Budapesti Műszaki és Gazdaságtudományi Egyetem
Műszaki Tudományág, Informatikai Tudományszak

VIZUÁLIS NYELVEK TRANSZFORMÁCIÓALAPÚ TÁMOGATÁSA

Ph.D. értekezés tézisei

Mezei Gergely

Tudományos vezetők:
Dr. Charaf Hassan, Ph.D.
docens

Dr. Levendovszky Tihamér, Ph.D.
adjunktus

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
AUTOMATIZÁLÁSI ÉS ALKALMAZOTT INFORMATIKAI TANSZÉK

Budapest, 2007

Ph.D. értekezés tézisei

Mezei Gergely

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

1111 Budapest, Goldmann György tér 3.

e-mail: gmezei@aut.bme.hu
tel: +36(1)4631007
fax: +36(1)4633478

Tudományos vezetők:
Dr. Charaf Hassan, Ph.D.
docens

Dr. Levendovszky Tihamér, Ph.D.
adjunktus

1. Előzmények és célkitűzések

Az informatikai rendszerek komplexitásának növekedésével párhuzamosan a modellalapú szoftverfejlesztés is egyre népszerűbbé vált. A modellek magas absztrakciós szinten képesek leírni a felmerülő feladatokat és azok megoldásait. Ez a magas absztrakciós szint biztosítja, hogy a problématér lényegi részeire koncentrálva lehessen megalkotni, kidolgozni a kívánt rendszereket, miközben a kevésbé lényeges, másodrangú, alárendelt információk a háttérbe szorulnak. A magas absztrakciós szint, valamint a modellezésben gyakran használt grafikus megjelenítés nagymértékben elősegíti a komplexitásból fakadó problémák leküzdését.

Egy problémához több különböző modell is készíthető attól függően, hogy a valóság leképzése során milyen absztrakciós szintet választunk, milyen eszközrendszert (modellezési nyelvet, módszertant) használunk. Ebből kifolyólag a modellek egyértelműsége, közérthetősége alapvető fontosságú. Az egyik leggyakrabban használt általános modellező nyelv, az Unified Modeling Language (UML) [UML] ezekre a problémákra ad megoldást. Az UML formális szintaxissal definiált vizuális nyelv, aminek segítségével a felmerülő problémákat könnyen átlátható formában lehet ábrázolni, dokumentálni.

Az UML ugyanazon eszköztárat nyújtja a feladat jellegétől függetlenül. Az egységesség növeli a modellek közérthetőségét, újrahasznosíthatóságát, viszont gyakran rugalmatlan, a modellezési terület sajátosságait figyelmen kívül hagyó leírásokat, modelleket eredményez. Napjainkban, többek közt az UML rugalmatlanságából kifolyólag, egyre népszerűbbé válnak a szakterület-specifikus modellezési nyelvek (Domain-Specific Modeling Language, DSMLs) [DSML]. A szakterületi modellezés fő gondolata, hogy egy-egy szakterület jellemzőit egybegyűjtve olyan nyelveket (nyelvcsaládokat) dolgozzunk ki, amelyek alkalmasak a kapcsolódó szakterületek problémáinak testreszabott leírására. Az így kapott nyelvekkel létrehozott modellek a szakterület keretein belül rugalmasabbak és a szakemberek számára könnyebben érthetőek, mint az általános nyelven megfogalmazottak.

A szakterület-specifikus nyelvek esetében nehézséget okoz, hogy míg a modellezési környezetek nagy részét egy konkrét nyelvcsalád támogatásához hozták létre, addig ebben az esetben szakterületenként különböző nyelvcsaládok támogatására van szükség. A probléma egyik gyors, felhasználóbarát és népszerű megoldása a metamodellezés [Nordstrom, 1999]. A metamodellezés esetében a modellezési nyelvek definícióját egy speciális modell, a metamodell tartalmazza. A metamodell definiálja a modellezési nyelvben felhasználható elemeket, az elemek attribútumait és az elemek közt kialakítható kapcsolatokat. A metamodell tehát a modell modelljének tekinthető. A metamodellből a modellt a példányosítás reláció segítségével kaphatjuk meg.

A példányosítás fogalma nem csak két modellezési szint között értelmezhető, hanem kiterjeszthető további szintekre, így létrehozhatunk pl. meta-metamodelleket, amik a metamodellek jellemzőit definiálják. A magasabb modellezési szintek természetszerűleg magasabb absztakciós szintet is jelentenek, azaz egy metamodell absztraktabb leírást tesz lehetővé, mint modelljei. Napjaink egyik legjelentősebb metamodellezési ajánlása, a Meta-Object Facility [MOF] kezdetben négyszintű modellhierarchiát definiált, ami a legtöbb probléma leírásához elegendő is. A modellezési rétegek ugyanakkor elméleti szinten bármilyen mélységben egymásra építhetőek. A rétegek számának növelésével egyre kisebb absztrakciós szint váltásra kényszerülünk a példányosítások során, így a modellezés tovább finomítható. Napjainkra a MOF egyre nyitottabbá vált a négyszintű hierarchia kibővítésére. Az n-szintű metamodellezési hierarchia azonban több problémát is felvet. A problémák egy része strukturális, topológiai jellegű, ezekről bebizonyosodott, hogy mind elméleti, mind gyakorlati szinten megoldhatóak [Levendovszky 20505]. A problémák fenn-

maradó része az attribútumokkal, az n-szintű attribútum-hierarchiával kapcsolatos, ezek megoldása jelenleg még nem megoldott.

A metamodell tehát képes rugalmasan leírni a szakterület-specifikus modellezőnyelvek topológiai felépítését. Problémát jelent ugyanakkor, hogy grafikus szakterületi nyelvek esetén azok jelölésrendszerét – a felhasználható modellelemek megjelenítését – a metamodellezési technika önmagában nem képes definiálni. A szakterületi nyelvek metamodellezéssel történő megadásához ezért szükség van egy, a megjelenítést, a vizualizáció definícióját leíró megoldásra.

A metamodellezés által topológiai kényszerekkel definiált nyelvek esetében további probléma, hogy a definíció gyakran nem teljes, vagy nem elegendően precíz. Ez a pontatlanság nem csupán a metamodellezésre jellemző, UML modellek esetében is gyakran előfordul. A probléma oka, hogy a nyelvek vizuális formában megadott definíciója nem, vagy csak nehézkesen képes bizonyos összetett összefüggéseket ábrázolni (pl. egyik attribútum értékének függését a másiktól). Az egyik legelfogadottabb megoldása a problémának, hogy szöveges kényszereket csatolunk a modellelemekhez evvel pontosítva a struktúrális definíciót. A Object Constraint Language (OCL) [OCL] napjaink egyik legnépszerűbb kényszerleíró nyelve. Az OCL egy egyszerű szintaktikájú, formális nyelv, amit erdetileg UML modellezéshez dolgoztak ki. Ennek az UML objektum hierarchián alapuló nyelvnek a rugalmassága viszont lehetővé teszi, hogy kiegészítések segítségével, a hagyományos, vagy akár az n-szintű metamodellezésben is felhasználhassuk.

A precíz modellek szükségessége miatt a kényszerek ellenőrzése a modellezés szempontjából alapvető fontossággal bír. A kényszerkiértékelés optimalizálása leghatékonyabban a fejlesztők által megfogalmazott kényszerkifejezések az eredetivel egyező értelmű, szemantikájú, de annál hatékonyabb átfogalmazásával végezhető el. Mivel a kényszerkifejezés az optimalizálás során megváltozik, így az optimalizált és az eredeti kényszer ekivalenciája minden esetben ellenőrizendő.

A modellalapú megközelítések esetében a modellezés gyakran csak a folyamat első lépését jelenti. A modellvezérelt architechtúra (MDA) [Soley et al., 2004] esetében először egy plattformfüggetlen model (Platform Independent Model, PIM) készül el, majd ezt modellfordítók segítségével bővítik ki a különböző plattformokra jellemző információkkal. A PIM-ből a modellfordítás eredményeképpen áll elő egy, vagy több plattformfüggő modell (Platform Specific Model, PSM). Egy másik népszerű modellalapú technika a Model-Integrated Computing (MIC) [MIC] a létrehozandó rendszer elkészítéséhez szakterület-specifikus nyelveket használ, majd ezekből készít forráskódot modelltranszformáció segítségével. A modellek feldolgozása, átalakítása, transzformálása mind a két bemutatott technológiában nagy hangsúlyt kap, de igaz ez más, modellalapú megközelítésekre is.

Napjainkban a modelleket gyakran címkézett, irányított gráf formájában írják le. Ez a leírás lehetővé teszi a gráfújraírási szabályokon alapuló gráftranszformáció használatát. Az újraírási szabályok bal oldalból (left-hand side, LHS) és jobb oldalból (right-hand side, RHS) tevődnek össze. A szabály alkalmazása során a bal oldalon definiált mintát keressük meg a transzformálandó ún. hosztmodellben, majd lecseréljük a jobb oldali mintára.

Bár a gráfújraírás hatékony módszer kis (néhány tíz elemből álló) modellek és minták esetében, de az újraírás során alkalmazott mintaillesztés algoritmikusan nehéz feladat, így a modellek és az újraírási szabályok méretének növekedésével a módszert követő transzformációk nagyon lassúvá vállhatnak. Mivel a feladat komplexitása csak speciális esetekben csökkenthető, ezért más megoldásra, a számítási kapacitás megnövelésére van szükség. Ennek a gyorsításnak az egyik módja az egymástól független részfeladatok párhuzamos

elvégzése. Általános esetben a különböző újraírási szabályok közt fennálló függőség erősen korlátozhatja a párhuzamos végrehajtás lehetőségeit, ugyanakkor elmondható, hogy a bizonyos kitételeknek eleget tevő transzformációk esetében a párhuzamosítás hatékony eszköz a modelltranszformáció felgyorsítására.

A párhuzamosítás történhet egy újraírási szabály alkalmazásán belül a mintaillesztési folyamat párhuzamosításával, vagy a transzformáció lépéseinek szintjén egymástól független lépéseket keresve.

Kutatásom célja a bemutatott problémáknak megfelelően egy olyan metamodellezési és modelltranszformációs megközelítés kidolgozása és implementálása volt, ami a korábbiaknál rugalmasabban, hatékonyabban támogatja a modellalapú szoftverfejlesztést. Mindezek alapján a következő célokat tűztem ki:

1 *A vizuális nyelvek n-szintű metamodellezésének megvalósítására alkalmas attribútumszerkezet és attribútumtranszformáció elméleti kidolgozása és formalizálása.*

2 *A metamodellezési paradigma kiegészítése a vizuális nyelvek megjelenítési információinak kezelésével.*

3 *A vizuális nyelvek modellezését támogató n-szintű metamodellezési keretrendszer elméleti kidolgozása. Az elméleti megfontolások gyakorlatban történő validálása.*

4 *Hatékony metamodell-alapú kényszerkiértékelés kidolgozása. A kényszerkifejezések automatikus optimalizálása. Az optimalizálás helyességének validálásához szükséges matematikai formalizmus kidolgozása és a validálás elvégzése. Az optimalizáció hatékonyságának bizonyítása mérések segítségével.*

5 *Modelltranszformációk párhuzamosíthatóságának vizsgálata gráftranszformációk esetén. A párhuzamosságot korlátozó és segítő tulajdonságok elemzése. Párhuzamos transzformációs algoritmusok kidolgozása és megvalósítása. A párhuzamosításból eredő hatékonyságnövekedés bizonyítása.*

2. Módszertani összefoglalás

A kitűzött célok meghatározták kutatásom irányát. Kutatásom első szakaszában a már létező elméleti és gyakorlati megoldásokat ismertem meg. Elméleti kutatásom a céloknak megfelelően alapvetően négy témakör köré épült: (i) n-szintű modellezés, (ii) a vizuális nyelvek megjelenítési információinak leírása, (iii) kényszerkiértékelés és optimalizáció, (iv) párhuzamos modelltranszformációk. Az egyes témakörökhöz kapcsolódó háttér három elméleti és egy gyakorlati részre bontható:

A modellezés terén végzett kutatásaim bevezetéseképpen a gráfalapú modellábrázoláshoz szükséges gráfelméleti megfontolásokat és algoritmusokat tanulmányoztam. A modellezés, a modellek reprezentációja során a MOF korábban már említett specifikációjat elemeztem [MOF]. Kutatásomban felhasználtam az n-szintű metamodellezés topológiai értelemben vett megvalósíthatóságának bizonyítását [Levendovszky 20505]. Ezeket az eredményeket egészíti ki az általam kidolgozott attribútumszerkezet és a modellező rétegek közti attribútum-transzformáció. A transzformáció formalizmusában és tulajdonságainak validálásához az **absztrakt állapotgépek** (Abstract State Machines, ASMs) [ASM] matematikai formalizmusát használtam fel.

A metamodellhez kapcsolódó megjelenítés definiálása során egy gráftranszformációt használtam, ami összeköti a metamodell által definiált topológiai, adat jellegű és a vizuális modell grafikai, megjelenítési jellegű információit. A felhasznált transzformáció tulajdonságainak vizsgálatához készített terminálási kritérium a **kategóriaelméleten** [Pierce, 1991] alapul, így a megoldás módszere is ezt a formalizmust használja fel.

A metamodell-alapú kényszerkiértékeléssel kapcsolatos kutatásomban első lépésben a OCL nyelv már létező, halmazelméleti formalizmusát tanulmányoztam át [OCLFormalism]. A formalizmus kiterjesztésében a korábban kidolgozott n-szintű attribútumszerkezetet használtam fel. Az optimalizált kényszerkezelés kidolgozása során építettem az aspektus-orientált kényszerkezelés néhány alapötletére [Lengyel, 2006], az ötletek átdolgozott, pontosított változatát az első optimalizáló algoritmusban használtam fel. A második optimalizációs algoritmus kidolgozása során az általános fordítókkal kapcsolatos optimalizációs lehetőségeket [Atho et al., 1998] tanulmányoztam át. A kidolgozott algoritmusok helyességének bizonyításakor felhasználtam **automataelméleti** konstrukciókat, valamint a korábban már említett absztrakt állapotgépek formalizálási technikát.

A párhuzamos transzformációk elméleti alapját a gráftranszformációk matematikai háttere [Rozenberg, 1997], valamint a transzformációk DPO (Double Pushout) megközelítése [Ehrig et al., 2005] jelentették. Az általam tanulmányozott gráftranszformációk szigorúan sorrendezettek és a DPO megközelítést követik. A szigorú sorrendezés ez esetben azt jelenti. hogy a transzformáció nem csak az újraírási lépéseket tartalmazza, de azok sorrendjét is megadja. A vizsgált transzformációk metamodell-alapú újraírási szabályokat használnak, ahol mind a bal (LHS), mind a jobb oldal (RHS) metamodell-elemekből épül fel. Az újraírás során így nem az LHS-ben definiált mintával izomorf részgráfot keresünk a gazdagráfban, hanem az LHS-ben definiált minta egy példányosításával izomorfat. A módszer egyrészről növeli az újraírási szabályok kifejezőerejét, másrészről viszont a hagyományos újraíráshoz képest új problémákat is felvet.

A párhuzamosan futtatható szabályok kiválasztásánál felhasználtam továbbá a DPO Párhuzamossági Tétel [Ehrig et al., 2005] kiterjesztését [Levendovszky 20505] is. Az újraírási szabály-szintű párhuzamosításnál építettem egy dinamikus teljesítményelosztási eredményre is [Kumar et al., 1994].

Kutatásaim során az elméleti részeredményeket a gyakorlatban is megvalósítottam, majd az így nyert tapasztalatokat visszacsatolva fejlesztettem tovább az elméleti hátteret. Az elméleti eredmények gyakorlati validálását a BME Automatizálási és Alkalmazott Informatikai Tanszékén fejlesztett Visual Modeling and Transformation System (VMTS) [VMTS] nevű metamodellező és modelltranszformációs keretrendszer segítségével végeztem el, amelynek fejlesztésében én is aktívan részt vettem. A rendszer megfelelő alapot biztosított az elméleti eredmények gyakorlatba történő átültetéséhez, ellenőrzéséhez és számos esetben a továbbfejlesztéséhez is. A VMTS rendszer nem csak az általam kidolgozott elméleti eredményekre alapul, tartalmaz számos korábbi kutatási eredményt is mint pl. a gráfújraírás matematikai háttere [Rozenberg, 1997], a validált, metamodell alapú modelltranszformációk [Lengyel, 2006], vagy az n-szintű metamodellezés topológiai jellegű problémáit megoldó eredmények [Levendovszky 20505]. A VMTS rendszerben a kutatásomhoz kapcsolódó modulok az elméleti kutatásom gyakorlati validációjának tekinthetők, a VMTS a bizonyítéka annak, hogy a kidolgozott megoldások valóban megvalósíthatóak, működőképesek.

3. Új tudományos eredmények

A kitűzött célok elérését lehetővé tevő eredményeimet négy tézisbe soroltam.

Az első tézisben az n-szintű metamodellezésben felhasználható attribútumszerkezetet és az attribútumok példányosítását elvégző transzformációt ismertetem. A tézis tartalmaz továbbá egy vizuális eszközökön alapuló módszert, ami alkalmas a grafikus modellező nyelvek vizualizációjának leírására.

A második tézis a vizuális nyelvek kényszereinek kiértékelésével kapcsolatos eredményeket tartalmazza. A tézis bemutatja, hogy az OCL nyelv hogyan terjeszthető ki az n-szintű attribútumszerkezet támogatására. A tézis ismerteti az általam kidolgozott, a kényszerellenőrzést optimalizáló algoritmusokat és ismerteti az algoritmusok együttműködésének feltételeit. A tézis tartalmazza továbbá az OCL nyelv egy újfajta matematikai formalizmusát és annak felhasználását az optimalizáló algoritmusok helyességének bizonyításában.

A harmadik tézis a párhuzamos gráftranszformációkkal kapcsolatos kutatás eredményeit ismerteti. A tézis bemutatja azt az elméleti keretrendszert, ami lehetővé teszi a párhuzamos algoritmusok használatát, a transzformáció-szintű és az újraírási szabályszintű párhuzamosítási algoritmusokat, valamint ismerteti az együttműködési lehetőségeket a két párhuzamosítási szint között.

A negyedik tézis az első három tézis elméleti eredményeit a gyakorlatban történő megvalósítás segítségével validálja. Ez a tézis tartalmazza az első tézis eredményeire alapuló metamodellező keretrendszer ismertetését, a kényszerkiértékelést végző fordító leírását, valamint a párhuzamos modelltranszformáció során felhasznált elméleti keretrendszer alapján készült alkalmazás bemutatását is.

I. Tézis
Vizuális nyelvek definíciója

Kidolgoztam egy, az n-szintű metamodellezésben használható attribútumstruktúrát, amely struktúrát követő attribútumok tetszőleges szint mélységben példányosíthatóak. Formalizust adtam a struktúrára és a transzformációra, aminek a segítségével megvizsgáltam a transzformáció néhány alapvető tulajdonságát.

Kidolgoztam továbbá egy módszert, ami a metamodellezési alapelvekhez illeszkedve, egy speciális vizuális modell segítségével lehetővé teszi a vizuális nyelvek megjelenítésének definiálását. Létrehoztam egy modelltranszformációt, amivel a kialakított, a megjelenítést megadó modell a topológiát leíró metamodellhez kapcsolható, így a metamodell adat jellegű információi rugalmasan kiegészíthetőek a megjelenítési jellegű információkkal. A modeltranszformáció tulajdonságait megvizsgálva beláttam, hogy a transzformáció minden esetben terminál.

A tézishez kapcsolódó publikációk:

[4][6][7][8][9][10][15][16][17][18][30][32][35].

A kutatás során kidolgoztam egy attribútumszerkezetet, ami képes leírni egy tetszőleges szakterületben megjelenő attribútumok szerkezetét. A szerkezet formális leírása biztosítja a precizítást, általános szerkezete pedig garantálja a nagyfokú kifejezőképességet.

3.1. Definíció. *Az Egyszerűsített N-szintű Attribútumalgebra egy* $\mathfrak{A}$ *állapotának szuperuniverzuma (*$|\mathfrak{A}|$*) több aluniverzum kompozíciójából tevődik össze, amelyek*

- U_{Bool} *a logikai értékek univerzuma {true/false}*

- U_{Number} *a racionális számok univerzuma* $\{\mathbb{Q}\}$

- U_{String} *véges hosszú karakterláncok univerzuma*

- U_{ID} *a lehetséges egyedi azonosítók univerzuma.*

3.2. Definíció. *Az Egyszerűsített N-szintű attribútumalgebra a következő karakterisztikus függvényeket tartalmazza: Meta/1, IsAttribute/1, Name/1, Root/1, Attribute/3, ahol a perjelet követően a paraméterek száma szerepel.*

I.1. Altézis

Létrehoztam a fenti definícióknak eleget tevő formális attribútum algebrát. Beláttam, hogy az algebra alkalmas az n-szintű attribútumszerkezet leírására. Matematikai formulákkal megadtam az érvényes modellek definícióját.

Megmutattam, hogy formális szintaxissal megadható egy transzformációs algoritmus, ami az attribútum algebrával megadott érvényes modellek esetében a példányosítás során elkészíti a példányszint attribútumszerkezetét.

Beláttam, hogy a kidolgozott algebra alkalmas az UML 2.0 szabványban szereplő osztálydiagram, objektumdiagram, ill. állapotdiagram attribútumszerkezetének leírására.

3.3. Definíció. *Egy modellezési hierarchiában alkalmazott példányosítást mély példányosításnak (deep instantiation) nevezzük, ha az n. szinten modellezett információ elérhető az n+x. (x>1) szinten [Atkinson and Kuhne, 2001].*

3.4. Definíció. *Az Továbbfejlesztett N-szintű attribútumalgebra a következő karakterisztikus függvényeket tartalmazza: Meta/1, IsAttribute/1, Name/1, Root/1, Attribute/3, ReInstantiate/1, ReInstantiationName/1, ahol a perjelet követően a paraméterek száma szerepel.*

I.2. Altézis

Megmutattam, hogy létrehozható egy attribútum algebra, amelynek szuperuniverzuma megegyezik az Egyszerűsített N-szintű attribútumalgebra szuperuniverzumával, karakterisztikus függvényei megfelelnek a fenti definíciónak és amely algebra támogatja a mély példányosított attribútumok kezelését is.

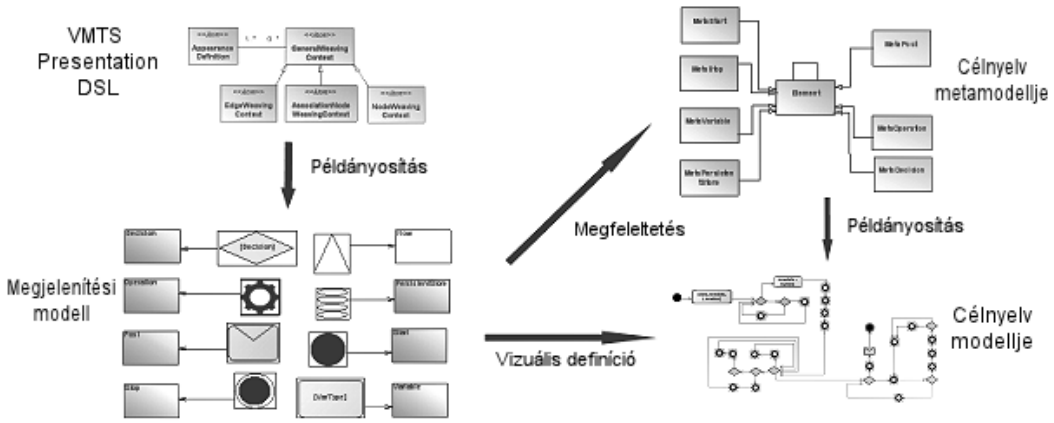
Igazoltam, hogy kidolgozható formális szintaxis segítségével egy transzformációs algoritmus, ami elvégzi a példányosítást.

Beláttam, hogy a példányosítási transzformáció minden véges számú modellelemet tartalmazó, és az attribútumokat véges mélységben egymásba ágyazó modell esetében terminál. A transzformáció komplexitását pontos formula segítségével is megadtam.

Igazoltam, hogy egy tetszőleges érvényes modellből kiindulva a transzformációs szabálynak és a modell érvényességét nem sértő módosítások kombinációjának sorozatát tetszőleges számban elvégezhetjük és az így kapott modell minden esetben érvényes lesz.

Beláttam, hogy a transzformáció alkalmas az UML 2.0 szabványban leírt osztálydiagramból a példányként kapott objektumdiagram attribútumainak generálására.

A rugalmas attribútumstruktúra segítségével a vizuális nyelvek topológiai értelemben véve definiálhatóak, de a sokszor rendkívüli fontossággal bíró megjelenítés megadását a



1. ábra. A megjelenítési és topológiai definíció összekapcsolása

metamodellezés nem írja le. Metamodellezés segítségével megadtam egy speciális vizuális nyelvet VMTS Presentation DSL (VPD) néven, ami képes más vizuális nyelvek megjelenítésének leírására. Kidolgoztam egy elméleti eszköztárat a megjelenítés definiálására, ami egyszerű elemekből építi fel a modellezni kívánt vizuális nyelv többnyire összetett megjelenítését.

I.3. Altézis

Igazoltam, hogy létrehozható olyan modell, amivel a vizuális nyelvek megjelenítése modellezhető. Beláttam, hogy a megjelenítéset megadó modell összekapcsolható az adott modellezési területet leíró, strukturális, topológiai jellegű metamodellel (1. ábra).

Beláttam, hogy a modellezési tér két eltérő aspektusának (a vizualizációnak és a struktúrális definíciónak) az összekapcsolása automatizálható modelltranszformáció segítségével. Megmutattam, hogy ez a modelltranszformáció leírható gráfújraírási szabályok sorozataként és kidolgoztam ezeket a szabályokat.

Megmutattam, hogy a VPD nyelv alkalmas többek közt az UML 2.0 szabványnak megfelelő tevékenységdiagram, a folyamatábra és a Nessie-Schneidermann diagram leírására.

A vizuális és struktúrális definíció összekapcsolására használt modelltranszformáció rugalmassá teszi a megoldást az esetleges változásokra (pl. ha a megjelenítés leírásához új elemeket szeretnénk felvenni). Más oldalról viszont ennek a rugalmasságnak az az ára, hogy a transzformáció helyességét és tulajdonságait minden változtatás esetén meg kell vizsgálni. A transzformáció egyik legfontosabb tulajdonsága a terminálás kérdése. Ennek vizsgálatához a [Plump,1998]-ben leírt terminálási tételt használtam fel.

I.4. Altézis

A transzformációk E-alapú kompozíciójára kimondott tételt [Ehrig et al., 2006] felhasználva bebizonyítottam, hogy a VPD nyelv modelljeinek feldolgozásakor használt transzformáció minden esetben terminál a bemeneti gráftól függetlenül (véges bemeneti gráfok esetében).

II. Tézis
Optimalizált, metamodell-alapú kényszerellenőrzés

Az eredetileg az UML nyelvhez létrehozott OCL nyelvet kiegészítettem az n-szintű attribútumok támogatásához szükséges elemekkel. Megmutattam, hogy az így kapott nyelv továbbra is teljesíti a formális kényszerkiértékelő nyelvekkel szemben támasztott követelményeket. Kidolgoztam két optimalizációs algoritmust, amelyek a kényszerkifejezések áthelyezésével ill. gyorsítótár alkalmazásával teszik optimálisabbá a modellek validációját. Kidolgoztam az OCL nyelv egy új típusú formalizmusát, valamint az új formalizmust felhasználva megvizsgáltam az optimalizációs algoritmusok helyességét.
A tézishez kapcsolódó publikációk:
[2][5][19][24][29][33][34][37][41][42].

Az OCL nyelv halmazelméleten alapuló formalizmusa fontos a modellek precizításának bizonyításában. A precizítás megtartása érdekében hasonlóan fontos az OCL nyelvet feldolgozó algoritmusok formális leírása is. Az eredeti formalizmus csak a nyelv statikus viselkedését, a nyelvi elemeket írja le és a leírás több helyen hiányos [OCLFormalism]. A meglévő formalizmus nem tartalmazza továbbá a végrehajtás, a kényszerkiértékelés folyamatánk definícióját. A dinamikus, az eredeti kényszert átformáló algoritmusok formalizálásához ezért egy új módszerre van szükség. Kidolgoztam az OCL nyelv egy új formalizmusát OCLASM néven. Az OCLASM az absztrakt állapotgépek programozási nyelvek esetében használatos technikáján [Stark et al., 2001] alapul.

Az OCLASM mintegy interpreterként dolgozza fel a kényszereket, az automata állapotai a végrehajtó környezet egy pillanatban vett állapotával egyeznek meg. A formalizmus monitorozott függvényeket használ a modellelemek kinyerésére (mivel az OCL definíció szerint nem változtathatja meg a modellt). Az OCLASM a kényszerkifejezéseket megosztott függények segítségével írja le. A megosztott függvényekre azért van szükség, mert bár a kényszere definiálását egy külső rendszer (vagy felhasználó) végzi, az algoritmusok a kényszereket átírhatják. Az OCLASM a kényszereket szintaxis fa formájában tárolja, a feldolgozás során ezt a fát alakítjuk át.

3.5. Definíció. *Az OCLASM a következő karakterisztikus függvényeket tartalmazza, ahol zárójelben a paraméterek száma található:*

Monitorozott függvények: *IsModelItem/1, AttrValue/1, Meta/1, To/2, Mul/2*

Megosztott függvények: *GetPhrase/1, Child/2, Parent/1*

Dinamikus függvények: *CurrentPos/0, Type/1, Value/1, Name/1, Local/1*

3.6. Definíció. *Az OCLASM egy $\mathfrak{A}$ állapotának szuperuniverzuma ($|\mathfrak{A}|$) több aluniverzum kompozíciójából tevődik össze, amelyek*

- U_{Phrase} *atomi OCL kifejezések univerzuma*

- $U_{Boolean}$ *logikai kifejezések (igaz/hamis) és az* undef *konstans univerzuma*

- $U_{Numbers}$ *a számokból képzett véges hosszú sorozatok univerzuma*

- $U_{Strings}$ *a véges hosszú karakterláncokból képzett véges hosszú sorozatok univerzuma*

- U_{ID} *a lehetséges egyedi azonosítók univerzuma .*

II.1. Altézis

Megmutattam, hogy az OCLASM segítségével az OCL nyelv formalizálható.

A formalizmus segítségével megmutattam, hogy az OCL nyelv kiegészíthető az I. tézisben leírt N-szintű attribútumalgebra attribútumainak támogatásával. A nyelvi kiegészítést formális módszerekkel leírtam.

A hatékony kényszerellenőrzés kritikus lehet a precíz modellek kialakításában. A kényszerkiértékelés közben elvégzett modellekérdezések száma döntően befolyásolja a validálás hatékonyságát, ezért optimalizációs algoritmusaim a modell lekérdezések számát csökkentik. Az optimalizáció során feltételeztem, hogy a kényszerkifejezések invariáns típusúak.

Kidolgoztam a *RelocateConstraint* algoritmust, ami a kényszerek áthelyezésével csökkenti a validálás által igényelt időt. Az algoritmusban felhasználtam korábbi kutatások eredményeit [Lengyel, 2006], de továbbfejlesztettem, kiegészítettem azokat a modellben megjelenő multiplicitások kezelésével. A multiplicitások figyelembevétele a kényszeráthelyezés helyességének alapvető feltétele a modellellenőrzés esetében.

II.2. Altézis

Megmutattam, hogy a kényszeráthelyezés algoritmusa elvégezhető offline módon, azaz a hosztmodell kiválasztása nélkül, a metamodell és a kényszerkifejezések ismeretében, így a kényszeráthelyezés ideje nem növeli a modellek validálásának idejét.

Bebizonyítottam, hogy az általános kényszeráthelyezés (két tetszőleges modellelem közt) minden esetben leírható szomszédos modellelemeken végzett áthelyezési lépések sorozataként. Megmutattam, hogy a kényszeráthelyezés általános helyességének elemzése az eredeti kényszert tartalmazó modellelemből kiinduló szélességi, vagy mélységi bejárás alapján elvégezhető.

Beláttam, hogy egy modellelemből egy kényszert nem helyezhetünk át olyan szomszédos modellelembe, ahol a céloldali minimum multiplicitás nulla. Igazoltam, hogy az áthelyezés minden más esetben lehetséges és konstrukciókat adtam az egyes esetek kivitelezésére.

Megmutattam, hogy a kényszeroptimalizáló algoritmus formalizálható az OCLASM formalizmus segítségével. A kényszeráthelyezés algoritmusának helyességét formális módon igazoltam.

A kényszeráthelyezés hatékony módszer a lekérdezések számának csökkentésére, de az áthelyezés nem mindig optimális és nem is mindig megoldható, ezért egy további algoritmust dolgoztam ki. A ReferenceCaching algoritmus egy általánosabb optimalizációt valósít meg a modellelemek lekérdezése során használt gyorsítótár segítségével. Annak érdekében, hogy feleslegesen ne foglaljunk memóriát a gyorsítótárral, az algoritmus először összegyűjti a kényszerkifejezésből azokat a hivatkozásokat, amik többször hajtódnak végre. A hivatkozások összegyűjtésének alapja egy offline referenciaszámlási algoritmus (GetCommonReferences). A kényszerkiértékelés során a CachingManagement algoritmus gondoskodik az gyorsítótár kezeléséről. A GetCommonReferences és a CachingManagement algoritmusok a ReferenceCaching algoritmus részei.

3.7. Definíció. *Egy fordítóautomata esetében két, ugyanarra a mezőre történő referenciát common subexpression-nek nevezünk [Atho et al., 1998], ha a két hivatkozás között a mező értéke nem változik.*

II.3. Altézis

Beláttam, hogy a kényszerben használt kifejezések minden esetben teljesítik a "common subexpression" kitételt , azaz az első modell lekérdezésének eredménye minden esetben megegyezik a későbbi lekérdezések eredményével.

Kidolgoztam a GetCommonReferences algoritmust és megmutattam, hogy a referenciaszámlálás offline módon elvégezhető.

Igazoltam, hogy a ReferenceCaching algoritmus alkalmazásával a modellek lekérdezésének száma nem növekedhet. Beláttam továbbá, hogy az algoritmus nem tárol el feleslegesen modellelemeket a memóriában.

Az OCLASM formalizmust felhasználva megmutattam, hogy a ReferenceCaching algoritmust minden esetben helyes eredményt ad.

III. Tézis
Párhuzamos gráftranszformáció

Egy gráfújraíráson alapuló modelltranszformáció általában több újraírási szabályból áll. Az újraírási szabályok komplexitása általános esetben $O(n^k)$, ahol n a hosztmodell elemeinek a száma, míg k a keresett minta nagysága. A teljes transzformáció komplexitása így $O(\sum_{i=1}^{r} n_i^k)$, ahol r a transzformációs lépések számát jelöli. Ez a komplexitás az egyes transzformációs lépések párhuzamos végrehajtásával redukálható. A párhuzamosítási lehetőségeket ugyanakkor korlátozzák a lépések közti fennálló függőségek, amelyeket el kell kerülni, vagy fel kell oldani. Ebben nagy szerepet kaphatnak olyan konfliktuscsökkentő heurisztikák, amelyek az egyes transzformációk jellemzői mentén optimális stratégiát valósítanak meg.

Lehetőség van párhuzamosításra az egyes újraírási lépéseken belül is. Itt az eredeti komplexitás csökkenthető a mintaillesztést különböző kezdőpontokról kezdve. A párhuzamosság felhasználását ebben az esetben az korlátozza, hogy legtöbbször nem az összes lehetséges mintaillesztést keressük, hanem csak az elsőt, így a komplexitás elméleti csökkenése nem mindig érzékelhető.

Kutatásomat a DPO megközelítést követő, metamodell-alapú újraírási szabályokat használó, szigorúan sorrendezett gráftranszformációkon végeztem el. Megmutattam, hogy ezen gráftranszformációk esetén gyorsítás érhető el, amennyiben az egymástól független feladatokat párhuzamosan hajtjuk végre. Beláttam, hogy a párhuzamosítás hatékonyságát befolyásolja a transzformáció jellege. Kidolgoztam két transzformációszintű és két újraírási szabályszintű párhuzamos mintaillesztési algoritmust, beláttam, hogy az algoritmusok eredménye mindig a szekvenciális végrehajtással egyező eredményt ad. Kidolgoztam továbbá egy elméleti keretrendszert, amely képes a két eltérő szintű algoritmus összehangolt végrehajtására.

A tézishez kapcsolódó publikációk:

[36][38].

3.8. Definíció. *Két metamodell-alapú újraírási szabály p_1 és p_2 közt fenáll a metakonfliktus reláció ($\varphi_{MetaConflict}$), ha készíthető olyan hosztmodell, ahol létezik a két szabálynak nem párhuzamosan független példányosítása.*

3.9. Definíció. *Egy szigorúan sorrendezett transzformáció feldolgozása esetén függetlenségi blokknak nevezzük a szabályok egy halmazát, ha halmaz bármely két elempárja végrehajtható párhuzamosan a hosztmodelltől függetlenül.*

Az eredetileg szigorúan sorrendezett szabálysorozatból álló transzformációkat a feldolgozáskor függetlenségi blokkok sorozatára bontottam. Az eredeti transzformáció átalakítása során egy-egy blokk akkor kerül lezárásra, ha a következő szabály már megsértené a függetlenséget.

III.1. Altézis

Megmutattam, hogy a $\varphi_{MetaConflict}$ megléte ellenőrizhető offline módon, azaz a hosztmodell kijelölése nélkül.

Megmutattam, hogy minden transzformáció szétbontható függetlenségi blokkok sorozatára. Beláttam, hogy a metamodell-alapú újraírási szabályok esetében a függetlenségi blokkok a $\varphi_{MetaConflict}$ reláció segítségével offline módon kialakíthatóak. Algoritmust adtam a blokkok kialakítására.

Kidolgoztam egy algoritmust, ami lehetővé teszi a blokkok átlapolt végrehajtását. Bebizonyítottam, hogy a párhuzamos végrehajtás helyessége nem sérül az algoritmus használata esetén még akkor sem, ha a szabályok közt függőség van.

Megmutattam, hogy az átlapolódásból származó szabályütközések esélye csökkenthető heurisztikák segítségével. Heurisztikákat adtam, amelyek az egyszerű transzformációs tulajdonságok figyelembevételével optimalizálják a szabályütközések számát.

A transzformációk párhuzamosítására nem csak az újraírási szabály feletti szinten lehetséges. Párhuzamosítható ugyanis az egyes újraírási szabályok mintakeresési algoritmusa is.

Definiáltam az $\mathbb{I}$, $\mathbb{P}$ és $\mathbb{T}$ relációkat, amelyek egy modellelem leszármazott típusait, őstípusait, ill. a modellelem metatípusát adják meg.

III.2. Altézis

Megmutattam, hogy a hosztmodell particionálható a metamodellben definiált típusok mentén és ez esetben a metamodell-alapú újraírási szabály

$$\prod_{i=1}^{k} \big(S(rn_i) - \sum_{j=1}^{i-1} N(rn_i, rn_j)\big), \tag{3.1}$$

lépést igényel, ahol k a modellelemek száma a kiválasztott szabályban, rn a szabály elemeit jelöli, míg az N egy függvény, amelynek definíciója:

$$N(l,m) = \begin{cases} 1, & if\ \mathbb{T}(m) \in \mathbb{I}(\mathbb{T}(l)) \\ 0, & otherwise \end{cases} \tag{3.2}$$

Megmutattam, hogy az újraírási szabályok mintakeresési algoritmusa párhuzamosítható az LHS definícióból tetszőlegesen választva egy modellelemet, amelybe csak olyan él vezet, amely a céloldalon (a választott modellelemnél) csak egyszeres multiplicitást enged. Igazoltam, hogy ebben az esetben a párhuzamosítás során választott modellelemhez párosított különböző hosztmodellbeli modellelemek (pivot pontok) független találatokat adnak. Beláttam, hogy a mintakeresés ebben az esetben

$$\prod_{i=2}^{k} \big(S(rn_i) - \sum_{j=2}^{i-1} N(rn_i, rn_j)\big) * S(rn_1)/c * h, \tag{3.3}$$

lépést igényel, ahol N az előzőekben definiált függvény, k a modellelemek száma a kiválasztott szabályban, rn a szabály elemeit jelöli, amelyek közül az algoritmus rn_1-t választotta ki, c a kliensek száma, h a relatív hálózati költsége felső becslése egy, kezdőponttal megadott résztalálat esetén.

Megadtam egy másik párhuzamos mintakeresési algoritmust, ami a pivotpontok kiértékelését véletlenszerű sorrendben végzi el. Bebizonyítottam, hogy az egyes kliensek minden olyan esetben megtalálják a keresett mintát, amikor a szekvenciális algoritmus megtalálná. Becslést adtam a mintakeresés költségére.

A transzformációk végrehajtásának párhuzamosítása a korábbiak szerint két szinten is lehetséges. A transzformációszintű algoritmusok akkor hatékonyak, ha az egyes transzformációs szabályok közti függőségek száma kicsi. Az újraírási szabályszintű elosztott algoritmusok hatékonyságát az befolyásolja, hogy a mintaillesztés komplexitását mennyire csökkenti az első csomópont meghatározása. Mivel a két algoritmuscsoport más-más esetben optimális, ezért fontos megvizsgálnunk, hogy együttes alkalmazásuk lehetséges-e. Igaz továbbá, hogy egy olyan keretrendeszerben, amely mindkét algoritmuscsoportot támogatja, a dinamikus terheléselosztás révén optimálisabb működést eredményezhet.

III.3. Altézis

Beláttam, hogy a transzformációszintű és az újraírási szabályszintű párhuzamos algoritmusok együtt is alkalmazhatóak. Megmutattam, hogy az algoritmusok kompoziciója kétszintű felépítést eredményez, ahol az első szintű kliensek a transzformáció szintű feladatokat oldják meg, míg a második szintű kliensek az újraírási szabályok párhuzamos végrehajtásáért felelősek.

Megmutattam, hogy a második szintű kliensek (amelyek az újraírási szabályszintű párhuzamosításban vesznek részt) és az első szintű kliensek dinamikus összerendelése megvalósítható.

Igazoltam, hogy a párhuzamos algoritmusokat futtató keretrendszer mind elméletben, mind a gyakorlatban megvalósítható. A párhuzamos futtatás hatékonyságát mérésekkel igazoltam. A mérések segítségével megmutattam, hogy a transzformáció szintű algoritmusok abban az esetben hatékonyak, ha az egyes szabályok végrehajtása nagyságrendileg azonos időt igényel. Beláttam, hogy a másodszintű kliensek segítségével a feltételnek eleget nem tevő transzformációk is végrehajthatóak hatékonyan.

IV. Tézis
Elméleti eredmények gyakorlati validációja

A Visual Modeling and Transformation System keretrendszeren keresztül megmutattam a következőket:

- *Megmutattam, hogy az n-szintű attribútumszerkezet és transzformáció a gyakorlatban is megvalósítható. Referenciaimplementációként elkészítettem az algebra XML-alapú és objektumorientált megvalósítását. Az attribútumok alkalmasságát számos műszaki problémában, pl. a funkciómodellben, ill. mobiltelefonok számára készített erőforrásszerkesztőben is igazoltam.*

- *Igazoltam, hogy az első tézisben bemutatott vizuális metamodell képes leírni a folyamatábra, a Nessie-Schniedermann diagram és az UML szabvány szerinti tevékenység diagram megjelenítését. Megmutattam, hogy a leírás eredményeképpen előálló definíció a gyakorlatban közvetlenül felhasználható.*

- *Beláttam, hogy a modellek közti tartalmazási hierarchia kettébontható a valós és logikai tartalmazás (a tényleges, és a modellezési aspektusokban használt relációk) mentén.*

- *Megmutattam, hogy az I. tézis eredményeire alapozva létrehozható egy olyan alkalmazás, ami támogatja UML diagramok, modelltranszformációs és erőforrásszerkesztő nyelvek grafikus szerkesztését.*

- *Igazoltam, hogy a kényszerellenőrzéshez a gyakorlatban is kidolgozható egy OCL fordító, ami mind a metamodell alapú kényszereket, mind az ismertetett optimalizációs algoritmusokat felhasználja a modellek validálása során.*

- *Mérések segítségével bebizonyítottam, hogy az optimalizált kényszerellenőrzés a gyakorlatban hatékonyabb az eredeti kényszereken alapuló modellvalidálásnál.*

- *Megmutattam, hogy a memóriaalapú modellábrázolás megközelítőleg három nagyságrenddel gyorsabb modelltranszformációt tesz lehetővé, mint az adatbázis alapú.*

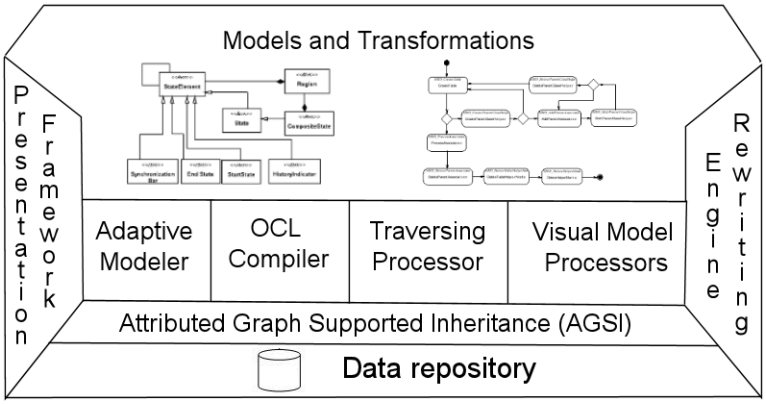
- *Igazoltam, hogy a párhuzamos gráftranszformációhoz kidolgozott elméleti keretrendszer a gyakorlatban is megvalósítható kétszintű szerver-kliens architektúrát követve.*

A tézishez kapcsolódó publikációk:
[1][3][11][12][13][14][20][21][22][23][25][26][27][28][29][31][39][40].

4. Az új tudományos eredmények alkalmazása

Az új tudományos eredmények a VMTS metamodellező és modelltranszformációs keretrendszerben, illetve a VMTS alkalmazásaiban jelennek meg. A VMTS egy n-szintű metamodellező környezet, amely lehetővé a metamodellek és modellek szerkesztését, automatikus sémaellenőrzést biztosítva számukra. A VMTS egy egyszerűsített UML osztálydiagrammal írja le alap metamodelljét („vizuális szókincsét"). A VMTS támogatja a vizuális nyelvek megjelenítésének grafikus megadását és a metamodellel történő összekapcsolását. A precíz modellek létrehozásához a felhasználók kényszereket definiálhatnak. A VMTS része a gráf transzformációs alapokon nyugvó modelltranszformációs rendszer és egy, a bejárás alapú modellfeldolgozást támogató eszköz is.

A VMTS segítségével megvalósított esettanulmányok lehetővé teszik az alkalmazhatóság bemutatását mind a tudományos világ, mind az ipar számára. Mivel mind az elméleti háttér, mind a rendszer evolúciója folyamatos, ezért az itt felsorolt alkalmazási területek listája nem tekinthető végesnek.
Az alkalmazásokkal kapcsolatos publikációk:
[3][6][13][15][20][21][22][39][40]



2. ábra. A VMTS keretrendszer architektúrája

Keretrendszer

Az eredmények gyakorlati alkalmazhatóságának igazolására kifejlesztettem a VMTS keretrendszer következő komponenseit (Fig. 2): (i) VMTS Presentation Framework, ami a modellező alkalmazás (Adaptive Modeler) alapja, felhasználóbarát módon támogatja a nyelvek létrehozását és kezelését, továbbá rugalmas, plugin-alapú architektúrát biztosít a speciális vizuális nyelvek implementására; (ii) Attribute Panel, ami az attribútumalgebra feldolgozását, az attribútumok megjelenítését és a metamodell által definiált attribútumséma automatikus ellenőrzését biztosítja; (iii) OCL Compiler, ami az OCL nyelven megfogalmazott kényszereket lefordítja, optimalizálja, majd kódot és bináris állományt generál belőlük; (iv) AGSI Compact Framework, ami a modellek memóriaalapú tárolását és szerializálását teszi lehetővé; (v) AGSI Parallel Transformation Engine, ami a párhuzamos újraírási algoritmusok számára biztosít keretrendszert. A szoftvercsomag egyéb részei Levendovszky Tihamér, Lengyel László és számos hallgató munkájának eredménye.

Alkalmazások

A vizuális nyelvek modellezését és transzformációját számos gyakorlati esetben használtuk műszaki problémák megoldására. Az attribútumstruktúra, a modellező keretrendszer és a kényszerek fordításához készített OCL compiler segítségével lehetővé vált több olyan terület egységes alapokon történő modellezése, ami korábban nehézségekbe ütközött. A teljesség igény nélkül néhány ezek közül: (i) Vizuális modellezőnyelv a transzformációk leírására (VCFL), ill. az újraírási szabályok megadására [3][12]. (ii) A mobiltelefonok erőforrásait leíró nyelvek létrehozása és szerkesztése [13][22]. (iii) az UML 2.0 szabványnak megfelelő nyelvcsaládok, pl tevékenység -, osztály -, objektum -, szekvencia- és állapotdiagram támogatása. (iv) Vizuális modellezőnyelv a megjelenítés megadására [4][18]. (v) A programozási folyamatok leírására szolgáló nyelvek támogatása (Folyamatábra, Nassi-Schneidermann diagram). (vi) Mobil hálózati protokoll modellezése tisztán grafikus elemekkel.

A memória alapú AGSI Compact Framework és a párhuzamos modelltranszformáció támogatása révén lehetőség nyílt a korábbiaknál sokkal összetettebb modellek feldolgozására és transzformációk támogatására. Amíg az adatbázis alapú megközelítés néhány tíz modellelemet és újraírási szabályt tartalmazó esetekhez volt tervezve, addig az új módszerekkel néhány ezer, esetenként néhány tízezer modellem is feldolgozhatóvá vált.

A példák illusztrálják, hogy a VMTS hatékony metamodellező rendszer, ami támogatja vizuális nyelvek definiálását, szerkesztését és transzformációalapú feldolgozását.

5. Related Publications

Folyóiratcikkek

1 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „A Strict Control Flow Specification for Model Transformation", WSEAS Transactions on Computers, February 2006, ISSN 1109-2750.

2 **G. Mezei**, L. Lengyel, T. Levendovszky, H. Charaf, „Minimizing the Traversing Steps in the Code Generated by OCL 2.0 Compilers", WSEAS Transactions on Information Science and Applications, Issue 4, Volume 3, February 2006, ISSN 1109-0832, pp. 818-824.

3 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, Model Transformation with a Visual Control Flow Language, International Journal of Computer Science (IJCS), Number 1, Volume 1, 2006, ISSN 1306-4428, pp. 45-53.

4 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, A Model Transformation for Automated Concrete Syntax Definitions of Metamodeled Visual Languages, Electronic Communications of the EASST, Volume 4, 2006, Graph and Model Transformation 2006

5 T. Levendovszky, **G. Mezei**, H. Charaf, Formalizing the Evaluation of OCL Constraints , Acta Polytechnica Hungarica, Volume 4, Issue 1, 2007, ISSN 1785-8860, pp. 89-110.

6 **G. Mezei**, Hatékony szakterület-modellezési technikák a távközlésben, Magyar Távközlés, Budapest, 2006, Accepted

Nemzetközi konferencia-kiadványban megjelent idegen nyelvű előadás

7 T. Levendovszky, L. Lengyel, **G. Mezei**, H. Charaf, „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.

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

9 **G. Mezei**, T. Levendovszky, L. Lengyel, H. Charaf, „Multilevel Metamodeling - A Case Study", MicroCAD, March 10-11, 2005, Miskolc, pp. 321-326.

10 **G. Mezei**, L. Lengyel, T. Levendovszky, H. Charaf, „A Metamodel-Based Technique for Attribute Transformation in Visual Model Processors", 5th Internal Conference of PhD Students, Engineering Sciences II, Miskolc, Hungary, 14-20 August 2005, pp. 133-138.

11 L. Lengyel, T. Levendovszky, **G. Mezei**, B. Forstner, H. Charaf, „Metamodel-Based Model Transformation with Aspect-Oriented Constraints", International Workshop on Graph and Model Transformation (GraMoT), Electronic Notes in Theoretical Computer Science, Volume 152, Tallinn, Estonia, September 28, 2005, pp. 111-123.

12 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „Control Flow Support in Metamodel-Based Model Transformation Frameworks", EUROCON 2005 International Conference on „Computer as a tool", Proceedings of the IEEE, Belgrade, Serbia & Montenegro, November 21-24, 2005, pp. 595-598.

13 L. Lengyel, T. Levendovszky, **G. Mezei**, B. Forstner, H. Charaf, „Towards a Model-Based Unification of Mobile Platforms", 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06), Dubai, Sharjah, UEA, March 8-11, 2006.

14 **G. Mezei**, T. Levendovszky, H. Charaf, „Traversing Processor for Metamodel Environments", 5th Internal Conference of PhD Students, August, 2005, Miskolc, Hungary, pp. 127-133.

15 **G. Mezei**, T. Levendovszky, H. Charaf, „A Presentation Framework for Metamodeling Environments", 4th Workshop in Software Model Engineering, 2005 October, Montego Bay, Jamaica.

16 **G. Mezei**, T. Levendovszky, H. Charaf, „Presentation Framework - an Environment for Editing Metamodels", 6th International Symposium of Hungarian Researchers, November, 2005, Budapest, Hungary, pp. 551-562.

17 **G. Mezei**, T. Levendovszky, H. Charaf, „A Flexible Graphics Framework for Modeling Environments", International Carpathian Control Conference, Miskolc, Hungary, 2005, pp. 383-389.

18 **G. Mezei**, T. Levendovszky, H. Charaf, „Visual Presentation Solution for Domain Specific Languages„ IASTED on SE, February, 2006, Innsbruck, Austria, pp. 194-199.

19 **G. Mezei**, T. Levendovszky, H. Charaf, „Implementing an OCL 2.0 Compiler for Metamodeling Environments", 4th Slovakian - Hungarian Joint Symposium on Applied Machine Intelligence, SAMI 2006, January, 2006, Herlany, Slovakia, pp. 544-555.

20 L. Lengyel, T. Levendovszky, T. Vajk, **G. Mezei**, H. Charaf, „Practical Uses of Validated Model Transformation", IEEE EUROCON 2007, Sept, 2007, Warsaw, Poland, accepted

21 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „A Visual Control Flow Language for Model Transformation Systems", IASTED International Conference on SOFTWARE ENGINEERING, SE 2006, Innsbruck, Austria, February 14-16, 2006, pp. 194-199.

22 B. Forstner, L. Lengyel, T. Levendovszky, **G. Mezei**, I. Kelényi, H. Charaf, „Model-Based System Development for Embedded Mobile Platforms", 13th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), Potsdam, Germany, March 27 - 30, 2006.

23 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „A Visual Control Flow Language", 5th WSEAS Int.Conf. on Software Engineering, Parallel and Distributed Systems, SEPADS '06, Madrid, Spain, February 15-17, 2006, pp. 30-35.

24 **G. Mezei**, L. Lengyel, T. Levendovszky, H. Charaf, „Optimization Algorithms For OCL Compilers", 5th WSEAS Int.Conf. on Software Engineering, Parallel and Distributed Systems, SEPADS '06, Madrid, Spain, February 15-17, 2006, pp. 55-60.

25 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „Execution Properties of a Visual Control Flow Language", 4th Slovakian - Hungarian Joint Symposium on Applied Machine Intelligence', SAMI 2006, January 20 - 21, 2006, Herlany, Slovakia, pp. 368-379.

26 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „Control Flow Support in Model Transformation Frameworks: An Overview", MicroCAD, March 16-17, 2006, Miskolc, pp. 193-198.

27 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „Weaving Constraint Aspects into Metamodel-Based Model Transformation Steps", MicroCAD, March 16-17, 2006, Miskolc, pp. 199-204.

28 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „Model-Based Development with Strictly Controlled Model Transformation", The 2nd International Workshop on Model-Driven Enterprise Information Systems, MDEIS 2006, May 23-24, 2006, Paphos, Cyprus, pp. 39-48.

29 T. Levendovszky, **G. Mezei**, H. Charaf, „Customized Error Handling Suport for OCL Compilers", MicroCAD, March, 2006, Miskolc, pp. 211-217.

30 **G. Mezei**, T. Levendovszky, H. Charaf, „A Domain-Specific Language for Visualizing Modeling Languages", ISIM, April, 2006, Prerov, Czech Republic.

31 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „Constraint Validation-Driven Visual Model Transformation", Automation and Applied Computer Science Workshop (AACS), June, 2006, Budapest, Hungary, pp 29-41.

32 L. Lengyel, T. Levendovszky, **G. Mezei**, H. Charaf, „A Model Transformation Method To Support Automated Concrete Syntax Definitions", Automation and Applied Computer Science Workshop (AACS), June, 2006, Budapest, Hungary, pp 41-53.

33 **G. Mezei**, T. Levendovszky, H. Charaf, „An optimizing OCL Compiler for Metamodeling and Model Transformation Environments, IFIP Working Conference on Software Engineering Techniques, October, 2006, Warsaw, Poland, pp 61-73.

34 **G. Mezei**, T. Levendovszky, H. Charaf, „Extending an OCL Compiler for Metamodeling and Model Transformation Systems: Unifying the Twofold Functionality", 10th International Conference on Intelligent Engineering Systems 2006, June, 2006, London, United Kingdom.

35 **G. Mezei**, T. Levendovszky, H. Charaf, „Automatized Concrete Syntax Definitions For Domain Specific Languages", The 7th International Conference on Technical, June, 2006, Timisoara, Romania, pp 47-52.

36 **G. Mezei**, T. Levendovszky, H. Charaf, „A Distribution Technique For Graph Rewriting and Model Transformation Sytems", IASTED on PDCN, February 15-17, 2005, Innsbruck, Austria

37 **G. Mezei**, T. Levendovszky, H. Charaf, „A New Formalism Technique For OCL", 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics, Jan, 2007, Poprad, Slovakia, pp 117-128.

38 **G. Mezei**, T. Levendovszky, H. Charaf, „Integrating Model Transformation Systems and Asynchronous Cluster Tools", 7th International Symposium of Hungarian Researchers on Computational Intelligence, Nov, 2006, Budapest, Hungary, pp 307-318.

39 T. Mészáros, **G. Mezei**, T. Levendovszky, „Using Patterns in Domain-Specific Languages", 7th International Symposium of Hungarian Researchers on Computational Intelligence, Nov, 2006, Budapest, Hungary, pp 167-178.

Technical Report

40 L. Lengyel, T. Levendovszky, **G. Mezei**, „Metamodell-alapú modelltranszformáció", Budapesti Műszaki és Gazdaságtudományi Egyetem, Automatizálási és Alkalmazott Informatikai Tanszék, Budapest, Hungary, 2005.

41 **G. Mezei**, T. Levendovszky, H. Charaf, „Restrictions for OCL constraint optimization algorithms", Technische Universitat Dresden, Genova, Italy, 2006.

42 **G. Mezei**, T. Levendovszky, H. Charaf, „Optimizing Refinement for Metamodeling Environments with OCL Constrains", Labroatoire d'Informatique de Robotique et de Microelectronique de Montpellier, MoDELS doctoral symposium, Genova, Italy, 2006.

6. Idegen hivatkozások

[1] publikációra hivatkozik:

A. Balogh, D. Varró, „Advanced Model Transformation Language Constructs in the VIATRA2 Framework", ACM Symposium on Applied Computing, 2006, pp. 1280-1287.

[7] publikációra hivatkozik:

E. Biermann, K. Ehrig, C. Köhler, G. Kuhns, G. Taentzer, E. Weiss, „Graphical Definition of In-Place Transformations in the Eclipse Modeling Framework", Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genova, Italy, October 1-6, 2006, LNCS Volume 4199, pp. 425-439.

[13] publikációra hivatkozik:

Anne Etien , Cedric Dumoulin , Emmanuel Renaux, „Towards a Unied Notation to Represent Model Transformation", Institut National de Recherce en Informatique et an Automatique, Rapport de recherche, 2007.

7. Hivatkozott Irodalom

Hivatkozások

[UML] Object Management Group Unified Modeling Language 2.0 Specification, http://www.uml.org/#UML2.0

[DSML] Domain-Specific Modeling, http://www.dsmforum.org/

[Nordstrom, 1999] Gregory G. Nordstrom, „Metamodeling - Rapid Design and Evolution of Domain-Specific Modeling Environments„, PhD Thesis, Nashville, US, 1999

[MOF] Meta Object Facility Specification, http://www.omg.org/mof/

[Levendovszky 20505] Tihamér Levendovszky, „Applying Metamodels in Software Model Transformation Methods", PhD Thesis, Budapest, Hungary, 2006

[OCL] Jos Warmer, Anneke Kleppe, „Object Constraint Language, The: Getting Your Models Ready for MDA", Addison Wesley, 2003

[Soley et al., 2004] Richard Soley and David S. Frankel and John Parodi, „The MDA Journal: Model Driven Architecture Straight From The Masters", Meghan Kiffer Pr, 2004

[MIC] Jonathan Sprinkle, „Model-Integrated Computing", IEEE Potentials, Issue 23(1), 2004.

[ASM] Egon Böerger, Robert Stärk, „Abstract State Machines", Springer-Verlag, Germany, 2003

[Pierce, 1991] Benjamin C. Pierce, „Basic Category Theory for Computer Scientists", MIT Press, 1991.

[OCLFormalism] Object Management Group Object Constraint Language Specification, http://www.omg.org/docs/ptc/03-10-14.pdf

[Lengyel, 2006] László Lengyel, „Online Validation of Visual Model Transformations", PhD Thesis, Budapest, Hungary, 2006

[Atho et al., 1998] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, „Compilers Principles, Techniques, and Tools", Addison - Wesley,1988

[Rozenberg, 1997] Grzegorz Rozenberg, „Handbook on Graph Grammars and Computing by Graph Transformation: Foundations", World Scientific, Singapore, 1997.

[Ehrig et al., 2005] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, and Gabriele Taentzer, „Fundamentals of Algebraic Graph Transformation", Springer Verlag, 2005.

[Kumar et al., 1994] V. Kumar and A. Y. Grama and Nageshwara Rao Vempaty, ´´Scalable Load Balancing Techniques for Parallel Computers", Journal of Parallel and Distributed Computing, Volume 22, 1994, pp. 60–79

[VMTS] VMTS Homepage, http://www.vmts.aut.bme.hu

[Atkinson and Kuhne, 2001] Colin Atkinson and Thomas Kühne, „The Essence of Multilevel Metamodeling", UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools, 2001, pp. 19-33

[Plump,1998] Detlef Plump„Termination of graph rewriting is undecidable", Fundam. Inform. 33, 1998, pp. 201 - 209

[Stark et al., 2001] Robert Stärk and Joachim Schmid and Egon Börger, „Java and the Java Virtual Machine: Definition, Verification, Validation", Springer Verlag, 2001

[Ehrig et al., 2006] Hartmut Ehrig, Tihamér Levendovszky, Ulrike Prange, „A termination criteria for DPO transformations with injective matches", In Graph Transformation for Verification and Concurrency, Bonn, Germany, 2006