Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Metamodellek alkalmazása szoftvermodellek transzformációs eljárásaiban

Ph.D. értekezés tézisei

Levendovszky Tihamér

Tudományos vezető:
Dr. Charaf Hassan Ph.D.
egyetemi docens

Budapesti Műszaki és Gazdaságtudományi Egyetem
Automatizálási és Alkalmazott Informatikai Tanszék

Budapest, 2005

Ph.D. értekezés tézisei

Levendovszky Tihamér

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

1111 Budapest, Goldmann György tér 3.

E-mail: tihamer.levendovszky@aut.bme.hu
Telefon: 463-1662
Fax: 463-3478

Tudományos vezető:

Dr. Charaf Hassan Ph.D.
egyetemi docens

1. Előzmények és célkitűzések

A szoftvertechnológia fejlődésében a nyelvi absztrakciós szint egyértelmű növekedését figyelhetjük meg. A kezdetekre jellemző alacsonyszintű assembly nyelvek egyeduralmát lassanként felváltották a strukturált, majd később az objektum orientált programozási nyelvek, amelyek egyre több magasszintű nyelvi konstrukcióval segítik a szoftverrendszerek fejlesztését, valamint a természetes nyelveken megfogalmazott feladatok és a programozási nyelvek adta lehetőségek közötti szemantikus rés csökkentését. Ez a jelenség leginkább azzal magyarázható, hogy a szoftverekkel szemben támasztott követelmények szinte nem ismernek határokat sem komplexitásban, sem a fejlesztési idő korlátosságában, sem pedig a használni kívánt platformok változatosságában.

Egy ilyen absztrakciós folyamatot figyelhetünk meg az először az objektum orientált paradigma kísérőjelenségeként felbukkanó **vizuális modellező nyelv**ek esetében, amelyek az OMG szabvány Modellvezérelt Architektúra (**Model-Driven Architecture**, MDA) [OMG MDA] [OMG MDAGuide] [Kleppe et al. 2003] [Mellor et al. 2004] keretein belül mintegy önálló programozási nyelvet alkotnak. Az MDA víziója szerint a fejlesztőknek egy platformfüggetlen modellt (Platform Independent Model, PIM) kell felépíteniük és részletesen specifikálniuk, főként a Unified Modeling Language (UML) [OMG UML 2003] [Pender & Pender 2003] nevű grafikus modellezési nyelven. A PIM-ből az éppen használni kívánt platformra vonatkozó információk hozzáadásával a modellfordítók (model compilers) automatikusan egy platformspecifikus modellt (Platform Specific Model, PSM) készítenek, amelyek futtathatók az adott platformon. Az MDA működéséhez tehát minél jobb modellfordítókra van szükség, amelyek képesek a gráf alapú modellekből kódot, vagy esetleg további, megváltoztatott grafikus modelleket létrehozni.

A modellfordítók legnagyobb része gráfelméleti algoritmusokat felhasználva bejárja a feldolgozni kívánt modellt (Traversing Model Processor, TMP), vagyis értelmezi, interpretálja azt (Model Interpreters, [Lédeczi et al. 2001]). Ugyanakkor az ilyen jellegű modellfordítók fejlesztése időigényes, sok hibalehetőséget rejtő, hosszadalmas feladat. Másfelől egy grafikus, modell orientált fejlesztés részeként természetesnek tűnik az absztrakciós szint emelése, vagyis a modellfordítók grafikus specifikálása (Visual Model Processor, VMP). Ehhez kitűnő matematikai hátteret biztosít a **gráf újraírás** (graph rewriting [Ehrig 1979] [Blostein et al. 1995] [Ehrig & Taenzer 1996] [Rozenberg 1997] [Ehrig et al. 1999] [Baresi & Heckel 2002]) módszere. A gráf újraírás szabályokból áll, amelyek megközelítés szempontjából nagyon hasonlítanak a fordítóprogramoknál alkalmazott helyettesítési szabályokra. Gráf újraírás esetén a szabálynak van egy bal oldala (Left Hand Side, LHS), amellyel izomorf részgráfot kell keresni a transzformálandó gráfban. Ezek után a megtalált részgráfot a szabály jobb oldalára (Right Hand Side, RHS) kell cserélnünk. A gráf újraírásnak több változata van, a szoftvermodellek esetében az algebrai megközelítések [Ehrig 1979] [Rozenberg 1997], nevezetesen a Double Pushout (DPO) és a Single Pushout (SPO) megközelítések bizonyultak a leghasznosabbnak. Ezek – az absztrakt algebrai formalizmusokon kívül – kategóriaelméleti [Pierce 1991] [Barr & Wells 1999] szemlélettel formalizálják az újraírás folyamatát. A DPO megközelítés esetén egy törlendő LHS elemhez csak egy bemeneti gráfbeli csomópont tartozhat és fordítva (identification condition), valamint csak olyan csomópontokat törölhetünk, amelyekhez a hozzá tartozó éleket is töröljük, élet viszont úgy törölhetünk, ha mindkét pontja benne van az LHS gráfban. Az SPO megközelítésnek nincsenek ilyen megkötései, ami ellentmondó műveletekhez vezethet, de az SPO megközelítés ezt a helyzetet a törlés mindig magasabb prioritásra helyezésével oldja fel.

A grafikus nyelvek többségét támogató eszközök nagy részét egy előre meghatározott grafikus nyelvcsaládra írták meg, ahol a programkód specifikussága lehetetlenné teszi a komolyabb bővítést, illetve az új nyelvek támogatását. Főként beágyazott rendszerek fejlesztésénél azonban sokszor felmerül az igény, hogy a modellező környezetek mindig új, úgynevezett tartományspecifikus nyelveket (Domain Specific Languages, DSL) támogassanak és tegyenek feldolgozhatóvá. Erre a problémára hatékony megoldásnak bizonyult a

metamodellezési technika, amely egy grafikus modellben (metamodell) definiálja azokat a kényszereket, amelyeket a modellnek be kell tartania, illetve azokat az elemeket, amelyeket a modellek tartalmazhatnak. A metamodell és az UML osztálydiagram között elég könnyen húzhatunk párhuzamot: egy metamodellező környezetben létrehozunk egy UML osztálydiagramot (metamodell), és a modellező üzemmódban a környezet csak azokat az objektumdiagramokat (modell) engedi létrehozni, amelyek példányosítják az osztálydiagramot. Ezt a technikát grafikus támogatással kiegészítve elég könnyen és gyorsan hozhatunk létre új modellezési nyelvek támogatására alkalmas környezetet. Az alkalmazási terület túlmutat a tartományspecifikus nyelveken, ugyanis az UML szabvány verziók közötti változása olyan nagy mértékű, hogy a szabványos modellezési nyelv naprakész támogatása könnyebben biztosítható metamodellezési technikák segítségével. A metamodellezési környezetek közül a disszertáció eredményeire leginkább a Vanderbilt Egyetemen fejlesztett Generic Modeling Environment (GME) [Lédeczi et al. 2001] volt a legnagyobb hatással, amelynek fejlesztésében rövid ideig lehetőségem nyílt részt venni.

Időszerűnek és érdemesnek látszik tehát a gyorsabb fejlesztés érdekében vizuális modellfordítókat létrehozni, amelynek formalizmusa közel áll a szabványos, jól ismert UML modellezési nyelvhez, ötvözi a gráf újraírás és a metamodellezés előnyeit. Mindezek fényében céljaim az alábbiak voltak:

1. Létrehozni egy metamodellező környezetet, amely a GME-nél szerzett tapasztalatokon nyugszik, de közvetlen támogatást nyújt VMP-k fejlesztésére, és az UML támogatása is hatékonyabb (a GME elsősorban tartományspecifikus nyelvek támogatására készült).
2. Megalkotni egy grafikus modellezési nyelvet a transzformációs szabályok leírására, amely hasonlít az UML-re, és figyelembe veszi a metamodellek előzetes rendelkezésre állását, valamint azt, hogy a metamodellben általános kényszerek adhatók meg.
3. Kialakítani egy olyan algoritmust, amely elvégzi az újraírást.
4. A metamodellek és a szabályok ismeretében a szabályokra hibaellenőrzést végezni.
5. Kiterjeszteni a már létező matematikai eredményeket metamodell alapú újraírásra.
6. A matematikai eredmények fényében megvizsgálni a szabályok párhuzamos futtatásának lehetőségeit.

A fenti célok megvalósításával lehetővé válik a grafikus modellező nyelvek egyszerű megalkotása, grafikus feldolgozása, a feldolgozási szabályok párhuzamos futtatása, illetve sorrendjének optimalizálása. Ez lehetőséget ad például mobiltelefonok szoftverének gyors és automatikus generátorokkal támogatott fejlesztésére, beágyazott rendszerek állapotdiagramjainak futtatható programmá alakítására és még számos alkalmazásra.

2. Módszertani összefoglalás

Mivel a kitűzött célok megvalósíthatósága, az elméleti eredmények alkalmazhatósága, valamint az újabb problémákat felvető gyakorlati visszacsatolás a szoftverrel foglalkozó tudományok szempontjából alapvető, ezért a BME Automatizálási és Alkalmazott Informatikai Tanszékén kifejlesztettük a **Visual Modeling and Transformation System** (VMTS) [21] nevű szoftvercsomagot, amelyen a fenti **empirikus szempontok**at vizsgáltam. Ez a megközelítés az egész kutatást jellemezte. A VMTS tehát a kutatás **gyakorlati eredmény**eként fogható fel. Hasonló jellegű segítséget nyújtottak az elméleti eredmények kialakításában, a feladatok felvetésében a GME [Lédeczi et al. 2001] és a GReAT [Karsai et al. 2003] rendszerek is.

Az **elméleti eredmények**kel foglalkozó kutatás módszereit elsősorban az adott problémakörhöz rendelkezésre álló matematikai formalizmus és a hozzá tartozó módszerek határozták meg, ezeket az alábbiakban részletezem.

Ha az UML osztálydiagram (metamodell) elemeit szerepeltetjük a DPO megközelítés szabályainak bal és jobb oldalán, akkor az LHS egy példányosítását kell megtalálnunk a transzformálandó gráfban, illetve az újraírás lokális eredménye az RHS példányosítása. Mivel a

részgráfkeresést felváltotta a példányosítás feladata, egy új algoritmus kidolgozása szükséges. Ennek alapjaként a VF2 gráf izomorfizmust kereső algoritmust [Cordella et al. 1999] [Cordella et al. 2001] [Foggia et al. 2001] választottam, amit kiterjesztettem példányosítás keresésére, valamint annak végrehajtását optimalizáló lépéseket vezettem be. Ennek keretében megvizsgáltam az UML szabványban [OMG UML] leírt példányosítás alapvető tulajdonságait, beleértve a megengedett kombinációkat a példányszinten. Itt a **gráfelmélet** (halmazelmélet) formalizmusait használtam. Mind a metamodellek, mind a modellek jól modellezhetők irányított, címkézett gráfokként, ahol a címke tartalmazza a típusinformációt, a példányosítás összerendelését, az attribútumokat és az elméleti szempontból lényegtelen megjelenítési információt. A megengedett példányosítások kiszámítására vonatkozó eredmények esetében a gráf reprezentáció nehézkesnek bizonyult, ezért áttértem **lineáris algebra**i formalizmusra, nevezetesen a gráf illeszkedési mátrix reprezentációjára, amelyet kiterjesztettem a multiplicitás értékekre, így a második tézisben szereplő eliminációs algoritmus lineáris algebrai formalizmust használ. Ez az áttérés nem szokatlan a gráfelméleti problémák megoldása során, mivel bizonyos esetekben egyszerűsíti az algoritmust. Mivel a multiplicitások pozitív egész számok, egyszerű **számelmélet**i összefüggéseket is felhasználtam.

Mivel a metamodell rendelkezésre állása többletinformációt jelent, érdemes megvizsgálni, hogy miként lehet azt kihasználni a szabályok topológiájának ellenőrzésében. A már létező formális modellek használata segíthet ebben feltéve, ha kiterjesztjük őket a metamodellt is tartalmazó esetre. A harmadik tézis kapcsolatot teremt a már létező elméleti modellek és a metamodell között, valamint feltételeket ad meg a szabályok topológiai helyességét illetően. Ezt lényegében **gráfelmélet**i megfontolások és eszközök alapján teszi, ugyanakkor a homomorfizmus fogalmát és a gráf homomorfizmust az **absztrakt algebrá**ból kölcsönzi. Ezekre az eredményekre azért van szükség, mert a harmadik tézis bizonyításainak formalizmusa, a kategóriaelmélet, már nem vizsgálja az objektumok belső szerkezetét, csak feltételeket ír elő rájuk, amit csak gráfelméleti és absztrakt algebrai eszközökkel lehet bebizonyítani.

A negyedik tézis szemléletmódját a másik két elméleti tézistől eltérően a **kategóriaelmélet** adja. Ugyanis a már rendelkezésre álló matematikai háttér, nevezetesen a DPO megközelítés, kategóriaelméleti szemléletet tükröz. Ugyanakkor némely bizonyítás felhasznál **absztrakt algebra**i eredményeket is.

3. Új tudományos eredmények

Az ebben a fejezetben megjelenő eredményeket egyfelől a szigorú matematikai formalizmus biztosította egzakt eredmények elérése motiválta, másfelől az eredményeknek a közvetlen gyakorlati alkalmazhatóság követelményének is eleget kellett tenniük. Az előbbit a különböző formális modellek segítségével értem el, az utóbbi szempontot egy, az eddigieknél kifejezőbb vizuális transzformációs eljárás biztosította, amelyet egy elkészült szoftvercsomag alkalmaz. Új tudományos eredményeimet az alkalmazott formalizmus és feladat alapján csoportosítva négy tézisben fogalmaztam meg.

Az első tézis a VMTS által biztosított gyakorlati eredményeket tartalmazza. A második tézis eredményeként egy metamodell alapú mintaillesztő algoritmust dolgoztam ki. Az algoritmust egyszerűsítette az érvényes példányosításban szereplő objektumok számának ismerete, amelynek meghatározására kidolgoztam az IMM (Incidence Matrix with Multiplicity) algoritmust, és beláttam annak helyességét. A példányosításnak az algoritmusokban alkalmazható tulajdonságait szintén az első tézis keretein belül bizonyítottam. A harmadik tézis a példány és a metagráf közötti kapcsolatot alakítja homomorfizmussá. Az első két altézisben a típustartó homomorf leképezésre két feltételt bizonyítottam be. Ezek után megadtam egy algoritmust, amely tetszőleges metagráfra megad egy új metagráfot, amelyben a példányosítás inverze tetszőleges példány gráfjára már homomorf leképezés. Egy, a gyakorlatban is jól használható ekvivalencia definíció alapján megmutattam, hogy az algoritmus eredményeként létrejövő metagráf ekvivalens

az eredeti metagráffal. Az ellenkező irányra vonatkozó homomorf leképezést nem lehet közvetlenül elérni, ezért egy dekompozíciós algoritmust adtam meg, amelynek eredménygráfjaira már homomorf a típust hozzájuk rendelő leképezés. A negyedik tézis általánosítja a DPO megközelítést: bebizonyítja, hogy bizonyos feltételek teljesülése esetén a kategóriaelméleti diagramokon felvett metatégalalapok dupla pushoutot alkotnak, nem sértik meg a lógó él feltételt. Végül belátom a párhuzamosság tételének általánosítását metamodell alapú szabályokra.

1. Tézis

Új szoftver keretrendszert (Visual Modeling and Transformation System, VMTS) terveztem moduláris felépítésű architektúrával. Ezen kívül a VMTS szoftvercsomagon keresztül megmutattam a következőket:

- Az UML 2.0 szabványban leírt osztálydiagram, használati eset diagram és állapotdiagram megvalósítható n-rétegű metamodellezési technikával, egy példányosítás relációt felhasználva, amely a MOF M0 és M1 rétegek között található.
- Funkciómodellek (feature models) és erőforrásszerkesztők szintén megvalósíthatók a fenti technikákkal.
- Az UML osztálydiagram alkalmazható mint a transzformációs szabályok jobb- és baloldalát leíró nyelv, és ez a megoldás jól használható gyakorlati alkalmazásokban.
- A metamodellező környezetek és a gráftranszformációs rendszerek szoros integrációja a szabályok leírásában és a transzformálandó modellek létrehozásában jól alkalmazható a modelltranszformáció területén.
- A VMTS segítségével illusztráltam az elméleti eredményeket.

Az 1. tézishez kapcsolódó publikációk:
[2][4][8][16][17][[19][20][21][22][23][24][25][28][29][30].

A VMTS szoftvert nemcsak illusztrációs célra használtam fel, hanem számos tanszéki K+F projekt alapját is képezte. Ezek keretében a VMTS-t számos kiegészítő modullal láttuk el, amelyek az alábbi célokat szolgálják:

- Generatív technikák támogatása
- Kódgenerálás UML osztály- és állapotdiagramból
- Mobil alkalmazások fejlesztésének támogatása

2. Tézis

Kidolgoztam egy algoritmust, amely egyértékű multiplicitásokat tartalmazó metamodellre meghatározza az érvényes példányosításban szereplő típusonkénti objektumok számát, bebizonyítottam az algoritmus helyességét. Az algoritmus szűrőfeltételként használható fel a metamodell alapú illesztés során. Kiterjesztettem továbbá a VF2 algoritmust a metamodell alapú mintaillesztésre, heurisztikákat vezettem be.
A 2. tézishez kapcsolódó publikációk: [5][7][9][12][13][14][15][19][21].

2.1. Definíció (Címkézett gráf):
Legyen adott két rögzített alfabeta, Ω_V és Ω_E rendre a csomópontok és az élek számára. Akkor a *címkézett gráf* (labeled graph) egy hatos:

$$G = \left\langle G_V, G_E, s^G, t^G, lv^G, le^G \right\rangle, \tag{1}$$

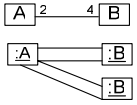
ahol G_V a csomópontok halmaza, G_E az élek halmaza, s^G ill. t^G a forrás és a cél függvényei $G_E \to G_V$, amelyek rendre az élekhez hozzárendelik azok kezdőpontját, illetve végpontját, valamint a címkéző függvények lv^G: $G_V \to \Omega_V$ és le^G: $G_E \to \Omega_E$, amelyek címkéket rendelnek az egyes csomópontokhoz ill. élekhez.

2.1. Altézis
Bebizonyítottam, hogy az 1. ábrán látható metamodell *na'* számosságú *A* típusú, illetve *nb'* számosságú *B* típusú objektumokkal példányosítható, ahol *n* egy tetszőleges pozitív szám, *a* és *b* az ábrán megjelenő multiplicitások, *GCD* jelöli a legnagyobb közös osztót.

$$a' = \frac{a}{GCD(a,b)} \qquad b' = \frac{b}{GCD(a,b)} \tag{2}$$

1. ábra. Asszociáció általános nemnulla egész multiplicitás értékekkel

Az 2.1. altézisben kimondottak sokszor szükségessé teszik, hogy párhuzamos éleket is alkalmaznunk kelljen. Ezt mutatja a 2. ábra.



2. ábra. Párhuzamos linkek

Amennyiben párhuzamos éleket nem engedünk meg, az egyes számosságok is szigorúbbak lesznek, amint azt a 2.2. altézis mutatja.

2.2. Altézis
Beláttam, hogy - a 2.1. altézis jelöléseit meghagyva - az 1. ábrán látható metamodell *na* számosságú *A* típusú, illetve *nb* számosságú *B* típusú objektumokkal példányosítható, ahol *n* egy tetszőleges pozitív szám.

Negatív eredmény, hogy általános esetben *n*-re csak egy kedvezőtlen felső határ adható meg, ugyanis létezik olyan konstrukció, ahol legalább az asszociáció egyik oldalán szereplő osztály összes példányosítását meg kell vizsgálnunk. Az értekezés konkrét konstrukcióval bizonyítja, hogy létezik olyan metamodell-modell páros, amely eléri ezt a felső határt.

2.3. Altézis
Megmutattam, hogy általános esetben a példányosítás tényének eldöntéséhez megvizsgálandó objektumok száma a legrosszabb esetben

$$\#A+\#B, \tag{3}$$

illetve a legnagyobb példányosításhoz tartozó *n* érték:

$$n = \min\left[\frac{\#A}{a}, \frac{\#B}{b}\right], \tag{4}$$

ahol $\#A$ az *A* típusú, $\#B$ pedig a *B* típusú objektumok számát jelöli.

Miután megvizsgáltam az egy asszociációt tartalmazó esetet, az ott kapott eredményekben megfogalmazott kényszereket egy általános, tetszőleges számú asszociációt tartalmazó esetre is általánosítottam. Ehhez alkalmassá tettem az illeszkedési mátrixot multiplicitások kezelésére is (Incidence Matrix with Multiplicity, IMM). Ezt a formalizmust az őt létrehozó algoritmussal adtam meg (2.1. algoritmus).

2.1. Algoritmus CREATEIMM( )
1 Járjuk be a metamodel gráfot!
2 Vegyük az összes e_j élet!
3 Ha v_k és v_l pontok illeszkednek az e_j élre, az IMM_{kj} legyen e_j v_k oldali multiplicitása, míg IMM_{lj} legyen az e_j v_l oldali multiplicitása.

Áttérve a lineáris algebrai reprezentációra kifejlesztettem egy eliminációs algoritmust, ami hibával terminál, ha az adott IMM által reprezentált metamodell nem példányosítható, illetve megadja minden változóra azt a pozitív egész számot, amelynek pozitív egész számú többszörösei vehetnek részt a metamodell példányosításában (2.2. algoritmus).

2.2. Algoritmus ELIMINATION(IMM IMM)

```
 1 for each j oszlopindex
 2   r0=az első nemnulla elemet tartalmazó sor indexe
 3   r1 = a második nemnulla elemet tartalmazó sor indexe
 4   if nem létezik r0 vagy r1 then break
 5   lcm=LCM(imm[r0,j],imm[r1,j])
 6   MultiplyRow(r0, lcm/imm[r0,j])
 7   MultiplyRow(r1, lcm/imm[r1,j])
 8   for each j2 oszlopindex
 9    if imm[r0,j2]= = imm[r1,j2] then
10      imm[r1,j2]=0
11    else if imm[r0,j2]!=0 and imm[r1,j2]!=0 then
12      Inkonzisztens párhuzamos utak. Nem létezik példányosítás.
13    else if imm[r1,j2]!=0
14      imm[r0,j2]=imm[r1,j2];
15      imm[r1,j2]=0;
16    end if
17  end for
18 end for
19 rowlcm= az imm nulladik sorának legkisebb közös többszöröse
20 for each j oszlopindex
21   imm[0,j]=rowlcm/ imm[0,j]
22 end for
```

Az algoritmust a VMTS rendszerben implementáltuk, forráskódja elérhető a [VMTS] irodalomjegyzékben megadott módon. Megmutattam, hogy az IMM algoritmus megadja az összes lehetséges példányosításban résztvevő objektumok számát, és a működése helyes.

2.4. Altézis
Beláttam, hogy az eliminációs algoritmus egyes lépéseiben létrejött IMM mátrix által reprezentált egyenletrendszer megegyezik a kiindulási egyenletrendszerrel, amely a 2.2. altézisben foglalt egyenletek felírását jelenti minden egyes asszociációra. Továbbá nincs olyan érvényes példányosítás, amely nem szerepelne az eliminációs algoritmus eredménykombinációi között.

Megvizsgálva a már létező algoritmusokat, a VF2 algoritmust választottam a saját mintaillesztési algoritmusom alapjának. Mivel többletinformációként a metamodell is rendelkezésre áll, egyfelől a VF2 algoritmust ki kellett egészítenem a típusok egyezésére vonatkozó kényszerekkel, másfelől néhány, az esetek többségében hatékony heurisztikát tudtam alkalmazni a mintaillesztés gyorsítására. Ezek az algoritmusok legrosszabb esetben az eredeti VF2 algoritmust adják vissza, de sohasem lesznek annál kevésbé hatékonyak. A VMTS felületén ezen a heurisztikákat explicit módon is igénybe lehet venni, így a hatékonyságuk a tervező által biztosított.

A fejezet további összefüggéseiben G^H jelöli a gráfot, amelyen a mintaillesztést végre kell hajtani, G^{LHS} a mintagráf. A gráfok élhalmazát E, a csomóponthalmazát V jelöli, a felső index a megfelelő gráfra utal. Az $n \in G^H, m \in G^{LHS}$ két csomópont, amelyet illeszteni akarunk egymáshoz, az $M \subseteq V^H \times V^{LHS}$ egy leképezés a minta, illetve G^H egy részgráfja között. A felső index a leképezés megfelelő oldalát jelöli. Az *x*,*y* elem multiplicitását egy él multihalmazban $\mu^{MS}(x,y)$ jelöli. A $pred(G,x)$ illetve a $succ(G,x)$ rendre az *x* csomópont irányított élen elérhető szomszédainak halmazát jelenti, a *pred* az *x* pontba befele mutató élek által elért szomszédokat, míg a *succ* a kifele mutató éleken keresztül elérhető szomszédokat tartja nyilván. A *typeof* függvény az elem típusát adja vissza. A *T* halmazok az aktuális *m*, ill. *n* csomópontokkal szomszédos csomópontok halmazát jelentik, ahol az *out* index a kifelé mutató élek által elérhető szomszédokat, míg az *in* index a befelé mutató éleken keresztül elérhető szomszédokat jelentik. Az alsó index a gráfra utal, amelyek tartalmazzák a fenti csomópontokat és éleket.

Az élek típusegyezésére az alábbi kényszereket vezettem be:

$$m' \in M^{LHS} \wedge \exists n' | (n',m') \in M, n' \in pred(G^H,n) \wedge \forall (typeof(n,n')\mu^{E^{LHS}}(m',m) \le \mu^{E^H}(n',n) \wedge typeof(m',m) = typeof(n'n) \quad (5)$$

$$m' \in M^{LHS} \wedge \exists n' | (n',m') \in M, n' \in succ(G^H,n) \wedge \forall (typeof(n,n')\mu^{E^{LHS}}(m,m') \le \mu^{E^H}(n,n') \wedge typeof(m,m') = typeof(n,n') \quad (6)$$

A VF2 algoritmus kényszerellenőrző részét az alábbi algoritmussal cseréltem ki (2.3. algoritmus):

2.3. Algoritmus

```
SIMPLE_FEASIBLE (Mapping M, Node n, Node m)
1 if not typeof(n)=typeof(m) return false;
2 for each predecessor m' of m
3       do if not Eq. (5)
4               return false
5       end if
6 end for
7 for each successor m' of m
8       do if not Eq. (6)
9               return false
10      end if
11 end for
```

Az új algoritmusnak a MetaVF2 nevet adtam. A VF2 algoritmus az illesztésre jelölt párok összeállító részét az alábbira módosítottam (2.4. algoritmus):

2.4. Algoritmus

COMPUTE_CANDIDATE_PAIRS()

1 **if** $T_H^{out} \neq \emptyset \wedge T_{LHS}^{out} \neq \emptyset$ **then** $P = T_H^{out} \times \{\min T_{LHS}^{out}\}$

2 **else if** $T_H^{in} \neq \emptyset \wedge T_{LHS}^{in} \neq \emptyset$ **then** $P = T_H^{in} \times \{\min T_{LHS}^{in}\}$

3 **else** $P = \text{Instanceof}(\min(V^{LHS} - M^{LHS})) \times \{\min(V^{LHS} - M^{LHS})\}$

2.5. Altézis

Bebizonyítottam, hogy a COMPUTE_CANDIDATE_PAIRS() függvény fenti módosítása legrosszabb esetben a MetaVF2 algoritmust adja.

Természetesnek tűnik, hogy a metamodellben való keresés miatt mindig a legkevesebb példánnyal rendelkező metamodell elemet válasszuk ki. Ez azonban ronthatja a teljesítményt.

2.6. Altézis

Beláttam, hogy ha a legtöbb példánnyal rendelkező V^{LHS} metamodell elemet választjuk, előfordulhat, hogy az eredeti algoritmusnál is több lépésben kapjuk meg az eredményt.

A fenti eredmények segítségével létrehoztam egy algoritmust, amely elvégzi a célkitűzésben megfogalmazott példányosítással egybekötött mintaillesztést. Ez az algoritmus alkotja a VMTS rendszer VMP-ket futtató motorjának lényegi alapját.

3. Tézis

Bebizonyítottam, hogy metamodell alapú szabályokra topológiai ellenőrzés végezhető homomorf metamodellek kompatibilitása mentén. Elégséges, valamint szükséges és elégséges feltételt adtam a homomorf példányosítás létezésére. Bebizonyítottam, hogy minden metamodellhez létezik egy ekvivalens metamodell, amelyhez tartozó *typeof* leképezés már homomorf. Beláttam, hogy a példány felbontható olyan nem izomorf részgráfokra, amelyekhez vezető példányosítás összerendelés homomorfizmus.
A 3. tézishez kapcsolódó publikációk: [3][6][9][18][21][30].

Ebben a tézisben elméleti alapot adtam arra, hogy a metamodellekkel megadott transzformációs lépések topológiáját ellenőrizni lehessen a bemeneti illetve a kimeneti modellek metamodelljeit alapul véve. Másrészről a már meglévő, kategóriaelméleti eszközöket használó újraírási módszerek (DPO) metamodellekre való általánosítását készíti elő. Mivel a **Graph** kategória nyilai gráf homomorfizmusok, a példányosítás leképezést is homomorfizmusokká kell alakítani.

3.1. Definíció (Gráf homomorfizmus irányított és címkézett gráfokra):
Az $f: G \to H$ *gráf homomorfizmus* egy olyan $f = \langle f_V : G_V \to H_V, f_E : G_E \to H_E \rangle$ függvénypár, amely megőrzi a kezdőpontokat, végpontokat és a címkéket, vagyis kielégíti az alábbi feltételeket:

- $f_V \circ t^G = t^H \circ f$
- $f_V \circ s^G = s^H \circ f_E$
- $lv^H \circ f_V = lv^G$
- $le^H \circ f_E = le^G$

3.2. Definíció (Metamodellek ekvivalenciája, teljes és részleges kompatibilitása):
Legyen *Meta1* és *Meta2* két irányított címkézett gráf, mely eleget tesz az UML osztálydiagram formai követelményeinek. *Meta1* és *Meta2* akkor és csak akkor ekvivalens, ha az összes UML objektumdiagram, amely példányosítja *Meta1*-et, példányosítja *Meta2*-t is, és fordítva. Ha csak egy irány teljesül, nevezetesen hogy a *Meta1* összes példánya példányosítja *Meta2*-t, akkor azt mondjuk, hogy *Meta1* kompatibilis *Meta2*-vel. Ha *Meta1* legalább egyetlen példánya példányosítja *Meta2*-t, akkor *Meta1* részlegesen kompatibilis *Meta2*-vel.

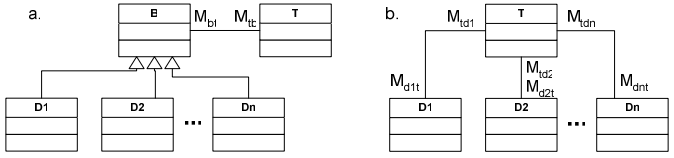
A példányosítást tekintve bebizonyítottam az alábbi altéziseket:

3.1. Altézis
Legyen a *Meta* gráf egy UML osztálydiagram. Legyen az *Instance* gráf a *Meta* gráf példányosításával nyert osztálydiagram. Ha a *Meta* gráf nem tartalmaz általánosítást (öröklést), és így szemantikai megfontolások miatt absztrakt osztályokat sem, akkor létezik típustartó homomorf leképezés az *Instance* gráfból a *Meta* gráfba, és ez a típustartó leképezés a csomópontokra nézve egyértelmű.

3.2. Altézis
Legyen a *Meta* gráf egy UML osztálydiagram. Legyen az *Instance* gráf a *Meta* gráf példányosításával nyert osztálydiagram. Megmutattam, hogy az *Instance* gráfból a *Meta* gráf akkor és csak akkor kapható meg típustartó homomorf leképezések sorozatával, ha az öröklési hierarchia különböző szintjein szereplő osztályokhoz tartozó asszociációk közül nem példányosul egyszerre egynél több különböző hierarchiabeli osztályhoz tartozó asszociáció.

A 3.1.-3.2. altéziseknek közvetlen következménye, hogy az általános UML osztálydiagramot általánosságban nem lehet gráf homomorfizmusokkal példányosítani. Ezért a következő lépésben egy olyan metamodellt konstruáltam, ami gyakorlati szempontokat és kifejezőképességét figyelembe véve megegyezik az eredeti metamodellel, de létesíthető típustartó homomorfizmus a példány modell és a metamodell között. Ehhez bevezettem egy ekvivalencia fogalmat (3.2. definíció), amely a gyakorlatban felvetődő ekvivalencia feltételeket fogalmaz meg. Ezek után megadtam egy algoritmust (3.1. algoritmus), amely az eredeti metamodellből egy olyan, az eredetivel kompatibilis metamodellt hoz létre, amelynél a példányosítás kapcsolat inverze homomorfizmus. Az algoritmus egy lehetséges bemenetére és kimenetére a 3. ábra mutat példát.

3.1. Algoritmus (vázlat)
1 Járjuk be az öröklési hierarchiát és másoljuk az összes öröklött adatot (attribútumokat és asszociációkat) a leszármazott osztályokba. Az új multiplicitások határait új változók bevezetésével jelöljük ki, és a korlátok összegének ki kell adnia az eredeti asszociációk összegét, amit kényszerekkel specifikálunk.
2 Távolítsuk el az öröklés kapcsolatokat.
3 Távolítsuk el az absztrakt osztályokat.



3. ábra. Generátor metamodell (a) és a hozzá tartozó homomorf metamodell (b)

A 3.1. algoritmusról beláttam az alábbi altézist:

3.3. Altézis
Bebizonyítottam, hogy a 3.1. algoritmus bemeneteként adódó generátor metamodell és az algoritmus kimeneteként adódó homomorf metamodell ekvivalens.

3.4. Altézis
Legyen *Meta1* és *Meta2* két címkézett, irányított gráf, melyek eleget tesznek az UML osztálydiagram formai követelményeinek. Erre az esetre beláttam a következő állításokat:
Ha *Meta1* kompatibilis *Meta2*-vel, *Meta1* (nem mindig összefüggő) részgráfja *Meta2*-nek. A multiplicitásra igaz az alábbi összefüggés:

$$\mathrm{M}_{\mathrm{Meta1}} \subseteq \mathrm{M}_{\mathrm{Meta2}}, \tag{7}$$

ahol $\mathrm{M}_{\mathrm{Meta1}}$ és $\mathrm{M}_{\mathrm{Meta2}}$ a *Meta1* illetve *Meta2* oldalon megengedett multiplicitások halmazai az azonos topológiai pozíciókban.
Részleges kompatibilitás esetén elég, ha az alábbi feltételek teljesülnek minden egyes multiplicitás párra:

$$\mathrm{SupM}_{\mathrm{Meta1}} \le \mathrm{SupM}_{\mathrm{Meta2}}, \tag{8}$$

$$\mathrm{M}_{\mathrm{Meta1}} \cap \mathrm{M}_{\mathrm{Meta2}} \neq \varnothing \tag{9}$$

ahol *Sup* jelöli a megengedett multiplicitásokat tartalmazó halmaz supremumát.

3.5. Altézis
Igazoltam, hogy minden I példánygráf felbontható olyan $I_1, I_2, ..., I_K$ nem izomorf részgráfok sorozatára, hogy a példányosítás kapcsolat az asszociációvégeken nulla multiplicitást nem tartalmazó homomorf M metagráf és az $I_k, (k = 1...K)$ gráfok között gráf homomorfizmus legyen. Feltettük, hogy az I gráf példányosítja az M gráfot.

3.6. Altézis
A 3.5. altézisben foglalt felbontásra megadtam egy algoritmust, amelyről bebizonyítottam, hogy bonyolultsága a példány- és a metagráf élszámának függvényében $O(\#E(I)\#E(M))$.

Az ebben a tézisben ismertetett eredmények lehetővé teszik a szabályok topológiájának ellenőrzését a bemeneti és a kimeneti modell előre megadott metamodelljével összevetve (ez az eszközök szempontjából igazán fontos tulajdonság), a későbbi kategóriaelméleti megközelítést, valamint új összefüggést tartalmaz a kompatibilitás és a részgráf izomorfia között. A tézis alkalmazását akkor találtam leghatékonyabbnak, amikor automatikusan teljesülnek a 3.2 altézisben megfogalmazott feltételek, ebben a speciális esetben a tartalmazás miatt gyors mintaillesztő és kényszerellenőrző algoritmusok konstruálhatók.

4. Tézis

A DPO megközelítés eredményeire építve bebizonyítottam, hogy a metamodell alapú szabályok esetén, ha a metamodell elemeknek nincs közös elemük, akkor a szabályok végrehajtási sorrendje közömbös, és párhuzamosan futtathatók; illetve ha két metamodell alapú szabály végrehajtási sorrendje felcserélhető, akkor azokat párhuzamosan futtatva az egyik szabály nem változtatja a másik szabály által érintett átírt területet. Megmutattam továbbá, hogy a fenti állítás visszafelé is igaz.

Az itt bebizonyított eredmények kategóriaelméleti szemlélete alapvetően különbözik a két előző tézis gráfelméleti, ritkán algebrai megközelítésétől. Sajnos a már ismert módszerek (DPO, SPO) sem tudnak *tisztán* kategóriaelméleti módszereket alkalmazni, bár az uralkodó szemlélet és

az általánosítások (pl. High-Level Replacement Systems) egyértelműen a kategóriaelmélet irányába mutatnak. Ezért a bizonyításokban a kategóriaelméleti eszközöket kiegészítendő absztrakt algebrai elemeket is felhasználtam.

A DPO megközelítés egyik fő eredménye a szabályok párhuzamosítására, illetve a szabályok felcserélésére kimondott tételek. A VMTS rendszerben a párhuzamos alkalmazás lehetőségét sokkal fontosabbnak tartottam, ezért ebben a tézisben két alapvető dolgot mutattam meg: (i) a DPO megközelítés bizonyos esetekben kiterjeszthető a metamodell alapú esetre, és ekkor a metamodellek sem sértik meg a DPO feltételeket, valamint (ii) hogy miként alkalmazható a DPO megközelítés párhuzamosságra vonatkozó tétele metamodellekből álló LHS esetén. A 4. tézishez kapcsolódó publikációk: [3][6][9][21][30].

4.1. Definíció (*Graph* kategória)

- Objektumok: irányított és címkézett gráfok
- Nyilak: gráf homomorfizmusok
- Kompozíciós operátor: gráf homomorfizmusok kompozíciója.
- Azonosság: az identitásfüggvény mind a csomópontok mind az élek számára.

4.2. Definíció (Gráf produkciós szabály – DPO megközelítés):

A gráf produkciós szabály $p:(L \xleftarrow{l} K \xrightarrow{r} R)$ egy nyílpár a **Graph** kategóriában, amely a következőkből áll:

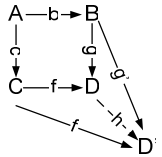
- a produkciós szabály nevéből
- és egy injektív gráf homomorfizmus párból: $l: K \to L$, és $r: K \to R$

4.3. Definíció (pushout, pushout komplemens):

Adott egy **C** kategória, és két nyíl: b: A→B, c: A→C. A ⟨D, g: B→D, f: C→D⟩ hármast a ⟨b, c⟩ **pushout**jának nevezzük, ha

1. g o b = f o c
2. Az összes D' objektumra valamint g': B→D', és f': C→D' nyilakra, ha g' o b= f' o c, létezik olyan egyértelmű h: D→D' nyíl, hogy h o g = g' és h o f = f'.

Az alábbi szituációban (4. ábra) a $\langle C, c: A \to C, f: C \to D \rangle$ hármast a $\langle b, g \rangle$ **pushout komplemensének**, a C-t pedig a $\langle b, g \rangle$ **pushout komplemens objektumá**nak nevezzük.



4. ábra. Pushout illusztrálása

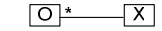
A további állítások felhasználják a 6. ábrán látható kategóriaelméleti diagramot, ahol *G* a bemeneti gráf, *H* a kimeneti gráf, az *M* végződésű jelölések a metamodellekre utalnak.

4.4. Definíció (helyes metamodellek)

Ha az LM és az RM kielégítik a részleges kompatibilitás feltételeit, GM-et, illetve HM-et illetően, akkor az LM és az RM metamodeleket **helyes metamodell**eknek nevezzük.

4.1. Altézis

Jelölje *O* a bementi gráf metamodell elemeinek a közös ősosztályát! Ha a metamodell alapú szabály baloldala tartalmazza az 5. ábrán szemléltetett részgráfot, a szabály eltávolíthatja az *X*-szel jelölt csomópontot a lógó él feltétel megsértése nélkül.



5. ábra. Részgráf a 4.1. altézishez

4.2. Altézis

Igazoltam, hogy ha a 6. ábrán látható diagramon a külső dupla téglalapban található morfizmusok tartalmazások, és a belső téglalapok dupla pushoutot alkotnak, akkor a külső téglalapok is dupla pushoutot alkotnak, ha az alábbi feltételek teljesülnek:

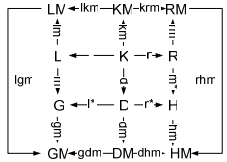
$$V_{DM} = V_{GM} \setminus (V_{LM} \setminus V_{RM}) \quad (10)$$

$$E_{DM} = E_{GM} \setminus (E_{LM} \setminus E_{RM}) \quad (11)$$

$$V_{DM} = V_{HM} \setminus (V_{RM} \setminus V_{LM}) \quad (12)$$

$$E_{DM} = E_{HM} \setminus (E_{RM} \setminus E_{LM}) \quad (13)$$

A 6. ábra a 4.2 definíció jelöléseit használja, valamint a belső téglalap a megfelelő részgráfokat a bemeneti (G) és a kimeneti (H) gráfban. Az „M" utótag az adott gráf metamodelljeit jelöli.



6. ábra. Illusztráció a 4.2. altézishez

4.3. Altézis

Megmutattam, hogy ha a transzformáció zárt a bemeneti metamodellre, és az L, K, R, G, D, H dupla téglalap (6.ábra) egy dupla pushoutot alkot, akkor a 7. ábrán látható diagramok is pushoutot alkotnak.

7. ábra. Illusztráció a 4.3. altézishez

4.4. Altézis

Bebizonyítottam, hogy ha a két szabály bal oldalának nincs közös elemük, akkor a szabályok egymás utáni végrehajtásának sorrendje közömbös, és párhuzamosan futtathatók.

4.5. Altézis

Beláttam, hogy ha két metamodell alapú szabály végrehajtási sorrendje felcserélhető, akkor azokat párhuzamosan futtatva az egyik szabály nem változtatja a másik szabály által érintett átírt területet. Továbbá megmutattam, hogy a fenti állítás visszafelé is igaz.

A 4. tézis matematikai jelentősége mellett formálisan megalapozott feltételeket ad arra, hogy az újraírási szabályokat párhuzamosan futtathassuk, amellyel a futtató hardvertől függően jelentős gyorsulás érhető el. Matematikai szempontból további, már bebizonyított tételek általánosíthatók automatikusan a metamodell alapú esetre.

4. Az új tudományos eredmények alkalmazása

Az új tudományos eredmények szoftvercsomagokban, illetve azok alkalmazásaiban jelennek meg. Mivel az eddigi esettanulmányok segítségével be tudjuk mutatni az alkalmazhatóságot az ipari szféra számára, egyre több az érdeklődés, így az alkalmazások sora valószínűsíthetően az itt felsorolt megoldásokkal még nem tekinthető lezártnak. Az alkalmazásokkal kapcsolatos publikációk (VMTS): [2][4][8][16][17][[19][20][21][22][23][24][25][28][29][30].

4.1. Szoftvercsomagok

Amint azt a módszertani összefoglalóban említettem, a feladatok felvetését, illetve az eredmények alkalmazhatóságát egy VMTS nevű szoftvercsomag jelentette. A fejlesztésben többek között *Lengyel László*, *Mezei Gergely* és *Angyal László* voltak a segítségemre, akik ezért számos cikkben társszerzőim is egyben.

A metamodell alapú újraírás sikerességét a GReAT transzformációs rendszer kezdeti verziói adták, amelyet a UDM [Magyari et al. 2003] keretrendszer fölé terveztünk és implementáltunk a Vanderbilt Egyetem ISIS kutatóintézetének tagjai segítségével. A [10][11][12][13] publikációk ezt a munkát tükrözik.

4.2. Alkalmazások

A metamodell-alapú, multiplicitással kiegészített újraírást számos gyakorlati esetben használtuk. Funkciómodellek (feature modell) normalizálása [17] a generatív programozás [Czarnecki & Eisenecker 2000] modellezésénél felmerülő probléma, amit a kifejlesztett módszerrel minimális időráfordítással meg lehet oldani. UML állapotdiagramból generálnak C++ kódot a [8][19]-ben részletezett eljárások, amelyek szintén a metamodell alapú, multiplicitással kiegészített újraírási technikát alkalmazzák transzformációs lépéseikben. UML osztálydiagramból való C# kód generálására mutat egy alkalmazást a [23] publikáció, valamint [19] a Quantum Framework [Samek 2002] alá generál kódot UML állapotdiagram bemenet alapján. Ezeket az alkalmazásokat a VMTS rendszerben valósítottuk meg.

Ahhoz, hogy a multiplicitással kiegészített metamodell alapú transzformáció egyáltalán megvalósulhasson, fel kell használni a második tézis eredményeit, amelyek a mintaillesztő algoritmushoz szükségesek. A harmadik tézis eredményeinek segítségével a szabályok alkalmazhatósága vizsgálható a bemeneti modell rendelkezésre állásával vagy anélkül, így a szabály írása közben képes a szabály esetleges hibáit felismerni. A negyedik tézis eredményeinek felhasználásával a transzformáció bemenetének rendelkezésre állása előtt képes a rendszer felismerni a párhuzamosítás lehetőségét illetve a szabályalkalmazás sorrendjének invarianciáját, amely további optimalizálást tesz lehetővé.

A fenti példák illusztrálják, hogy a VMTS alkalmas modellfordítóként, illetve modelltranszformáló rendszerként működni, és számos esetben az újraírást alkalmazó grafikus modelltranszformációs rendszerek szemléletesebb, könnyebben fejleszthető kiváltói lehetnek a gráfbejárást alkalmazó tradicionális modellfeldolgozó eljárásoknak.

A BME Automatizálási és Alkalmazott Informatikai Tanszéken folyó *Simplian* projekt célkitűzése egyszerű felhasználói felület létrehozásának támogatása Symbian operációs rendszerek alá. A felhasználói felület modellezése a VMTS csomaggal történik, a C++ kódgeneráló rész átírása VMP segítségével megtörtént. A fejlesztés során UML állapotdiagram bevonására is szükség van, aminek megoldhatóságát a VMTS már bizonyította.

5. Az értekezés témaköréből készült publikációk

Könyv:

[1] Bányász G, **Levendovszky T**, Linux Programozás,
Szak Kiadó, 2003, ISBN 963 9131 57 1

Könyvrész, könyvfejezet:

[2] Albert I, Balássy Gy, Charaf H, Erdélyi T, Horváth Á, **Levendovszky T**, Péteri Sz, Rajacsics T, A .NET Framework és programozása, Szak Kiadó, 2004, ISBN 963 9131 62 8

Folyóiratcikkek:

[3] **Levendovszky T**, Lengyel L, Charaf H, Implementing a Metamodel-Based Model Transformation System, Buletinul Stiintific al Universitatii "Politehnica" din Timisoara, ROMANIA Seria AUTOMATICA si CALCULATOARE PERIODICA POLITECHNICA, Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE Vol.49 (63), 2004, ISSN 1224-600X , pp. 89-95

[4] **Levendovszky T**, Angyal L, Charaf H, Software Design Trade-Offs in Highly Configurable User Interface Construction, ROMANIA Seria AUTOMATICA si CALCULATOARE PERIODICA POLITECHNICA, Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE Vol.49 (63), 2004, ISSN 1224-600X. pp. 95-101

[5] **Levendovszky T**, Lengyel L, Charaf H, A UML Class Diagram-Based Pattern Language for Model Transformation Systems, WSEAS Transactions on Computers, Issue 2, Volue 4, February 2005, ISSN 1109-2750, pp. 190-195

[6] **Levendovszky T**, Lengyel L, Charaf H, Extending the DPO Approach for Topological Validation of Metamodel-Level Graph Rewriting Rules, WSEAS Transactions on Information Science and Applications, Issue 2, Volue 2, February 2005, ISSN 1790-0832, pp. 226-231

[7] **Levendovszky T**, Charaf H, Pattern Matching in Metamodel-Based Model Transformation Systems, Periodica Polytechnica Electrical Engineering, to be pulished

[8] Lengyel L, Levendovszky T, Charaf H, Constraint Validation Support in Visual Model Transformation Systems, ACTA CYBERNETICA, to be pulished

[9] Fill HG, Geiger L, Zündorf A, **Levendovszky T**, Lengyel L, Mezei G, Charaf H, A comparison of three tool-based approaches for the practical realization of UML statechart diagrams, Invited Paper, Software and System Modeling, Springer, submitted

[10] Lengyel L, **Levendovszky T**, Aspektus-orientált programozás, Híradástechnika, Volume LIX, 2004/10, pp. 8-12

[11] Lengyel L, **Levendovszky T**, Introduction to Aspect-Oriented Programming, Híradástechnika, Volume LX. 2005, pp. 18-23

Nemzetközi konferencia-kiadványban megjelent idegen nyelvű előadás

[12] **Levendovszky T**, Karsai G, Maroti M, Ledeczi A, Charaf H, Model reuse with metamodel-based transformations, Lecture Notes in Computer Science - ICSR7, Austin, TX, April 18, 2002, pp.166-178

[13] Agrawal A, **Levendovszky T**, Sprinkle J, Shi F, Karsai G, Generative Programming via Graph Transformations in the Model-Driven Architecture, OOPSLA - Workshop on Generative Techniques in the Context of Model Driven Architecture, Seattle, WA, November 5, 2002.

[14] Sprinkle J, Agrawal A, **Levendovszky T**, Shi F, Karsai G, Domain Evolution in Visual Languages Using Graph Transformations, OOPSLA - 2nd Workshop on Domain-Specific Languages, Seattle, WA, November 4, 2002.

[15] Sprinkle J, Agrawal A, **Levendovszky T**, Shi F, Karsai G, Domain Translation Using Graph Transformations, Tenth IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Huntsville, AL, April 7, 2003., pp. 159-168

[16] Angyal L, **Levendovszky T**, Charaf H, Software Design with Generative Programming and Traditional Object-Oriented Techniques, microCAD, March 18-19, Miskolc, 2004, Miskolc. pp. 9-14

[17] Lengyel L, **Levendovszky T**, Charaf H, Supporting Round-Trip Engineering in Modeling Environments with the Application of Meta-Modeling Techniques, IASTED on SE, Innsbruck, 2004, pp. 178-182

[18] **Levendovszky T**, Charaf H, Parametrized Multiplicity Constraints in UML Like Metamodels, microCAD, March 18-19, Miskolc, 2004, Miskolc, pp. 287-292

[19] **Levendovszky T**, Lengyel L, Charaf H, Software Composition with a Multipurpose Modeling and Model Transformation Framework, IASTED on SE 2004, Innsbruck, 2004, pp. 590-594

[20] Lengyel L, **Levendovszky T**, Charaf H, Metamodel-Based Software Model Storage and Transformation System, microCAD, March 18-19, Miskolc, 2004, pp. 281-286

[21] **Levendovszky T**, Lengyel L, Mezei G, Charaf H, A Systematic Approach to Metamodeling Environments and Model Transformation Systems in VMTS, International Workshop on Graph-Based Tools (GraBaTs) Electronic Notes in Theoretical Computer Science, 2004, to be pulished

[22] Mezei G, **Levendovszky T**, Lengyel L, Charaf H, A Flexible Attribute Instantiation Technique for Visual Languages, IASTED on SE, February 15-17, 2005, Innsbruck, Austria, pp. 355-359

[23] Lengyel L, **Levendovszky T**, Charaf H, Constraint Handling in Feature Models, 5th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest, 2004, pp.279-290

[24] Lengyel L, **Levendovszky T**, Kozma P, Charaf H, Compiling and Validating OCL Constraints in Metamodeling Environments and Visual Model Compilers, IASTED on SE, February 15-17, 2005, Innsbruck, Austria, pp. 48-54

[25] Lengyel L, **Levendovszky T**, Charaf H, Weaving Crosscutting Constraints in Metamodel-Based Transformation Rules, 8th International Conference on Information Systems Implementation and Modeling, ISIM '05, April 19-20, 2005, Hradec nad Moravicí, Czech Republic, pp. 119-126

[26] Bogárdi-Mészöly Á, Imre G, **Levendovszky T**, Charaf H, Determining the Distribution of the Response Time in J2EE Web Applications, microCAD, March 10-11, Miskolc, 2005, pp.33-38

[27] Iváncsy R, **Levendovszky T**, Charaf H, .NET Facilities in Data Mining Applications, microCAD, March 10-11, Miskolc, 2005, pp. 167-172

[28] Lengyel L, **Levendovszky T**, Charaf H, Graph Transformation and Constraint Validation-Driven User Interface Handler Code, microCAD, March 10-11, Miskolc, 2005, pp. 267-272

[29] Lengyel L, **Levendovszky T**, Charaf H, Implementing a Metamodel-Based OCL-Compiler, microCAD, March 10-11, Miskolc, 2005, pp. 273-278

[30] **Levendovszky T**, Lengyel L, Charaf H, Creating a Homomorphic Instantiation Mapping Between Metamodels and the Model Parts, microCAD, March 10-11, Miskolc, 2005, pp. 279-283

[31] Mezei G, **Levendovszky T**, Lengyel L, Charaf H, Multilevel Metamodeling - A Case Study, microCAD, March 10-11, Miskolc, 2005, pp. 321-326

[32] D. Bisztray, **Levendovszky T**, Charaf H, Defining Feature Modeling-Based Constraints, microCAD, March 10-11, Miskolc, 2005, pp. 19-24

[33] Lengyel L, **Levendovszky T**, Charaf H, Managing Crosscutting Constraints in Metamodel-Based Model Transformation Frameworks, International Carpathian Control Conference, ICCC'2005, Miskolc-Lillafüred, Hungary, May 24-27, 2005

[34] Lengyel L, **Levendovszky T**, Charaf H, Implementing an OCL Compiler for .NET, .Net Technologies 2005, Plzen, May 2005

[35] Lengyel L, **Levendovszky T**, Charaf H, Aspect-Oriented Constraints in Metamodel-Based Model Transformation, 5th Internal Conference of PhD Students, Miskolc, Hungary, 14-20 August 2005, pp. 109-114

[36] Lengyel L, **Levendovszky T**, Charaf H, A Case Study: Supporting Metamodel-Based Graph Transformation with Aspect-Oriented Constraints, 5th Internal Conference of PhD Students, Miskolc, Hungary, 14-20 August, 2005 pp.115-120

6. Idegen hivatkozások

[12] publikációra hivatkozik:

Schloegel K, Oglesby D, Engstrom E, Towards Next Generation Metamodeling Tools, Second Workshop on Domain Specific Visual Languages, OOPSLA 2002

Tratt L, Clark T, Using Icon-derived technologies to drive model transformations, WiSME@UML'2003 - UML Workshop W2, San Francisco, USA, 2003

Tratt T, Clark T, Model transformations in Converge, Workshop in Software Model Engineering (WiSME) 2003

Appukuttan B, Clark T, Reddy S, Tratt L, Venkatesh R, A Pattern based model driven approach to model transformations, Metamodelling for MDA. University of York, UK. November 2003

Appukuttan B, Clark T, Reddy S, Tratt L, Venkatesh R, A model driven approach to building implementable model transformations,WiSME@UML'2003 - UML Workshop W2, San Francisco, USA, 2003

[13] publikációra hivatkozik:

Willink ED, A concrete UML-based graphical transformation syntax : The UML to RDBMS example in UMLX, Metamodelling for MDA. University of York, UK. November 2003

Boulet P, Dekeyser J-L, Dumoulin C, Marquet P, MDA for SoC Embedded Systems Design, Intensive Signal Processing Experiment" Model Driven Architecture in the Specification, Implementation and Validation of Object-oriented Embedded Systems, San Francisco, California, USA, 2003

Wagelaar D, Jonckers V, A Concept-Based Approach to Software Design. In proceedings of the 7th IASTED International Conference on Software Engineering and Applications (SEA 2003), Marina del Rey, USA, November 2003. ISBN/ISSN: 0-88986-394-6

Graw G, Herrmann P, Generation and Enactment of Controllers for Business Architectures Using MDA, Lecture Notes in Computer Science, Volume 3047, May 2004, Pages 148 – 166

Wagelaar D, Towards a Context-Driven Development Framework for Ambient Intelligence. In: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCS 2004 Workshops), IEEE Computer Society, 2004

Suvee D, Vanderperren W, Wagelaar D, There are no Aspects. Software Composition Workshop @ ETAPS 2004, Electronic Notes in Theoretical Computer Science, Elsevier Science, April 2004

[19] publikációra hivatkozik:

Cechticky V, Pasetti A, Rohlik O, Schaufelberger W, XML-Based Feature Modelling pp. 101-114, Software Reuse: Methods, Techniques and Tools: 8th International Conference, ICSR 2004, Madrid, Spain, July 5-9, 2009. Proceedings. Lecture Notes in Computer Science 3107 Springer 2004, ISBN 3-540-22335-5

7. Hivatkozott Irodalom

[Baresi & Heckel 2002] Baresi L, Heckel R, Tutorial Introduction to Graph Transformation: A Software Engineering Perspective, In Corradini, A.,H. Ehrig, H.-J. Kreowski, G. Rozenberg (Eds): Proc. 1st Int. Conference on Graph Transformation (ICGT 02), Barcelona, Spain, Volume 2505 of Lecture Notes in Comp. Science. Springer-Verlag, October 2002.

[Barr & Wells 1999] Barr M, Wells C, Category Theory Lecture Notes for ESSLLI, www.folli.uva.nl/CD/1999/library/pdf/barrwells.pdf, 1999.

[Blostein et al. 1995] Blostein D, Fahmy H, Grbavec A, Practical Use of Graph Rewriting, Technical Report No. 95-373, Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada, January, 1995.

[Cordella et al. 1999] Cordella LP, Foggia P, Sansone C, Vento M, Performance Evaluation of the VF Graph Matching Algorithm, Proc. of the 10th ICIAP, IEEE Computer Society Press, vol. 2, pp. 1038-1041, 1999.

[Cordella et al. 2001] Cordella LP, Foggia P, Sansone C, Vento M, An Improved Algorithm for Matching Large Graphs, Proc. of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, Ischia, May 23-25, pp. 149-159, 2001.

[Czarnecki & Eisenecker 2000] Czarnecki K, Eisenecker UW, Generative programming: methods, tools, and applications, Addison-Wesley, 2000

[Ehrig & Taenzer 1996] Ehrig H, Taentzer G, Computing by Graph Transformation. A Survey and Annotated Bibliography Technical Report, TU Berlin, No. 96-21, 1996

[Ehrig 1979] Ehrig H, Introduction to the Algebraic Theory of Graph Grammars, In:Graph Grammars and Their Applications to Computer Science and Biology, Springer, Ed. Claus V, Ehrig H, Rozemberg G, Berlin, 1979.

[Ehrig et al. 1999] Ehrig H, Engels G, Kreowski H-J, Rozemberg (ed.), Handbook on Graph Grammars and Computing by Graph Transformation: Application, Languages and Tools, Vol.2. World Scientific, Singapore, 1999.

[Foggia et al. 2001] Foggia P, Sansone C, Vento M, A Performance Comparison of Five Algorithms for Graph Isomorphism, Proc. of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, Ischia, May 23-25, pp. 188-199, 2001.

[Karsai et al. 2003] Karsai G, Agrawal A, Shi F, On the Use of Graph Transformations for the Formal Specification of Model Interpreters, Journal of Universal Computer Science, Volume 9, Issue 11, pp. 1296-1321, November, 2003

[Kleppe et al. 2003] Kleppe A, Warmer J, Bast W, MDA Explained: The Model Driven Architecture-Practice and Promise, Addison-Wesley, 2003, ISBN: 032119442X

[Lédeczi et al. 2001] Lédeczi Á, Bakay Á, Maróti M, Völgyesi P, Nordstrom G, Sprinkle J, Karsai G, Composing Domain-Specific Design Environments. IEEE Computer 34(11), pp. 44-51, November, 2001

[Magyari et al. 2003] Magyari E, Bakay A, Lang A, Paka T, Vizhanyo A, Agrawal A, Karsai G, UDM: An Infrastructure for Implementing Domain-Specific Modeling Languages, The 3rd OOPSLA Workshop on Domain-Specific Modeling, OOPSLA 2003, Anahiem, California, October 26, 2003.

[Mellor et al. 2004] Mellor SJ, Scott K,Uhl A, Weise D, MDA Distilled - Principles of Model-Driven Architecture, Addison-Wesley, ISBN: 0201788918, 2004

[OMG MDA] OMG Model Driven Architecture homepage, www.omg.org/mda/

[OMG MDAGuide] MDA Guide Version 1.0.1, OMG, document number: omg/2003-06-01, 12th June 2003, www.omg.org/docs/omg/03-06-01.pdf

[OMG UML 2003]Unified Modeling Language Specification, v1.5, www.uml.org

[Pender &Pender 2003] Pender T, Pender T, UML Bible, Wiley, 2003, ISBN: 0764526049

[Pierce 1991] Pierce BC, Basic Categoy Theory for Computer Scientists, MIT Press, 1991, ISBN 0-262-66071-7

[Rozenberg 1997] Rozenberg G (ed.), Handbook on Graph Grammars and Computing by GraphTransformation: Foundations, Vol.1 World Scientific, Singapore, 1997.

[Samek 2002] Samek M, Practical Statecharts in C/C++,CMP Books, 2002, ISBN: 1578201101

[VMTS] http://avalon.aut.bme.hu/~tihamer/research/vmts/