ecl] Eclipse.org. AspectJ project home page. `http://eclipse.org/aspectj`. Last accessed in March 2004.

[Fil03] Robert E. Filman. A bibliography of aspect-oriented programming, version 1.22. Technical Report 03.01, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, California, June 2003.

[Flo79] Robert W. Floyd. The paradigms of programming. *Communications of the ACM*, 22(8):455–460, 1979.

[Gey00] Lars Geyer. Feature modelling using design spaces. In *Proc. of the 1st German Product Line Workshop (1. Deutscher Software-Produktlinien Workshop, DSPL-1)*, Kaiserslautern, Germany, November 2000. IESE.

[GFd98] Martin L. Griss, John Favaro, and Massimo d'Alessandro. Integrating feature modeling with the RSEB. In P. Devanbu and J. Poulin,

editors, *Proc. of 5th International Conference on Software Reuse*, pages 76–85, Vicoria, B.C., Canada, 1998. IEEE Computer Society Press.

[IBM] IBM Research. Subject-Oriented Programming home page. `http://www.research.ibm.com/sop`. Last accessed in March 2004.

[JG02] Yu Jia and Yuqing Gu. The representation of component semantics: A feature-oriented approach. In Ivica Crnković, Stig Larsson, and Judith Stafford, editors, *Proc. of the Workshop on Component-based Software Engineering: Composing Systems From Components (a part of 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems)*, Lund, Sweden, April 2002.

[KBV00] Charles D. Knutson, Timothy A. Budd, and Hugh Vidos. Multiparadigm design of a simple relational database. *ACM SIGPLAN Notices*, 35(12):51–61, December 2000.

[KCH+90] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. Feature-oriented domain analysis (FODA): A feasibility study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, November 1990.

[KKL+98] Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Euiseob Shin, and Moonhang Huh. FORM: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering*, 5:143–168, January 1998.

[KLM+97] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Christina Vidiera Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In Mehmet Aksit and Satoshi Matsuoka, editors, *Proc. of 11th European Conference on Object-Oriented Programming (ECOOP'97)*, LNCS 1241, Jyväskylä, Finland, June 1997. Springer.

[Kuh70] Thomas S. Kuhn. *The Structure of Scientific Revolutions.* University of Chicago Press, Chicago, 1970. Czech translation, OIKYMENH, 1997.

[Lie] Karl J. Lieberherr. Connections between Demeter/adaptive programming and aspect-oriented programming. Web document, College of Computer Science, Northeastern University, Boston, USA.

[Mer] Merriam-Webster OnLine. Merriam-Webster's Collegiate Dictionary. `http://www.m-w.com`. Last accessed in April 2004.

[Mey97] Bertrand Meyer. *Object-Oriented Software Construction.* Prentice Hall, second edition, 1997.

[RBSP02] Matthias Riebisch, Kai Böllert, Detlef Streitferdt, and Ilka Philippow. Extending feature diagrams with UML multiplicities. In *Proc.*

*of the 6th Conference on Integrated Design and Process Technology (IDPT 2002)*, Pasadena, California, USA, June 2002. Society for Design and Process Science.

[Röd99] Lutz Röder. Transformation and visualization of abstractions using the intentional programming system. Presentation abstract, GCSE'99 Young Researchers Workshop, 1st International Conference on Generative and Component-Based Software Engineering, Erfurt, Germany, September 1999.

[Rie03] Matthias Riebisch. Towards a more precise definition of feature models. In M. Riebisch, J. O. Coplien, and D. Streitferdt, editors, *Modelling Variability for Object-Oriented Product Lines*, pages 64–76, Norderstedt, 2003. BookOnDemand Publ. Co.

[Sim95] Mark A. Simos. Organization domain modeling (ODM): Formalizing the core domain modeling life cycle. In *Proc. of the 1995 Symposium on Software reusability*, pages 196–205, Seattle, Washington, United States, 1995. ACM Press.

[Sim99] Charles Simonyi. The future is intentional. *IEEE Computer*, 32(5):56–57, May 1999.

[SRP03] Detlef Streitferdt, Matthias Riebisch, and Ilka Philippow. Details of formalized relations in feature models using OCL. In *Proc. of the 10th IEEE Symposium and Workshops on Engineering of Computer-Based Systems (ECBS'03)*, pages 297–304, Pasadena, California, USA, April 2003. IEEE Computer Society.

# Publikácie autora

## Medzinárodné časopisy

Valentino Vranić. Feature Modeling Based Transformational Analysis in Multi-Paradigm Design. Submitted to *Computers and Informatics (CAI)*, December 2003.

Valentino Vranić. Towards multi-paradigm software development. *Journal of Computing and Information Technology (CIT)*, 10(2):133–147, 2002.

Valentino Vranić, Peter Dolog, and Mária Bieliková. Representing change by aspect. *ACM SIGPLAN Notices*, 36(12):77–83, December 2001.

## Medzinárodné konferencie

Valentino Vranić. AspectJ paradigm model: A basis for multi-paradigm design for AspectJ. In Jan Bosch, editor, *Proc. of 3rd International Conference on Generative and Component-Based Software Engineering (GCSE 2001)*, LNCS 2186, pages 48–57, Erfurt, Germany, September 2001. Springer.

Valentino Vranić. Multiple software development paradigms and multi-paradigm software development. In J. Zendulka, editor, *Proc. of 3rd International Conference on Information Systems Modelling (ISM 2000)*, pages 191–196, Rožnov pod Radhoštěm, Czech Republic, May 2000. MARQ.

## Ostatné

Valentino Vranić. Multi-Paradigm Design with Feature Modeling. Presentation slides of the lecture given at the Faculty of Organizational Sciences, University of Belgrade, Serbia, March 2002. In Serbian.

Valentino Vranić. AspectJ paradigm model: A basis for multi-paradigm design for AspectJ. Technical report, Slovak University of Technology in Bratislava, Slovakia, May 2001.

Valentino Vranić. A new basis for multi-paradigm design. Technical report, Slovak University of Technology in Bratislava, Slovakia, March 2001.

Valentino Vranić. Incorporating variability dependency graphs into multi-paradigm design with feature modeling. In *Proc. of 4th Scientific Conference on Electrical Engineering and Information Technology for Ph.D. Students* ElITech 2001, Bratislava, Slovakia, 2001.

Valentino Vranić. A concept of paradigm in the multi-paradigm software development. In *Proc. of the 3rd Scientific Conference on Electrical Engineering and Information Technology for Ph.D. Students* ELITECH 2000, Bratislava, Slovakia, September 2000.

Valentino Vranić. Towards multi-paradigm software development. Slides and notes of the presentation given at 4th Joint Conference on Knowledge-Based Software Engineering (JCKBSE 2000), Brno, Czech Republic, September 2000.

Valentino Vranić. Towards Multi-Pradigm Software Development. Slovak University of Technology in Bratislava, Slovakia, September 2000. Written part of the PhD examination.

## Resume

Based on the analysis of multi-paradigm software development and the concept of paradigm, a new method of multi-paradigm design with feature modeling is proposed in this thesis. The method enables an explicit reasoning about paradigms, viewed as solution domain concepts, and their appropriateness for given application domain concepts. Both application and solution domain are modeled using a conceptual modeling technique known as feature modeling adapted to the needs of multi-paradigm design. The process of paradigm selection is defined also in terms of feature modeling as a bottom-up paradigm instantiation over application domain concepts. Its output is a set of paradigm instances annotated with the information about corresponding application domain concepts and features. According to these paradigm instances, the code skeleton is being designed. The method is demonstrated and evaluated on the solution domain of AspectJ programming language and the application domain of feature modeling.

## Summary of Contributions

The main contributions of this thesis are as follows:

1. Multi-paradigm design with feature modeling, a new method of multi-paradigm software development based on feature modeling which improves paradigm selection process.
2. Multi-paradigm design with feature modeling for AspectJ, the method specialization to AspectJ defined by providing an AspectJ paradigm model.
3. Improvements of feature modeling:
   (a) Concept instantiation with respect to instantiation time with concept instances represented by feature diagrams.
   (b) Parameterization in feature models.
   (c) Representing constraints and default dependency rules by logical expressions.
   (d) Concept references to enable to deal with complex feature models.
   (e) A dot convention to enable referring to concepts and features unambiguously.
   (f) A parameterized concept for representing cardinality in feature modeling.
4. A feature model of the domain of feature modeling itself, which provides a basis for further reasoning on this modeling technique.
5. An application of multi-paradigm design with feature modeling for AspectJ to the domain of feature modeling, which provides a basis for developing systems to support feature modeling.

In addition to contributions listed above, the thesis contributes to the understanding of the concept of paradigm in software development, provides an overview of selected multi-paradigm approaches to software development, and evaluates approaches to feature modeling.