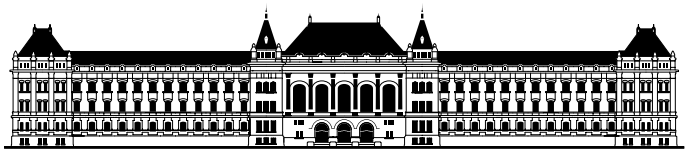

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
MŰSZAKI TUDOMÁNYÁG – INFORMATIKAI TUDOMÁNYSZAK

AUTOMATIKUS MODELLTRANSZFORMÁCIÓK INFORMATIKAI RENDSZEREK VIZSGÁLATÁHOZ

PHD ÉRTEKEZÉS TÉZISEI

VARRÓ DÁNIEL
OKL. MÉRNÖK-INFORMATIKUS

TÉMAVEZETŐ:
DR. PATARICZA ANDRÁS
EGYETEMI DOCENS
A MŰSZAKI TUDOMÁNY KANDIDÁTUSA

BUDAPEST, 2003. DECEMBER

1. A kutatás előzményei

Ma már nem felelőtlenség kijelenteni, hogy az 1990-es évek az objektum-orientált modellezési és programozási filozófia térhódítását és egyeduralmát hozták meg. Az első objektum-orientált programozási nyelvek és tervezési metodikák azonban mind egyediek voltak, magukban hordozva a tervezőműhely rendszerfejlesztési szokásait. A szoftver bázisú rendszerek komplexitásának növekedésével azonban nemcsak a rendszerek maguk, de a tervezés folyamata is elosztottá vált, és egyre inkább előtérbe kerültek a szoftverfejlesztő csoportok egymás közötti illetve a megrendelővel folytatott kommunikációjának problémái.

Az igazi áttörést a **Unified Modeling Language (UML)** [RJB99], az objektum-orientált rendszertervezés szabványos, grafikus modellezési nyelvének megalkotása hozta. Az UML sikere egyrészről szabványosságában, másrészről grafikus mivoltában rejlik. *Szabványossága* révén megszűnt az egyedi tervezési metodikákra jellemző bábeli zűrzavar, hiszen a tervező és a programozó közös nyelvet használ a kommunikációra. Az UML nyelv közérthetőségét pedig grafikus jellege adja, a szoftverrendszereket különféle nézetekből *grafikus diagramok* segítségével lépésről lépésre finomítva tervezhetjük.

A mindennapi használat során azonban számos probléma merült fel az UML-lel kapcsolatban mind ipari, mind akadémiai felhasználásokban. Egyrészt az UML nyelv egysíkú, monolit volta miatt *nehezen hangolható speciális alkalmazási területekre* (például valós idejű alkalmazások, ütemezési problémák), amely "házi szabványok", ún. UML Profile-ok létrejöttéhez vezetett. A legfontosabb problémát azonban a *precíz szemantika* hiánya jelenti: noha a grafikus nyelv *elemkészlete precízen definiált* (formálisan), az egyes *elemek jelentését informális* (angol) nyelven definiálták. Ennek következtében a nyelv egyes elemei félreérthetők: mást érthet rajta a tervező és mást a programozó.

Formális módszerek az informatikában. A precíz szemantika hiánya még hangsúlyosabban jelentkezik a nagymegbízhatóságú (pl. telekommunikációs és banki rendszerek) és biztonságkritikus alkalmazások (mint vasúti biztonságfelügyelő rendszerek, légiforgalom irányítás) esetén, ahol az előírt szolgáltatás elmulasztása anyagi ill. emberi veszteségekkel járhat. Ilyen rendszerekben a hagyományos tesztelés alapú hibakeresés eredményessége nem kielégítő, mivel minden hiba megfelelő tesztesettel való lefedése lehetetlen. *E rendszerek **verifikációja** és **validációja** során matematikai módszerekkel kell bizonyítanunk, hogy a tervezett rendszer kielégítően fog viselkedni.*

Szolgáltatásbiztos rendszerek matematikai verifikációját **formális módszerek** segítségével végezhetjük, de e módszerek használatának széleskörű elterjedését számos tényező gátolja. Használatukhoz egyrészről nagyfokú matematikai jártasság szükséges, amely ritkán várható el egy rendszertervezőtől. Másrészről nem garantált a kapott *matematikai modellek valósághűsége*, valamint a tervezés alatt álló *rendszermodell és matematikai modelljének konzisztenciája* sem.

A 1990-es évek végén a HIDE akronim alatt a BME Méréstechnika és Információs Rendszerek Tanszék Hibatűrő Csoportjának részvételével futó európai ESPRIT projekt kutatásainak célja az volt, hogy **automatikus modelltranszformációk** segítségével [BDCL$^+$01, 1] áthidalja a grafikus rendszermodellek közérthetősége és a formális, matematikai modellek precizitása közti szakadékot. Az UML bázisú rendszermodellből au-

tomatikusan generált matematikai leírásokon elvégzett formális analízis eredményét *automatikusan visszavetítjük* a kiinduló UML leírásba, elrejtve ezáltal a matematikát a rendszermérnökök elől.

Számos transzformáció készült el e rendszeren belül, többek között funkcionális helyességellenőrzés [LMM99] vagy kvantitatív analízis [BMM99] céljából. A benchmark kísérletek és alkalmazások során azonban kiderült, hogy maguk a transzformációs szkriptek jelentik a HIDE keretrendszer minőségi szűk keresztmetszetét. A fő problémák többek között a következők voltak:

- A transzformációk *tervezése* teljesen ad hoc módon történt. Mi több, a publikációkban közölt matematikai háttér és a transzformációs keretrendszer alacsonyszintű leírónyelve között hatalmas absztrakciós szakadék tátongott.
- Megfelelő *automatizáltság* nélkül a komplexebb modelltranszformációk implementálása komoly (hónapokban mért) ráfordítást igényelt.
- Formális matematikai háttér híján gyakorlatilag lehetetlen volt ellenőrizni *a transzformációk helyességét*. Enélkül pedig nem lehetséges eldönteni, hogy amennyiben a matematikai analízis hibát jelez, akkor az az eredeti rendszermodell tervezési hibájából vagy a transzformációs program hibájából ered.

Kutatásom célja ennek megfelelően egy olyan modelltranszformációs keretrendszer kidolgozása és implementálása volt, mely támogatja a magas szinten tervezett transzformációk automatikus implementációját és helyességellenőrzését. Egy komplex modelltranszformációs keretrendszerrel szemben az alábbi követelményeket támasztottam:

1. Szükséges a különféle (matematikai és mérnöki) *modellek és modellezési nyelvek matematikailag precíz* és ipari szabványokhoz illeszkedő leírása.
2. Kidolgozandó egy általános *magasszintű specifikációs módszer* a *modelltranszformációk* formális leírására.
3. Megalkotandó egy *visszavetítési mechanizmus*, mely támogatja a matematikai analízis eredményeinek UML szintű értelmezését.
4. Kidolgozandó a *modelltranszformációk formális helyességellenőrzésének* egy, lehetőség szerint automatizált módszere.
5. Végezetül szükséges e bizonyítottan helyes specifikációból kiindulva a *modelltranszformációkat implementáló programok automatikus szintézise*.

2. A kutatás módszere

A fenti követelmények szabták meg alapvetően a kutatásom irányát és az egyes megoldandó részfeladatokat.

Modellezési nyelvek struktúrájának tervezése. Első lépésként megvizsgáltam a modellezési nyelvek mérnöki specifikációjának de facto ipari szabványaként tekintett MOF metamodellezés (Meta Object Facility [OMG03]) elemkészletét és módszertanát. A MOF metamodellezés egy ún. négy metaszintű architektúrát vezet be a modellezési paradigma (metametamodell), a modellezési nyelvek (metamodellek), a modellezett rendszer (modellek) és a valós világ objektumainak elkülönítésére, ahol egy felsőbb metaszint elemeinek *példányai* mindig a közvetlenül alatta lévő metaszinten jelennek meg.

Azt tapasztaltam, hogy — noha grafikus elemkészlete és objektum-orientált szemlélete igen intuitívvá teszi használatát a mérnöki gyakorlatban — a MOF metamodellezésnek számos olyan hiányossága van, amely megakadályozza a matematikai és mérnöki modellezési nyelvek egységes kezelését.

Ezért a MOF grafikus jelölésrendszerét megtartva a metamodellezés módszertanát mélységében bővítettem ki a metaszintek nélküli és dinamikusan származtatható öröklési és példányosítási reláció bevezetésével (1. tézis), ügyelve ugyanakkor arra, hogy a kapott metamodellezési keretrendszer elemkészletét tekintve egyszerűbb legyen a MOF-nál, ugyanakkor a modellezési leíróképességében kibővítse azt.

Alapvető feladatnak tekintettem, hogy ne kizárólag egy elméleti metamodellezési keretrendszert dolgozzak ki, hanem konkrét javaslatot adjak a keretrendszer konzisztenciájának kritériumaira, továbbá olyan elemi műveletekre, melyek végrehajtásuk esetén garantálják e konzisztenciát.

Modelltranszformációk grafikus leírása. A metamodellezés köré csoportosuló kutatásokkal párhuzamosan kezdtem vizsgálni, hogy milyen formalizmusok léteznek a szakirodalomban a modellezési nyelvek közötti transzformációk leírására. Kutatásaim kezdetekor (még 1999-ben) azt tapasztaltam, hogy a kutatások e területen még igen kezdetleges stádiumban folytak.

A *gráftranszformáció* paradigmájának megismerése után rögtön kézenfekvőnek és célszerűnek látszott a paradigma adaptációja a modelltranszformáció feladatára, hiszen a gráftranszformáció – a matematikai precizitást megőrizve – a mérnöki gyakorlathoz jól illeszkedő módon grafikusan képes a legkülönfélébb transzformációk specifikálására.

Ezen adaptáció elsősorban a gráftranszformáció és a VPM metamodellezés ötvözését, a legfontosabb vezérlési szerkezetek bevezetését, valamint a modelltranszformációk visszavetítését támogató referencia metamodellek koncepciójának megalkotását jelentette. E területen elért eredményeimet foglalja össze a disszertáció 2. tézise, melyben egy olyan formalizmust ismertetek, amely a modelltranszformációkon túl a modellezési nyelvek dinamikus szemantikájának specifikációját is lehetővé teszi.

Modelltranszformációk automatizálása. Már a kutatásomat megelőzően futó HIDE projekt is demonstrálta, hogy informatikai rendszerek verifikációjának és validációjának modelltranszformációs megközelítése szempontjából kritikus a modelltranszformációk *implementációjának* automatikus és bizonyíthatóan helyes szintézise.

Ezért megvizsgáltam, hogy hogyan implementálható egy gráftranszformációs virtuális gép különféle programozási környezetekben. Megállapítottam, hogy bár számos gráftranszformációs modellező eszköz létezik akadémiai körökben, ezek mind kutatási prototípusok, és kétséges, hogy e módszerek és eszközök hogyan hasznosíthatók

közvetlenül ipari méretű modelltranszformációk esetén.

Megvizsgáltam továbbá, hogy hogyan specifikálható magának az automatikus programgenerálásnak a folyamata. Azt tapasztaltam, hogy az iparban létező kódgeneráló eszközök általában igen ad hoc módon működnek. Kivételt képeznek ez alól azok a legmodernebb eszközök, melyek a szabványos UML Akció Szemantikáját használják fel leírónyelvként.

Ezért a 3. tézisben javaslatot teszek a programgenerálás egy új módszerére, melyben a kódgenerálás folyamatát szukcesszív modelltranszformációk által specifikálom. Kidolgoztam továbbá egy Prolog alapú és egy szabványos UML Akció Szemantikai leírást használó virtuális gráftranszformációs gépet is.

Modellezési nyelvek formális verifikációja. Mivel mind a modellezési nyelvek dinamikus szemantikáját, mind pedig a nyelvek közötti modelltranszformációkat gráftranszformáció segítségével definiáljuk, ezért létfontosságú olyan automatikus módszerek összegyűjtése, mellyel formálisan is analizálhatók e specifikációk.

Egy rendszermodell helyességének vizsgálata céljából igen nagy számban használnak különféle modellellenőrző és automatikus tételbizonyító eszközöket. A tételbizonyító eszközökkel tipikusan általánosabb következtetéseket vagyunk képesek levonni egy modellről, használatukhoz azonban nagyfokú matematikai jártasság és felhasználói interakció szükséges. Ezért figyelmemet ezután a nagyfokú automatizmust biztosító modellellenőrző eszközökre fordítottam (melyek természetesen csak modellek egy speciális részhalmazát képesek vizsgálni).

Megvizsgáltam, hogy a szakirodalom milyen lehetőséget ad a gráftranszformációs rendszerek automatikus, modellellenőrző eszközök által támogatott helyességellenőrzésére. Meglepve tapasztaltam, hogy bár a gráftranszformációs rendszerek modellellenőrzésének feladatára létezett néhány elméleti megközelítés, egyetlen esetben sem került sor olyan vizsgálatokra (konkrét modellellenőrző eszközök felhasználásával), mely igazolná az egyes módszerek gyakorlati alkalmazhatóságát.

A 4. tézisben egy olyan módszert ismertetek, mely nemcsak elméleti, hanem gyakorlati szemszögből is megalapozottnak tekinthető, hiszen a módszer matematikai helyességén kívül benchmark verifikációs problémákon igazolom annak gyakorlati alkalmazhatóságát is.

Modelltranszformációk formális helyességvizsgálata. Mivel automatikus modelltranszformációk esetén maga a transzformáció specifikációja is lehet hibás, ezért szükségszerű volt megvizsgálni, hogy milyen módszerekkel lehet formálisan (és lehetőség szerint automatizáltan) igazolni egy modelltranszformáció helyes voltát.

Először kidolgoztam egy kritériumrendszert, melynek egy modelltranszformáció bizonyíthatóan meg kell, hogy feleljen. Ez alapján beszélhetünk a modelltranszformációk szintaktikus és szemantikus helyességéről. Szintaktikus helyességvizsgálat esetén eldöntendő, hogy a transzformált célmodell vajon tényleg a célnyelv egy helyes modellje-e, míg a szemantikus helyességvizsgálat esetén célunk annak eldöntése, hogy a transzformáció megőrzi-e a (transzformációra jellemző) konzisztencia tulajdonságokat.

A szintaktikus helyességvizsgálat céljából először egy, tervkészítő algoritmusokon alapuló módszert dolgoztam ki, majd a 4. tézis eredményeit felhasználva kidolgoztam egy,

a forrásnyelvi és célnyelvi modellek modellellenőrzésén alapuló megközelítést, melylyel eldönthető, hogy egy adott tulajdonság invariánsa-e a transzformációnak. Eme új eredmények összefoglalását szolgálja az 5. tézis.

3. Új tudományos eredmények

3.1. VPM: A metamodellezés matematikája

Megvizsgáltam a metamodellezés Meta Object Facility (MOF) szabványát [OMG03], és egyszerre tapasztaltam számos ellentmondást, hiányosságot és redundanciát egy olyan metamodellező nyelvben, melynek minimalitása, matematikai precizitása, és kifejezőképessége kritikus a grafikus modellezési nyelvek specifikációjának minősége szempontjából. Formális szemantika híján problematikus továbbá a MOF alkalmazhatósága a matematikai modellek precíz metaszintű leírására.

1. tézis. *Kidolgoztam egy új, a matematikai definíciók felépítésén és a metaszintek (típus-példány viszony) fogalmának általánosításán alapuló grafikus (UML jelölésrendszert használó) metamodellezési koncepciót (VPM: Visual and Precise Metamodeling) modellek és metamodelljeik egységes kezelésére.*

- ***Statikus modellelemek finomítása.*** *A MOF egy minimális elemkészletére építve (osztályok/entitások, asszociációk/relációk és attribútumok/függvények) kidolgoztam a statikus modellelemek finomításának (azaz öröklésének és példányosításának) axiómáit (2.4 fejezet), melyek az osztályok öröklésén kívül tartalmazzák az asszociációk és metamodellek finomítási szabályait is.*

- ***Statikus konzisztencia analízis.*** *Kidolgoztam egy, a VPM modellelemek öröklési és példányosítási relációk által definiált részben rendezésén alapuló módszert (2.5 fejezet), mellyel formálisan kimutatható és automatikusan feloldható a statikus finomítási axiómák megsértése VPM modellekben.*

- ***VPM modellek algebrai leírása.*** *Elkészítettem a VPM modellek meta-szintű és modell-szintű algebrai reprezentációját, valamint jólformáltsági kritériumait (3.3.1, 3.3.2 és 3.5 fejezet) az absztrakt állapotgépek [Gur95] matematikai formalizmusának felhasználásával.*

- ***Elemi VPM műveletek.*** *A VPM modellek algebrai reprezentációjára építve olyan elemi műveleteket vezettem be (3.4 fejezet), melyek bizonyíthatóan garantálják a VPM modellek konzisztens manipulációját.*

- ***Statikus és dinamikus modellelemek szétválasztása.*** *Bevezettem a metamodellekben a hagyományos statikus modellelemeken kívül a dinamikus modellelemeket (2.2.1 fejezet), melyek nélkülözhetetlenek a modellezési nyelvek dinamikus szemantikájának formális specifikációjakor.*

A tézis a disszertáció 2. és 3. fejezetén, valamint a [3, 4, 12, 13, 18, 27–29] publikációkon alapul.

Kiemelendő, hogy az öröklési és példányosítási viszonyok dinamikusan változhatnak a modellek evolúciója folyamán, ezáltal a viselkedést leíró szabályok (lásd 2. tézis) is modellként tárolhatók és manipulálhatók, így tulajdonképpen a Neumann-elvet valósítottam meg a VPM metamodellezés módszertanában.

3.2. Modellezési nyelvek és transzformációik formális specifikációja

Megvizsgáltam és értékeltem az UML nyelv szakirodalmában fellelhető modelltranszformációs megközelítéseket (pl. [Mil02, AK02, HKT02, Whi02, dLV02]). Megállapítottam, hogy a mérnöki gyakorlathoz legjobban illeszkedő megoldások grafikus megközelítést használnak a transzformációk specifikációjára, a metamodellezés és a gráftranszformáció módszertanára építve. Ugyanakkor azt tapasztaltam, hogy (i) problematikus a létező grafikus formalizmusok adaptálása a VPM metamodellezési módszeréhez, (ii) gondot jelent a visszavetítés támogatása, (iii) a dinamikus viselkedés specifikációjának újrafelhasználása szintén nem támogatott, továbbá (iv) a modelltranszformációk tervezése ad hoc módon történik.

2. tézis. *A gráftranszformáció paradigmájára építve kidolgoztam egy grafikus, általános és matematikailag precíz formalizmust, mely egyidejűleg támogatja tetszőleges modellezési nyelv operációs szemantikájának metaszintű (nyelvszintű) definiálását, valamint tetszőleges modellezési nyelven belüli illetve nyelvek közötti modelltranszformációk magasszintű specifikálását.*

- ***Gráftranszformációs szabályok szemantikai leírása absztrakt állapotgépekkel.*** *Kidolgoztam a gráftranszformációs szabályok egy matematikailag precíz leírását absztrakt állapotgépek felhasználásával (4.3 fejezet), mellyel operációs módon definiálhatók a legfőbb gráftranszformációs megközelítések.*

- ***Vezérlési szerkezetek.*** *Egy speciális vezérlési folyam gráf formájában definiáltam a gráftranszformáció nemdeterminizmusát megszorítani hivatott vezérlési szerkezeteket (4.4.1 fejezet).*

- ***Visszavetítés támogatása.*** *A formális analízis eredményeinek visszavetítésére bevezettem a referencia metamodellek és modellek fogalmát (4.4.4 fejezet), mely lehetővé teszi a forrás- és célmodellek szabatos összekapcsolását.*

- ***Dinamikus viselkedés finomítása.*** *Kidolgoztam a gráftranszformációs szabályok finomítását (4.5 fejezet), mely lehetővé teszi a dinamikus viselkedés újrafelhasználását más modellezési nyelvek operációs szemantikájának definiálásakor illetve modelltranszformációk specifikációja során.*

A tézis a disszertáció 4. fejezetén, valamint a [2–4, 8, 11, 12, 14, 16, 18–21, 24–27, 29] publikációkon alapul.

3.3. Automatikus modell- és programgenerálás

A létező gráftranszformáció alapú modelltranszformációs megközelítéseket és eszközöket áttekintve azt tapasztaltam, hogy a modelltranszformációk automatizálására az általános

célú gráftranszformációs eszközök (elvileg) felhasználhatók ugyan (lásd pl. [HKT02]), ám az általuk nyújtott támogatás nem elégséges nagyméretű UML modelleken végrehajtott komplex modelltranszformációk esetén, mivel általában nehezen illeszthetők létező UML eszközökhöz. Problematikus továbbá, hogy ezen eszközök mind kutatási prototípusok, így az általuk generált transzformációs programok nem használhatóak fel közvetlenül ipari transzformációkra.

3. tézis. *A kidolgozott modelltranszformációs formalizmus megvalósítását támogatandó, automatikus modell- és programgenerálási módszereket dolgoztam ki, mely által a modelltranszformációk precíz matematikai specifikációiból az azt megvalósító program automatikusan származtatható, és az adott nyelv (vagy nyelvek) tetszőleges modelljére végrehajtható.*

- ***Modelltranszformáció alapú automatikus programgenerálás.*** *A modelltranszformációk implementációs programjainak automatikus szintézisének céljából kidolgoztam egy szukcesszív modelltranszformációkon alapuló reflektív programgenerálási módszert (5.2.3 fejezet).*
- ***Modelltranszformációs virtuális gép Prolog megvalósítása.*** *Kidolgoztam egy, tetszőleges modelltranszformációt végrehajtó virtuális gép Prolog alapú megvalósításának egy módszerét, melyben (i) a gráftranszformációs szabályok Prolog megvalósítása egy mélységi gráfbejáráson és a Prolog egyesítési mechanizmusán alapuló gráfmintaillesztési algoritmuson alapul (5.2.1 fejezet), míg (ii) a vezérlési folyam gráf megvalósítása a Prolog metaprogramozási adottságait aknázza ki (5.2.2 fejezet).*
- ***Modelltranszformáció UML Akció Szemantikai leírása.*** *Elkészítettem a modelltranszformációk szabványos, UML Akció Szemantika alapú reprezentációját (5.4 fejezet), melynek során a gráftranszformációs szabályok implementációját egy lokális keresésen alapuló algoritmussal, a vezérlési folyam gráfot pedig az UML Akció Szemantika beépített összetett akcióival valósítottam meg.*

A tézis a disszertáció 5. fejezetén, valamint a [5, 11, 22, 23, 27] publikációkon alapul.

3.4. Grafikus modellezési nyelvek és modelltranszformációk formális verifikációja

Egy komplex modelltranszformációs rendszerrel szemben követelmény a modellek és transzformációik formális és automatizált helyességellenőrzése. A modelltranszformáció szakirodalmát áttekintve megállapítottam, hogy a modelltranszformációk helyesség- és teljességellenőrzésének szükségességét és kérdését először (társszerzőimmel együtt) én tárgyaltam a [2, 15, 19] cikkekben. Megvizsgáltam továbbá a gráftranszformációs rendszerek formális analízisére létező módszereket és eszközöket, mellyel egy modell dinamikus viselkedése írható le. Megállapítottam, hogy bár a gráftranszformációs rendszerek helyességvizsgálatának elméleti problémáit nemrégiben többen tárgyalták [BK02, PE02, Hec98], nem létezett olyan általános, a konkrét modellezési nyelvtől független megközelítés, mely közvetlenül a gyakorlatban is alkalmazható lenne. Ugyanakkor, a

modellellenőrző eszközök esetén gyakorta tapasztalt állapottér-robbanás miatt a kidolgozott módszerek gyakorlati alkalmazhatóságának vizsgálata elengedhetetlen lenne.

4. tézis. *Kidolgoztam a grafikus modellezési nyelvek (metamodellezésen és gráftranszformáción nyugvó) operációs szemantikájának létező modellellenőrző eszközök felhasználásával elvégezhető verifikációját tetszőleges modellezési nyelv tetszőleges modellje esetén. Definiáltam továbbá a modelltranszformációk helyességének kritériumrendszerét, és módszereket adtam modelltranszformációk szintaktikus (nyelvi tartalmazást bizonyító) és szemantikus (tulajdonságmegőrzést igazoló) helyességének formális ellenőrzésére.*

- ***Bizonyítottan helyes tranzíciós rendszer reprezentáció.*** *Tetszőleges gráftranszformációs rendszerre kidolgoztam annak egy olyan tranzíciós rendszer alapú reprezentációját (6.3.1 és 6.3.2 fejezet), mely az eredeti gráftranszformációs rendszerrel bizonyíthatóan ekvivalens működést valósít meg.*

- ***Optimalizált modell-leírás.*** *Számos optimalizálást építettem be a generált tranzíciós rendszer modelljébe (6.3.3 fejezet), kihasználva, hogy a gráftranszformációs szabályok kizárólag a dinamikus modellelemeket módosítják, így a statikus modellrészek elkódolása egy fordítási idejű előfeldolgozási szakasz után elhagyható.*

- ***Verifikációs teljesítményanalízis.*** *Verifikációs benchmark problémákon demonstráltam a módszer gyakorlati alkalmazhatóságát és felmértem a kapott tranzíciós rendszeren elvégzett modellellenőrzés hatékonyságát (6.5 fejezet). Konklúzióként megalkottam a "kevés komplex jobb, mint a sok egyszerű" ökölszabályt, mely gyakorlati támpontot nyújt a modellezési nyelv tervezője számára a verifikáció szempontjából hatékony gráftranszformációs szabályok megalkotására.*

- ***Modelltranszformációk helyességének kritériumrendszere.*** *Kidolgoztam a modelltranszformációk helyességének egy általános kritériumrendszerét (7.1.1 fejezet), amelynek minden egyes modelltranszformáció bizonyíthatóan meg kell, hogy feleljen.*

- ***Transzformációk szintaktikus helyesség- és teljességellenőrzése.*** *Kidolgoztam egy tervkészítő (planner) algoritmusokon alapuló technikát a modelltranszformációk szintaktikus helyességének és teljességének vizsgálatára (7.2 fejezet), mellyel ellenőrizhető, hogy a transzformáció eredményéül kapott célmodell valóban a célnyelv egy jólformált modellje-e.*

- ***Modelltranszformációk szemantikus helyességvizsgálata.*** *Kidolgoztam egy általános és automatikus keretrendszert (7.3 fejezet), mellyel (modellellenőrző eszközök segítségével) formálisan is vizsgálható, hogy egy modelltranszformáció egy adott forrásmodellről a célmodellre megőriz-e előre definiált (nyelvfüggő) szemantikus konzisztencia kritériumokat.*

A tézis a disszertáció 6. és 7. fejezetén, valamint a [2, 4, 6, 7, 9, 10, 14, 15, 17, 19, 20] publikációkon alapul.

Megállapítható, hogy az általam javasolt módszer a szakirodalomban létező első komplex gyakorlati feladatokon is alkalmazott nyelvfüggetlen megoldás a modellezési nyelvek modellellenőrzésen alapuló formális helyességvizsgálatára.

4. Az új tudományos eredmények hasznosíthatósága

Kutatásommal párhuzamosan számos olyan fejlesztési (mérnöki) munka is folyt (részint általam, részint más doktorandusz, diplomatervező, és önálló laboros hallgatók által), amelynek célja az volt, hogy bebizonyítsa az új tudományos eredmények gyakorlati, ipari hasznosíthatóságát.

Modelltranszformációs keretrendszer implementációja és alkalmazásai. A létező ipari szabványokhoz idomulva megterveztem és Prolog nyelven implementáltam a VIATRA modelltranszformációs rendszert, mely szabványos XMI formátumban adott modellek transzformációját végzi UML-ben megadott (a fent definiált matematikai háttérnek megfelelő) szabályleírásokból automatikusan generálható és végrehajtható Prolog programok formájában a a 2. és 3. tézis eredményeit felhasználva.

A VIATRA modelltranszformációs rendszer felhasználásával az IKTA 065/2000 projekthez (“Keretrendszer nagymegbízhatóságú, biztonságkritikus rendszerek fejlesztéséhez és teszteléséhez”) kapcsolódóan modelltranszformációkat terveztem meg és implementáltam, melyeket ipari partnerektől kapott modelleken is teszteltem. Ez alapján leszűrhető, hogy futási idő szempontjából már egy akadémiai prototípus is kielégítőnek bizonyult valós méretű modellek esetén is.

Modellezési benchmarkok. A kidolgozott transzformációs formalizmus kifejezőerejét és gyakorlati alkalmazhatóságát demonstrálandó, elkészítettem az igen elterjedten használt Petri hálók és a Kiterjesztett Hierarchikus Automaták (mely utóbbi az UML állapottérképek szemantikai formalizálása) modellezési nyelvének operációs szemantikáját olyan modelltranszformációs rendszerek formájában, melyek bizonyíthatóan ekvivalensek (azonos viselkedést valósítanak meg) a szakirodalomból ismert tradicionális szemantikai leírásokkal.

Modelltranszformációk tervezési módszertana. Kidolgoztam továbbá komplex modelltranszformációk tervezésének egy metodikáját, mely definiálja a tervezési ciklus főbb lépéseit, metamodell szinten bevezetve a statikus és dinamikus, valamint a származtatott nyelvi elemek fogalmát, és felosztva a transzformációs ciklust egy inicializáló és egy végrehajtó szakaszra.

Modellezési nyelvek modellellenőrzése. Elkészült a CheckVML nevű rendszer [10], mely a 4. tézis felhasználásával a bemenetként kapott metamodell, modell és gráftranszformációs szabályok alapján egy Promela nyelvű modellt hoz létre, mely a

SPIN [Hol97] modellellenőrző eszköz bemenetéül szolgál. Ezután a bizonyítandó temporális logikai állítást közvetlenül a SPIN rendszerben megadva tetszőleges modellezési nyelv tetszőleges (konkrét) modelljére automatikusan eldönthető, hogy az adott tulajdonság fennáll-e.

Folyamatban lévő fejlesztések, további tervek. Folyamatban van egy, a VPM alapjain (1. tézis) nyugvó metamodellező rendszer kifejlesztése és integrálása a VIATRA rendszerbe. E prototípus kifejlesztésének célja az ún. Meta-CASE eszközök egy olyan új generációjának a létrehozása, mely matematikai precizitást, dinamikus viselkedésleírást és nagyfokú újrafelhasználhatóságot biztosít a grafikus modellezési nyelvek igen széles körében.

A további tervek között szerepel a modelltranszformációs specifikációk átültetése olyan UML CASE eszközökbe, melyek támogatást nyújtanak a szabványos Akció Szemantikai leírások futtatására (követve a 3. tézis eredményeit), ezáltal a kidolgozott matematikai módszerek közvetlenül integrálhatók lesznek ipari környezetben is.

5. Kapcsolódó publikációk

Könyvrész

[1] D. Varró. UML modeling of web applications. In C. Garcia and A. Pataricza, editors, *E-Business for SMEs*. Prentice Hall, 2004. Reviewed by the publisher and the editors.

Külföldön megjelent, idegennyelvű folyóiratcikk

[2] D. Varró, G. Varró, and A. Pataricza. Designing the automatic transformation of visual languages. *Science of Computer Programming*, 44(2):205–227, August 2002, Elsevier.

[3] D. Varró and A. Pataricza. VPM: A visual, precise and multilevel metamodeling framework for describing mathematical domains and UML. *Journal of Software and Systems Modelling*, (2)3:187–210, October 2003, Springer.

[4] D. Varró. Automated formal verification of visual modeling languages by model checking. *Journal of Software and Systems Modelling*, 2003, Springer. Accepted to the Special Issue on Graph Transformation and Visual Modelling Techniques.

Magyarországon megjelent, idegennyelvű folyóiratcikk

[5] D. Varró and A. Pataricza. UML Action Semantics for model transformation systems. *Periodica Polytechnica*, 2003. In press.

Nemzetközi konferenciakiadványban megjelent idegen nyelvű előadás

[6] L. Baresi, R. Heckel, S. Thöne, and D. Varró. Modeling and analysis of architectural styles. In *Proc ESEC 2003: European Software Engineering Conference*, pages 68–77, Helsinki, Finland, September, 2003, ACM Press.

[7] L. Baresi, R. Heckel, S. Thöne, and D. Varró. Modeling and analysis of architectural styles based on graph transformation. In *The 6th ICSE Workshop on Component Based Software Engineering: Automated Reasoning and Prediction*, Portland, Oregon, USA, May 3-4.

[8] Gy. Csertán, G. Huszerl, I. Majzik, Zs. Pap, A. Pataricza, and D. Varró. VIATRA: Visual automated transformations for formal verification and validation of UML models. In *Proc. ASE 2002: 17th IEEE Intern. Conf. on Automated Software Engineering*, pages 267–270, Edinburgh, UK, September 23–27 2002. IEEE Press.

[9] G. Salamon, D. Varró, and A. Pataricza. Formal verification of model transformation systems. In *EDCC 2002: Fourth European Dependable Computing Conference: Fast Abstracts*, pages 15–16, Toulouse, France, October 23–25 2002.

[10] Á. Schmidt and D. Varró. CheckVML: A tool for model checking visual modeling languages. In P. Stevens, J. Whittle and G. Booch, editors. *Proc. UML 2003: 6th International Conference on the Unified Modeling Language*, volume 2863 of LNCS, pages 92–95, San Francisco, CA, USA, October 20-24 2003, Springer.

[11] D. Varró. Automatic program generation for and by model transformation systems. In H.-J. Kreowski and P. Knirsch, editors, *Proc. AGT 2002: Workshop on Applied Graph Transformation*, pages 161–173, Grenoble, France, April 12–13 2002.

[12] D. Varró. A formal semantics of UML Statecharts by model transition systems. In A. Corradini, H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proc. ICGT 2002: 1st International Conference on Graph Transformation*, volume 2505 of *LNCS*, pages 378–392, Barcelona, Spain, October 7–12 2002. Springer-Verlag.

[13] D. Varró. A pattern-based constraint language for metamodels. In T. Csendes, editor, *Proc. CSCS 2002: The Third Conference of PhD Students in Computer Science*, page 109, Szeged, Hungary, July 1–4 2002.

[14] D. Varró. Towards symbolic analysis of visual modelling languages. In P. Bottoni and M. Minas, editors, *Proc. GT-VMT 2002: International Workshop on Graph Transformation and Visual Modelling Techniques*, volume 72 of *ENTCS*, pages 57–70, Barcelona, Spain, October 11-12 2002. Elsevier.

[15] D. Varró. Towards formal verification of model transformations. In T. Jones, editor, *PhD Student Workshop of FMOODS 2002, Formal Methods for Open Object-Based Distributed Systems*, Enschede, The Netherlands, March 20–22 2002.

[16] D. Varró, Sz. Gyapay, and A. Pataricza. Automatic transformation of UML models for system verification. In J. Whittle et al., editors, *WTUML'01: Workshop on Transformations in UML*, pages 123–127, Genova, Italy, April 7th 2001.

[17] D. Varró and A. Pataricza. Automated formal verification of model transformations. In J. Jürjens, B. Rumpe, R. France and E.B. Fernandez *Proc. CSDUML 2003: Workshop on Critical Systems Development in UML*, pages 63–78, San Francisco, CA, USA, October 20–24 2003. Published as Technical Report, TUM-I0323, September, Technische Universität München.

[18] D. Varró and A. Pataricza. Metamodeling mathematics: A precise and visual framework for describing semantics domains of UML models. In J.-M. Jézéquel, H. Hussmann, and S. Cook, editors, *Proc. Fifth International Conference on the Unified Modeling Language – The Language and its Applications*, volume 2460 of *LNCS*, pages 18–33, Dresden, Germany, September 30 – October 4 2002. Springer-Verlag.

[19] D. Varró, G. Varró, and A. Pataricza. Designing the automatic transformation of visual languages. In H. Ehrig and G. Taentzer, editors, *GRATRA 2000 Joint APPLIGRAPH and GETGRATS Workshop on Graph Transformation Systems*, pages 14–21, Berlin, Germany, March 25–27 2000.

[20] D. Varró, G. Varró, and A. Pataricza. Visual graph transformation in system verification. In E. Gramatova, H. Manhaeve, and A. Pawlak, editors, *DDECS 2000 Design and Diagnostics of Electronic Circuits and Systems*, pages 137–141, Bratislava, Slovakia, April 5–7 2000.

Magyar nyelvű konferencia-előadás

[21] D. Varró. Visual automated model transformation. In *Mini–Symposium 2001*, pages 48–49, Budapest University of Technology and Economics, Department of Measurement and Information Systems, January 31 – February 1 2001. IEEE Hungary Section (BUTE Student Branch).

[22] D. Varró. Automated program generation in VIATRA. In *Mini–Symposium 2002*, pages 34–35, Budapest University of Technology and Economics, Department of Measurement and Information Systems, February 4–5 2002. IEEE Hungary Section (BUTE Student Branch).

[23] D. Varró. UML Action Semantics for model transformation systems. In *Mini–Symposium 2003*, pages 22–23, Budapest University of Technology and Economics, Department of Measurement and Information Systems, February 2003. IEEE Hungary Section (BUTE Student Branch).

[24] G. Huszerl, I. Majzik, Zs. Pap, D. Petri, A. Pataricza, and D. Varró. Keretrendszer nagymegbízhatóságú, biztonságkritikus rendszerek fejlesztéséhez és teszteléséhez. In *OOOK 2002: 5. Országos Objektum-Orientált Konferencia*, Dobogókõ, Hungary, October 16–17 2002. In Hungarian.

[25] D. Varró and A. Pataricza. UML modellek automatikus transzformációi. In *OOOK 2002: 5. Országos Objektum-Orientált Konferencia*, Dobogókõ, Hungary, October 16–17 2002. In Hungarian.

[26] D. Varró and P. Domokos. A VIATRA modelltranszformációs rendszer. In Bitay E. (editor) *FMTÜ 2003: Fiatal Magyarok Tudományos Ülésszaka*, pages 51–54, Kolozsvár, Románia, March 21–22, 2003, Erdélyi Múzeum Egyesület. In Hungarian.

Elektronikus publikáció (Technical report)

[27] D. Varró and A. Pataricza. Mathematical model transformations for system verification. Technical report, Budapest University of Technology and Economics, May 2001.

[28] D. Varró and A. Pataricza. A unifying semantic framework for multilevel metamodelling. Technical report, Budapest University of Technology and Economics, October 2001.

[29] D. Varró and A. Pataricza. Formalizing model transformation systems by abstract state machines. Technical report, Budapest University of Technology and Economics, June 2003.

6. Hivatkozott irodalom

[AK02] D. Akehurst and S. Kent. A relational approach to defining transformations in a metamodel. In J.-M. Jézéquel, H. Hussmann, and S. Cook, editors, *Proc. Fifth Intern. Conf. on the Unified Modeling Language – The Language and its Applications*, volume 2460 of *LNCS*, pages 243–258, Dresden, Germany, September 30 – October 4 2002. Springer-Verlag.

[BDCL+01] A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza, and G. Savoia. Dependability analysis in the early phases of UML based system design. *International Journal of Computer Systems - Science & Engineering*, 16(5):265–275, 2001.

[BK02] P. Baldan and B. König. Approximating the behaviour of graph transformation systems. In A. Corradini, H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proc. ICGT 2002: First International Conference on Graph Transformation*, volume 2505 of *LNCS*, pages 14–29, Barcelona, Spain, October 7–12 2002. Springer.

[BMM99] A. Bondavalli, I. Majzik, and I. Mura. Automatic dependability analyses for supporting design decisions in UML. In *Proc. HASE'99: The 4th IEEE International Symposium on High Assurance Systems Engineering*, pages 64–71, 1999.

[dLV02] J. de Lara and H. Vangheluwe. AToM3: A tool for multi-formalism and meta-modelling. In R.-D. Kutsche and H. Weber, editors, *5th Intern. Conf.*,

FASE 2002: Fundamental Approaches to Software Engineering, Grenoble, France, April 8-12, 2002, Proceedings, volume 2306 of *LNCS*, pages 174–188. Springer, 2002.

[Gur95] Y. Gurevich. Evolving Algebras 1993: Lipari Guide. In E. Börger (editor), *Specification and Validation Methods*, Oxford University Press, 1995.

[Hec98] R. Heckel. Compositional verification of reactive systems specified by graph transformation. In *Proc. FASE: Fundamental Approaches to Software Engineering*, volume 1382 of *LNCS*, pages 138–153. Springer, 1998.

[HKT02] R. Heckel, J. Küster, and G. Taentzer. Towards automatic translation of UML models into semantic domains. In *Proc. AGT 2002: Workshop on Applied Graph Transformation*, pages 11–21, Grenoble, France, April 2002.

[Hol97] G. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[LMM99] D. Latella, I. Majzik, and M. Massink. Automatic verification of UML statechart diagrams using the SPIN model-checker. *Formal Aspects of Computing*, 11(6):637–664, 1999.

[Mil02] D. Milicev. Automatic model transformations using extended UML object diagrams in modeling environments. *IEEE Transactions on Software Engineering*, 28(4):413–431, April 2002.

[OMG03] Object Management Group. *Meta Object Facility Version 2.0*. `http://www.omg.org`.

[PE02] J. Padberg and B. J. Enders. Rule invariants in graph transformation systems for analyzing safety-critical systems. In A. Corradini, H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proc. ICGT 2002: Firs International Conference on Graph Transformation*, volume 2505 of *LNCS*, pages 334–350, Barcelona, Spain, October 7–12 2002. Springer.

[PL99] I. Paltor and J. Lilius. vUML: A tool for verifying UML models. In R. J. Hall and E. Tyugu, editors, *Proc. of the 14th IEEE Intern. Conf. on Automated Software Engineering, ASE'99*. IEEE, 1999.

[RJB99] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

[Whi02] J. Whittle. Transformations and software modeling languages: Automating transformations in UML. In J.-M. Jézéquel, H. Hussmann, and S. Cook, editors, *Proc. Fifth Intern. Conf. on the Unified Modeling Language – The Language and its Applications*, volume 2460 of *LNCS*, pages 227–242, Dresden, Germany, September 30 – October 4 2002. Springer-Verlag.