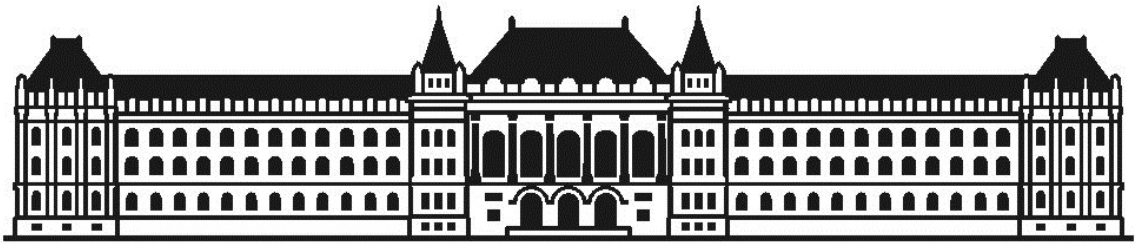


Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Automated Offline Verification of Graph Rewriting-Based Model Transformations

---

Gráfújraírás-Alapú Modelltranszformációk Helyességének Offline Ellenőrzése

Ph.D. Értekezés Tézisei

Asztalos Márk
okleveles mérnök-informatikus

Tudományos vezetők:
Levendovszky Tihamér Ph.D.
Lengyel László Ph.D.

Budapest, 2012.

Ph.D. Értekezés Tézisei

Asztalos Márk

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

1117 Budapest, Magyar tudósok krt. 2. (Q épület) .

e-mail: asztalos@aut.bme.hu
tel: +36(1)463-42-25

Tudományos vezetők:

Dr. Levendovszky Tihamér, Ph.D.
Dr. Lengyel László, Ph.D.

1. Előzmények és célkitűzések

Napjainkban a grafikus modellek alkalmazása egyre elterjedtebb a szoftverfejlesztésben. A modellalapú fejlesztés jelentős mértékben növeli a programok átláthatóságát, csökkenti a fejlesztés idejét és automatizált fejlesztési módszerek alkalmazását [SVC06] teszi lehetővé. Továbbá a modellek a szöveges programnyelveknél magasabb absztrakciós szintet képviselnek, ez előnyös nagyméretű és komplex rendszerek tervezése esetén. A modelleket ezért gyakran az adott alkalmazási terület szakértői is megértik, ez egy fontos előrelépés a szoftverleíró nyelvek fejlődésében. A szoftverfejlesztés során használt leggyakoribb modellek általában csomópontokból és élekből állnak, ezért ezeket gráfokkal szoktuk leírni [EEPT06]. A csomópontok reprezentálják az adott szakterületen megjelenő entitásokat, míg az élekkel a közöttük meglévő relációkat, kapcsolatokat írjuk le. Az így definiált csomópontok és élek attribútumokkal rendelkeznek, melyekkel az adott elem tulajdonságait specifikálják. A gráfokkal történő leírás lehetővé teszi a modellek grafikus ábrázolását, akár szakterület-specifikus konkrét szintaxissal is. Megjegyezzük, hogy bár a szoftverfejlesztés során sokszor grafikus modellekkel dolgozunk, természetesen a szöveges modellek is gyakran formalizálhatók hasonlóan.

A Unified Modeling Language (UML) [FS99, Obj07] az objektumorientált rendszerek statikus felépítésének és dinamikus viselkedésének modellezésére leggyakrabban használt nyelvcsalád. Az UML-ben definiált nyelvek általánosak, azaz nem kötődnek konkrét szakterülethez, ezért bizonyos esetekben nehéz velük konkrét szakterületekhez szorosan kötődő koncepciókat kifejezni. Ezért az UML mellett gyakran használunk szakterület-specifikus modellező nyelveket (Domain-Specific Modeling Language – DSML) [KT08] a modellek specifikálásához. Ezek a nyelvek az adott szakterületen megjelenő entitások definícióját adják meg, hozzájuk a szakterület szabályainak megfelelő kényszereket írnak le, és gyakran szakterület-specifikus jelölésrendszert is biztosítanak. Mivel egy DSML egy konkrét alkalmazási területhez kötődik, ezen a területen belül hatékonyabb vele a fejlesztés, mint egy általános célú rendszerrel. Grafikus DSML-ek definiálásnak az egyik legelterjedtebb módja a *metamodellezés*. Ilyenkor egy speciális modell (*metamodell*) rögzíti, hogy a (példány-)*modell*ekben milyen típusú elemek jelenhetnek meg, és azokra milyen topológiai, vagy az attribútumokat érintő megkötések érvényesek.

A legtöbb modellalapú megközelítésben [KWB03, SK97] a modellekkel történő munka során gyakran használunk automatizáltan futtatott modellfeldolgozó programokat [SK03, BBG+06, MCG05], amelyeket *modelltranszformáció*knak is nevezünk. Tipikusan ilyen programok segítségével történik például (i) a kódgenerálás, amikor végrehajtható forráskódot generálunk UML diagramokból, vagy más szakterület-specifikus modellekből (a forráskód is reprezentálható modellként), (ii) magasabb absztrakciós szinten definiált modellekből alacsonyabb szintű modellek automatikus előállítása, vagy az ellenkező irányban alacsonyabb szintű modellek visszafejtése, (iii) a modellek refaktorálása (iv) a modellek közötti szinkronizáció (v) modellek operációs szemantikájának leírása [dLV10]. Egy modellfeldolgozó programot természetesen bármilyen hagyományos programozási nyelven is megírhatunk, azonban – mivel a modelleket is egy magasabb absztrakciós szinten specifikáljuk – előnyös lehet olyan nyelvek használata, melyek segítségével a feldolgozás lépéseit a modellekkel azonos absztrakciós szinten tudjuk kifejezni. Gráfalapú modellek feldolgozásának formális hátterét általában az algebrai gráfújraírás [EEPT06, Hec06] elméletével formalizálják. Az algebrai gráftranszformációk esetében a feldolgozás elemi lépéseit az *újraírási szabályok* rögzítik. Ezek végrehajtását legtöbbször kategóriaelméleti [BW90] konstrukciók segítségével formalizálják, a leggyakrabban használt módszerek a Single Pushout (SPO), vagy Double Pushout (DPO) megközelítések. Általánosan, egy gráfújraírási szabályt két gráf (baloldali – left-hand side (LHS) és jobboldali – right-hand-side (RHS)) definiál. Amikor a bemeneti modellen alkalmazunk egy szabályt, megkeressük benne a baloldal egy izomorf példányát (ezt a folyamatot nevezzük illesztésnek), és azt kicseréljük a jobboldal egy példányával. A gráfújraírás szakirodalmában a „transzformáció” kifejezést használják a szabályok egy sorozatának egy konkrét végrehajtására is, de én a korábban bemutatott, a modellvezérelt szoftverfejlesztés területén belül megszokott jelentést értem alatta, vagyis egy mo-

delltranszformáció egy modellfeldolgozó programnak a definíciója, nem pedig annak egy végrehajtása. A gráfújraírási szabályok specifikálása mellett egy gráfújraírás alapú modelltranszformáció rögzíti a szabályok végrehajtási sorrendjét, ezt nevezzük a transzformáció vezérlő mechanizmusának. Többféle módszer is létezik ennek specifikálására [BFG96, SV09], ezek közül az egyik, amikor a szabályok sorrendjét egy irányított vezérlésfolyamgráf rögzíti, amelynek a csomópontjai a szabályok.

A hozzá kapcsolódó formális módszerek eredményeképpen a modellalapú szoftverfejlesztés jelentős szerepet játszik olyan rendszerek fejlesztésében, amelyek verifikációja, azaz helyes működésének bizonyítása kiemelkedő fontosságú [GH06, WML10]. Ilyen biztonságkritikus rendszerekre van szükség több ipari alkalmazási területen is, többek között az autóiparban, repülőgépiparban, vagy orvosi alkalmazások készítése során [GHN10, WL06]. A formális modellek használatának és a modellfeldolgozó programok gyakori és automatizált használatának elterjedésével, a modelltranszformációk helyes működésének ellenőrzése is egyre fontosabbá válik [SK03, KWB03]. Egy modelltranszformáció verifikálása során biztosítani szeretnénk, hogy a program helyesen működik, így nem kell minden egyes végrehajtás után annak eredményét ellenőrizni. A verifikáció során tehát megmutatjuk, hogy a transzformáció megfelel bizonyos funkcionális és nemfunkcionális követelményeknek. A funkcionális követelményeket gyakran a kimeneti modellen fogalmazzuk meg, vagy a bemeneti és a kimeneti modellek közötti kapcsolatra mondjuk ki, vagyis a funkcionális követelmények azt rögzítik, hogyan kell a transzformációnak a bemenetet feldolgoznia. A leggyakrabban vizsgált nemfunkcionális követelmények a *terminálás* és a *determinizmus*. A terminálás, vagyis az, hogy a program tetszőleges bemenet esetén véges időn belül befejezze a futását, minden szoftver esetén fontos tulajdonság, azonban gráfújraírás-alapú modelltranszformációk esetén bizonyos esetekben ennek bizonyítása nehezebb, mint hagyományos programok esetén, és speciális módszereket igényel. Ugyanis a gráfújraírás természetéből adódóan lehetnek nemdeterminisztikus lépések a feldolgozás során. Például egy szabály baloldalának illesztése során, a legtöbb rendszerben véletlenszerűen választjuk ki a baloldalnak azt a példányát, amelyen alkalmazzuk az újraírást. A determinizmus, vagy *konfluencia* az előbb elmondottak miatt szintén külön vizsgálatot igényelhet, mert egy programfejlesztő általában olyan céllal ír meg egy programot, hogy az azonos bemenet esetén ugyanazt a kimenetet produkálja.

A gráfújraírás elméleti háttere lehetővé teszi a gráftranszformáció-alapú modellfeldolgozó-programok verifikációját formális módszerek segítségével [ALS+12]. Ezeket a módszereket három kategóriába sorolhatjuk attól függően, hogy az általuk belátott eredmények függetlenek-e a konkrét bemenettől, illetve egy konkrét végrehajtásától.

(i) *Online* validációnak azt nevezzük, amikor a transzformáció egy konkrét végrehajtásának eredményét validálja a végrehajtó motor [Len06]. Azaz, a rendszer biztosítja, hogy amennyiben a transzformáció sikeresen befejeződik, akkor kimenet megfelel a követelményeknek, egyéb esetben a lefutás sikertelen lenne. Azt azonban nem tudjuk biztosítani, hogy transzformáció sikeresen lefusson.

(ii) A *statikus* ellenőrzés során egy modelltranszformáció helyességét egy konkrét bemenet esetén bizonyítjuk, de az ellenőrzés a transzformáció végrehajtása nélkül történik. Modelltranszformációk segítségével gyakran szoktunk gráfnyelvtanokat definiálni, amelyek egy konkrét kiindulási modellből és újraírási szabályok egy halmazából állnak. A szabályok tetszőleges sorrendű alkalmazásával vizsgáljuk, hogy a kezdő modellből milyen állapotok érhetők el, ez a módszer a szöveges nyelvtanok módszeréből származik [Roz97a]. Ez a példa jól mutatja az online és a statikus módszerek közötti különbséget. Az utóbbi általánosabb eredményt biztosít, hiszen egy adott bemenet összes lehetséges végrehajtásáról állít tulajdonságokat. Statikus analízisre mutat példát [KN08].

(iii) Az *offline* ellenőrzés során csak a transzformáció definícióját és a modelleket leíró nyelvek specifikációját figyelembe véve látunk be tulajdonságokat. Ezért az offline ellenőrzés eredménye független a bemeneti modelltől, és minden lehetséges végrehajtás esetén igaz lesz. Ezért az offline verifikációt mindössze egyszer kell elvégezni. Természetesen, az offline ellenőrzés

jóval bonyolultabb feladat, mint az előző két módszer. Érdemes megemlíteni, hogy egy gráftranszformáció terminálása bizonyítottan eldönthetetlen [Plu98] általánosságban.

Mivel a modellfeldolgozást általában automatizáltan alkalmazzuk, e programok verifikációja egyre fontosabbá válik. A gráfújraírás-alapú modelltranszformációk módszere egy ígéretes megközelítés modellfeldolgozó programok definiálására a következők miatt:

(i) A grafikus modellező nyelveket és a példánymodelleket legtöbbször gráfokkal szokták formalizálni.

(ii) A gráfújraírás-alapú modelltranszformációk működése az algebrai gráfújraíráson és gráfnyelvtanok matematikai elméletén alapul, amely lehetővé teszi a formális vizsgálatukat.

(iii) A gráfújraírás-alapú modelltranszformációknak speciális struktúrája van, az elemi műveleteket az újraírási szabályok definiálják. Ez a felépítés lehetővé teszi, hogy a transzformációt a feldolgozandó modellel azonos absztrakciós szinten írjuk le, továbbá gyakran szakterület-specifikus konkrét szintaxist is használhatunk a megjelenítésükhöz. Ezáltal az adott terület szakértői könnyebben megérthetik a transzformáció működését.

Nyitott kérdések

Munkám során gráfújraírás-alapú modelltranszformációk offline helyesség-ellenőrzését kutattam. A szakirodalomban található verifikációs módszerek gyakran csak egy-egy transzformáció- vagy tulajdonságosztály vizsgálatát teszik lehetővé [BH07], és sokszor csak manuálisan végezhetők el. Ezért nagy jelentősége van azoknak a módszereknek, melyek szakterülettől függetlenül, automatizáltan alkalmazhatók. Egy gyakori módszer a formális ellenőrzés elvégzésére, hogy a modelltranszformáció definícióját lefordítják egy általános matematikai modellre, amelyre vizsgálatára már rendelkezésre állnak kidolgozott formális elemzőmódszerek. Azonban, ez a modell – mivel egy általános matematikai model – legtöbbször nem szimmetrikus az eredeti szakterület-specifikus leírással. Egy ilyen megközelítés hátránya, hogy az összetett funkcionális követelményeket tipikusan az eredeti szakterületen tudjuk kifejezni, míg a transzformációk vizsgálata egy másik modellen történik. Így egyrészt, a vizsgálandó követelményeket át kell fordítani az egyik területről a másikra, másrészt a második modell vizsgálata során megtalált hibák javításához az eredeti szakterületen is tudni kell értelmezni azokat. A nehézséget az okozza, hogy a két különböző terület nem feltétlenül szimmetrikus, így az egyik területen meghatározott tulajdonságok nehezen értelmezhetők a másik területen. A fenti módszerre egy tipikus példa, amikor a modelltranszformáció definícióját lefordítják valamilyen tételbizonyító rendszer bemenetére [ABK07]. Ezek alapján, a modelltranszformációk offline verifikációjának fontosabb nyitott kérdései az alábbi pontokban foglalhatók össze:

- Léteznek modelltranszformációk verifikációjára módszerek, melyek a transzformáció-definíciókat általános formális modellre fordítják le, amely elemzéséhez már rendelkezésre állnak módszerek. Azonban ilyenkor a szakterület-specifikus követelményeket is le kell képezni az általános modellre. Előnyös lenne egy olyan formalizmus kifejlesztése, mellyel kifejezhetőek lennének a verifikálandó tulajdonságok az eredeti szakterületen belül, és amellyel a verifikáció elvégezhető lenne az ezen formalizmus megtartásával.
- Szintén előnyös lenne, ha egy ilyen formalizmussal az újraírási szabályok is leírhatók lennének, és segítségével a szabályok formális elemzésének vizsgálata elvégezhető lenne. Így a modelltranszformációk vizsgálata egyetlen formalizmus segítségével történhetne meg.
- Hasonló módon szeretnénk a modelltranszformációk vezérlési mechanizmusát is leírni olyan módon, hogy lehetővé váljon az egyes újraírási szabályokra belátott tulajdonságok végigvezetése a teljes modelltranszformáción. Így lehetővé válna azon tulajdonságok levezetése, amelyek akkor teljesülnek, amikor a transzformáció befejezi a futását.
- A modelltranszformációk tulajdonságainak vizsgálata algoritmikusan nehéz, ezért hasznos lenne olyan részproblémák meghatározása, melyekre a verifikálandó tulajdonságok még

eldönthetők. Természetesen, ilyen részproblémák meghatározása a lehetséges transzformációk, vagy elemezhető tulajdonságok körének korlátozásával érhetők el. Azonban a legfontosabb szempont azon részproblémák megtalálása, melyek segítségével valós mérnöki problémák is megoldhatók.

- Nagyban növelné a módszerek hatékonyságát, ha támogatást tudnánk nyújtani a transzformációk ellenőrzése során gyakran felmerülő, visszatérő problémák megoldására. A modelltranszformációk implementálása során alkalmazhatnánk bizonyos mintákat, melyek eredményeképpen automatikusan biztosíthatnánk bizonyos követelmények teljesülését.
- Hasznos lenne, ha azokra a problémákra sikerülne ilyen mintákat megfogalmazni, melyek algoritmikusan nehezen elemezhetők. Ezen minták vizsgálatát szakértők segítségével előre, a konkrét modelltranszformációtól függetlenül lehetne elvégezni.

Célkitűzések

Kutatásom célja az volt, hogy formális, automatizált módszereket fejlesszek gráfújraírás-alapú modelltranszformációk funkcionális tulajdonságainak verifikációjához, és integráljam ezek a módszereket egy létező modelltranszformációs keretrendszerbe. Célkitűzéseim az alábbi pontokban foglalhatók össze:

- Célom volt, hogy általános (szakterület-független), formalizmust adjak modelltranszformációk leírásához, amely formalizmus segítségével elvégezhető azok formális elemzése, és amely nem korlátozza jelentősen az elemezhető modelltranszformációk, illetve a kifejezhető funkcionális tulajdonságok körét.
- Egy formális nyelv definiálása, mely képes kifejezni azokat a funkcionális tulajdonságokat, melyeket vizsgálni szeretnénk. A nyelv által kifejezhető tulajdonságok legyenek később kiterjeszthetők újabb típusokkal.
- Célom volt, hogy formálisan leírjam és támogassam olyan technikák automatizált alkalmazását, melyeket gyakran használunk modelltranszformációk kézzel történő elemzése során.
- Olyan (fél-)automatikus algoritmusok biztosítása, melyek képesek különálló újraírási szabályokat és modelltranszformációk különböző vezérlő mechanizmusait elemezni és ezek tulajdonságait bizonyítani. Mindezt a modelltranszformáció eredeti definícióján célszerű megvalósítani, hogy ne kelljen ezt más formalizmusra átalakítani.
- A modelltranszformációk offline ellenőrzésének algoritmikus nehézségei miatt fontos, hogy az automatizált algoritmusok képesek legyenek értelmezni és beépíteni a szakértő fejlesztők által elvégzett manuális vizsgálatok eredményeit.
- Célom volt, hogy az elméleti módszerek alapján egy valós modelltranszformációs eszköz keretében elkészítsem egy működő keretrendszer implementációját, és bemutassam, hogy ez a keretrendszer alkalmas mérnöki problémák megoldására.

2. Módszertani összefoglalás

A vázolt nyitott kérdések meghatározták kutatásaim irányát. Bevezetésképpen tanulmányoztam a modellalapú szoftverfejlesztés és modelltranszformációk különböző megközelítéseit [KT08, SK03, MCG05]. Az általam kifejlesztett formalizmus leíráshoz megvizsgáltam a gráfok definiálásához, illetve szakterület-specifikus nyelvek, metamodellek és modellek formális specifikációjához használt különböző módszereket [EEPT06]. A gráfok közül külön foglalkoztam az attributált gráfok, típusos gráfok, címkézett gráfok leírásával.

A modelltranszformációk működésének megértéséhez tanulmányoztam a gráfújraírási rendszerek, gráftranszformációk és gráfnyelvtanok [Roz97b] elméletét. A kategóriaelmélet [Pie91, BW90] a matematikának olyan ága, mely objektumok és közöttük lévő leképezések (morfizmusok) általános

(magasabb absztrakciós szinten történő) leírásával foglalkozik. A kategóriaelméletnek több jelentős számításelméleti alkalmazása is megtalálható. Ahogyan ez a modelltranszformációk elméletének kutatási területén megszokott, kategóriaelméleti eszközöket használtam az elméleti tételek és bizonyításaik tömör és formális megfogalmazására. Amint már korábban említettem, a gráfújraírás-alapú modelltranszformáció újraírási szabályokból és azok sorrendjét meghatározó vezérlési mechanizmusból áll. Ezért tanulmányoztam az újraírási szabályok definíciójának és alkalmazásának különböző formális leírásait, melyek közül a leggyakoribbak az SPO és a DPO módszerek. Ezen kívül megismertem a vezérlési mechanizmusok specifikálásának különböző megközelítéseit (például prioritásalapú sorrendezés, rétegalapú sorrendezés, explicit vezérlésfolyamgráf használata) [SV09].

Mielőtt módszeremet kifejlesztettem volna a modelltranszformációk elemzésére, megvizsgáltam több modelltranszformációs keretrendszert, különös tekintettel azokra, melyek a verifikációt eszközszinten támogatják [dLV02, Tae04, Agr03, CHM+02, KNNZ99]. Ezen kívül tanulmányoztam olyan publikált esettanulmányokat, amelyek speciális modelltranszformációk, vagy speciális tulajdonságok manuális verifikálását mutatják be [ALS+12, LBA10, Pen09]. Habár kutatásom során a modelltranszformációk funkcionális tulajdonságainak ellenőrzése volt az első számú célkitűzésem, tanulmányoztam a nemfunkcionális tulajdonságok (például terminálás, konfluencia) vizsgálatával foglalkozó módszereket is [LPE06, EEPT06, EEdL+05]. A fentieken kívül, magam is készítettem manuális elemzést több transzformációhoz is. Az esettanulmányok készítése és a szakirodalom tanulmányozása során szerzett tapasztalatok alapján fejlesztettem ki módszeremet és határoztam meg azt a formális nyelvet, amellyel a verifikálandó funkcionális tulajdonságok egy halmazát le lehet írni. Az általam fejlesztett nyelv a propozicionális logikán alapul, kidolgoztam hozzá egy következtetőrendszert, melyhez levezetési szabályokat definiáltam és bizonyítottam azok helyességét. A kalkulus kidolgozásához részletesen tanulmányoztam különböző matematikai logikai módszereket [HR04, BA93], különös tekintettel a propozicionális és nemkanonikus propozicionális logikák elméletét.

Az BME Automatizálási és Alkalmazott Informatikai Tanszékén részt vettem a Visual Modeling and Transformation System (VMTS) [VMT10] nevű többszintű modellező és modelltranszformációs keretrendszer fejlesztésében. A verifikációs keretrendszert, melynek elméleti alapjait formálisan meghatároztam, ebben a keretrendszerben valósítottam meg. Az elkészült szoftver lehetővé teszi modelltranszformációk (fél)automatikus formális helyesség-ellenőrzését.

Doktori kutatásom során iteratív és inkrementális módszer szerint haladtam. Az elméletben kidolgozott módszereket gyakorlati példákon vizsgáltam, majd az így szerzett tapasztalatok alapján finomítottam a formalizmuson.

3. Új tudományos eredmények

Kutatásom tudományos eredményeit négy tézisben foglaltam össze, azokon belül pedig altézisekbe soroltam. Az elméleti eredményeket matematikai és mérnöki módszerekkel bizonyítottam, ipari alkalmazhatóságukat pedig esettanulmányokkal igazoltam. Az egyes téziseket az alábbiakban részletesen kifejtem, a tézisek szerkezetét és az altézisek egymásra épülését az 1. ábra mutatja.

I. Az első tézisben egy egységes formalizmust dolgozok ki, amely lehetővé teszi modelltranszformációk formális ellenőrzését. Módszert adok típusos gráfokon alapuló modellező nyelvek és modellek leírására. Bemutatok három fontos kategóriaelméleti kategóriát, amelyek lehetővé teszik, hogy a fenti formalizmust gráfújraíró rendszerek leírására is használjuk. A fentieken kívül módszert adok modelleken értelmezett attribútumkényszerek specifikálásra és vizsgálatára.

II. A második tézisben egy propozicionális logikán alapuló formális nyelvet dolgozok ki, melynek segítségével a modelltranszformációk elemzése során bebizonyítandó funkcionális tulajdonságok egy fontos halmazát lehet kifejezni. Meghatározom a nyelv szintaxisát és szemantikáját. Ahhoz,

hogy következtetéseket tudjunk megfogalmazni ezen a nyelven, következtetési szabályokat definiálok, bebizonyítom ezek helyességét, és elemzem a következtetőrendszer teljességét. A fentieken kívül a következtetési szabályok alapján kidolgozok egy következtetések elemzésére szolgáló algoritmust, és formálisan elemezem ennek tulajdonságait.

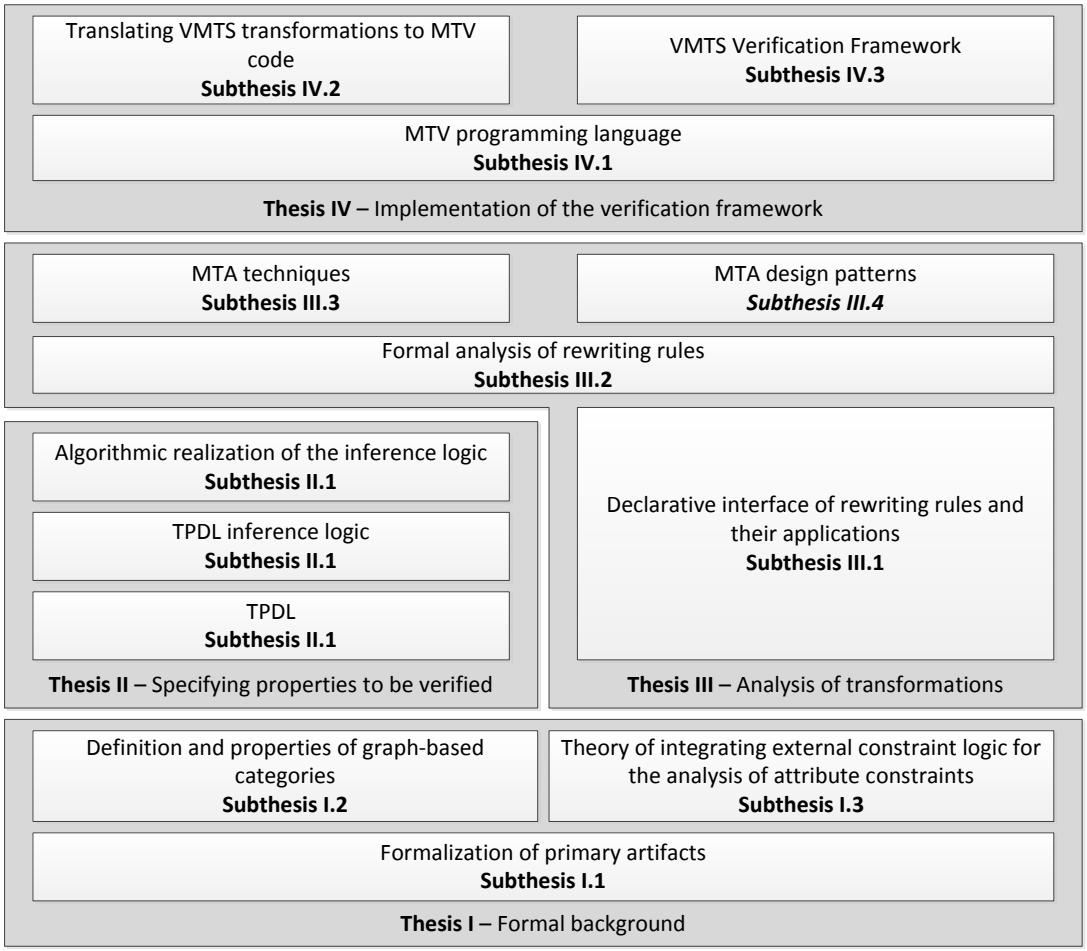
III. A harmadik tézis az előző kettőre épít. Módszert adok újraírási szabályok és azok modelleken történő alkalmazása deklaratív interfészének leírására. Ezek alapján formális módszereket dolgozok ki, amelyek segítségével különálló újraírási szabályok elemezhetők. A fentieken kívül bevezetem a modelltranszformációk implementálása és elemzése során használható MTA (Model Transformation Analysis) módszerek fogalmát. Az MTA technikák a modelltranszformációk elemzése során gyakran előkerülő módszerek formális leírásai. Ez a formális leírás lehetővé teszi, hogy különböző eszközökben automatizáljuk ezen technikák alkalmazását. Az MTA minták a hagyományos tervezési mintákhoz hasonló transzformáció részletek, amelyek elemzését előre elkészítjük. Így ezeket a mintákat nem csak a transzformációk implementálása, hanem azok verifikációja során is felhasználjuk.

IV. A negyedik tézis a fentiekben leírt elméleti alapra építve egy valós keretrendszer megvalósítását mutatja be. Kifejlesztettem egy szöveges programozási nyelvet (MTV), amellyel az előző tézisekben definiált formalizmusnak megfelelően modelltranszformációk deklaratív definícióját lehet leírni. Egy szoftvercsomagban megvalósítottam egy verifikációs keretrendszert, amely az előbbi programozási nyelven leírt transzformációkat értelmezi, és azokat a korábbi tézisekben leírt módszerekkel a gyakorlatban is elemzi. A fentieken kívül olyan algoritmusokat készítettem, melyek a VMTS keretrendszerben készített modelltranszformációkból automatikusan MTV programkódot generálnak. Ugyanakkor megmutatom, hogy a keretrendszer képességei nem korlátozódnak azokra a transzformációkra, melyek a VMTS keretrendszerben készültek, hanem az előbbi programozási nyelven keresztül könnyen együtt lehet működni más eszközökkel is. A fentieken kívül a módszereim gyakorlati használhatóságát egy esettanulmány bemutatásával igazolom.

I. Tézis - Modelltranszformációk vizsgálatának formális megalapozása

Az első tézis egy formális keretrendszert definiált, amely a típusos gráfok elméletén alapul [EEPT06]. Módszert adok modellező nyelvek, modellek, modellek sablonjai, és a közöttük lévő leképezések specifikálására. Informálisan, a legfontosabb definíciók a következők:

- Egy *metamodell-interfész* egy metamodell absztrakciója. Segítségével egy modellező nyelvet írunk le. Formálisan egy metamodell-interfész egy típusos gráf, ahol a csomópontokhoz és élekhez hozzárendeljük a lehetséges attribútumaik neveit.
- Egy metamodell-interfész *(példány)gráf*ja egy típusos gráf. Egy gráf elemeinek az egyes attribútumaihoz értékeket rendelhetünk, amit egy függvénnyel specifikálunk.
- Egy metamodell-interfész egy *(példány)modell*je egy példánygráfból és egy hozzá tartozó attribútumérték-hozzárendelésből áll.
- Az *absztrakt attribútumkényszerek* egy gráf bizonoys attribútumain definiált logikai függvények. Egy metamodell-interfész *(modell)sablon*ja egy gráfból és azon definiált attribútumkényszerekből áll.
- Egy *gyengén típusos morfizmus* egy hagyományos gráf homomorfizmus, amely egy metamodell-interfész két példánygráfja közötti leképezést definiál. A leképezés meghatározása során figyelembe kell venni, hogy a metamodellben az egyes csomópontok öröklődhetnek egymásból. Az injektív és teljes, gyengén típusos morfizmust *sablonmorfizmus*nak nevezem.
- Tételezzük fel, hogy olyan modelltranszformációval dolgozunk, melynek bemenete egyetlen modell. A végrehajtás során, a transzformáció ezt a modellt módosítja, azaz változtatja az attribútumait, töröl belőle elemeket, vagy újakat ad hozzá. A transzformáció kimenete tehát a módosított bemeneti modell. Ezért egy lehetséges bemeneti-kimeneti modellpár közötti relációt



1. ábra. Outline of the theses

két gráffal és egy közöttük definiált részleges morfizmussal lehet formalizálni. Ezek leírásához bevezetem a *relációgráf*, *relációmodell* és *relációsablon* fogalmát.

- A *relációmorfizmus* két relációgráf közötti leképezés.

Ahhoz, hogy a fenti formalizmus segítségével le tudjak írni gráfújraíró rendszereket, bevezetek három új kategóriaelméleti kategóriát és belátom ezek néhány fontos tulajdonságát.

- A $\mathbf{GraphsWM}_T$ kategória egy T típusos gráf példánygráfjaiból és azok közötti gyengén típusos morfizmusokból áll.
- A $\mathbf{GraphsPM}_{\mathfrak{M}}$ kategória egy $\mathfrak{M}$ metamodell-interfész példánygráfjaiból és azok közötti sablonmorfizmusokból áll.
- A $\mathbf{RelGraphsPM}_{\mathfrak{M}}$ kategória egy $\mathfrak{M}$ metamodell-interfész relációgráfjaiból és azok közötti relációmorfizmusokból áll.

Az attribútumkényszereket absztrakt módon definiálom. Meghatározok két relációt, melyek segítségével különböző kényszerhalmazok vizsgálhatók.

- Két kényszer *konfliktusban van*, ha egyszerre sosem lehetnek igazak.
- Egy kényszer levezethető egy másikból, ha a második kényszer teljesülése maga után vonja, hogy az elsőét.

A fenti két relációt a verifikációs keretrendszertől különálló kényszerelemző rendszerek segítségével elemezhetjük és láthatjuk be. Ahhoz, hogy ezeket használhassuk, definiálok azon függvények interfészét, melyekkel a relációk vizsgálhatók. Továbbá, bevezetek bizonyos korlátozásokat a kényszerekre vonatkozóan, amelyek lehetővé teszik, hogy gráfokhoz nem kapcsolódó általános

kényszerelemző rendszereket használjunk. Ez azért előnyös, mert a kényszer logikai programozás és a kényszerek kielégíthetőségének vizsgálata egy különálló kutatási terület, melynek eredményeit így integrálhatjuk a saját rendszerünkbe.

I.1 Altézis *Kidolgoztam egy formalizmust, és ennek segítségével leírtam metamodelleket, modelleket, modellek sablonjait, az ezek közötti relációkat és leképezéseket, továbbá attribútumkényszereket. Megmutattam, hogy minden modellhez megadható egy sablon, amely ekvivalens az eredeti modellel. Megmutattam, hogy egy sablonmorfizmuson keresztül egy forrásgráfon értelmezett attribútumérték-hozzárendelés leképezhető egy célgráfra, és bebizonyítottam, hogy az így leképzett attribútumértékhozzárendelés-függvény kompatibilis a célgráffal, azaz a leképzett attribútumérték-hozzárendelés egy, a célgráfon értelmezett érvényes attribútumérték-hozzárendelés. Megmutattam, hogy egy sablonmorfizmuson keresztül egy forrásgráfon értelmezett absztrakt attribútumkényszer leképezhető egy célgráfra, és bebizonyítottam, hogy a leképzés eredménye kompatibilis a célgráffal, azaz a leképzett attribútumkényszer egy, a célgráfon értelmezett valóban érvényes kényszer. Módszert adtam arra, hogy egy célgráfon értelmezett attribútumérték-hozzárendelést leképezzünk ellenkező irányban egy forrásgráfra egy sablonmorfizmuson keresztül, és bebizonyítottam, hogy a célgráf egy teljesen specifikált attribútumérték-hozzárendelését a forrásgráf egy teljesen specifikált attribútumérték-hozzárendelésévé képezzük le.*

I.2 Altézis *Bebizonyítottam, hogy* $\mathbf{GraphsWM_T}$ *kielégíti a kategória axiómáit, vagyis* $\mathbf{GraphsWM_T}$ *egy érvényes kategória a kategóriaelméletben. Megmutattam, hogy a sablonmorfizmusok zártak a kompozícióra, és ezáltal* $\mathbf{GraphsPM}_{\mathfrak{M}}$ *is egy érvényes kategória. Bebizonyítottam továbbá, hogy* $\mathbf{GraphsPM}_{\mathfrak{M}}$ *rendelkezik a következő tulajdonságokkal: (i) a pushout mindig létezik, azaz mindig megkonstruálható (ii) amennyiben két közös forrásobjektummal rendelkező sablonmorfizmus közül az egyik erősen típusos, akkor a belőlük képzett pushout konstrukcióban az ezzel átellenes morfizmus szintén erősen típusos, (iii) két közös célobjektummal rendelkező sablonmorfizmusnak a párfaktorizációja mindig megkonstruálható. Bebizonyítottam továbbá, hogy* $\mathbf{RelGraphsPM}_{\mathfrak{M}}$ *szintén eleget tesz a kategória definíciójának.*

I.3 Altézis *Formalizmust dolgoztam ki absztrakt attribútumkényszer-halmazok relációinak leírására. Elégséges feltételeket adtam arra, hogy ezen relációk alapján meghatározzuk sablonok és sablonmorfizmusok tulajdonságait. Leírtam azon függvények interfészét, melyek segítségével attribútumkényszer-halmazok relációi vizsgálhatók, és megmutattam, hogy ezzel a módszerrel külső kényszerellenőrző rendszerek – melyek képesek ezen relációk elemzésére – integrálhatók a verifikációs keretrendszerbe. Megmutattam, hogy ezen függvényinterfészek biztosítják a rendszer konzisztenciáját, azaz beláttam, hogy a függvények különböző implementációi nem mondhatnak ellent egymásnak.*

II. Tézis – Modelltranszformációk funkcionális tulajdonságainak kifejezése

Ebben a tézisben bemutatom a TPDL (Transformation Property Description Language) nyelvet, amely a propozicionális logikán alapul, és képes kifejezni azon funkcionális tulajdonságok egy fontos halmazát, amelyeket ellenőrizni szeretnénk a transzformációkon. Definiálom a TPDL nyelv atomi kifejezéseinek az interfészét. Egy atom egy logikai függvény, amelynek értelmezési tartománya egy adott metamodel-interfész relációmodelljeinek a halmaza. Ez a definíció lehetővé teszi, hogy a TPDL nyelv könnyen kiterjeszthető legyen újabb tulajdonságok leírásának lehetőségével. Bevezetek továbbá egy konkrét TPDL kifejezést, melyet *relációsablon-feltétel*nek nevezek.

A nyelv segítségével megfogalmazott állításokon következtetéseket szeretnénk végrehajtani, ezért következtetési (levezetési) szabályokat definiálok, amelyekről belátom, hogy helyesek. A levezetések segítségével implikációkat vizsgálunk, vagyis azt szeretnék bebizonyítani, hogy egy TPDL kifejezés egy másik kifejezés logikai következménye.

Ebben a tézisben elemezem annak a problémának az eldönthetőségét, hogy egy implikáció mindig kielégíthető-e. Bebizonyítom, hogy a probléma általánosságban eldönthetetlen, de bemutatom a problématér egy eldönthető részhalmazát.

A második tézis egy fontos eredménye a kalkulus egy algoritmikus megvalósításának bemutatása. Mivel a probléma általánosságban eldönthetetlen, fontos, hogy az algoritmus futásának egy értelmes korlátot szabjunk meg. Jogos elvárás, hogy ezt a korlátot úgy határozzuk meg, hogy az algoritmus képes legyen eldönteni a korábban vázolt eldönthető problémákat.

A II. Tézis a disszertáció 5. fejezetében található.
A II. Tézishez kapcsolódó publikációk: [5, 23, 22, 4, 2, 19, 15, 16, 9].

II.1 Altézis *Kidolgoztam egy propozicionális logikán alapuló formális nyelv interfészét (TPDL), mely képes kifejezni modelltranszformációk funkcionális tulajdonságainak egy halmazát. Megmutattam, hogy a fenti nyelv segítségével gyakorlati szempontból hasznos tulajdonságok fogalmazhatók meg, melyek a bemeneti és kimeneti modellek közötti összefüggéseket írják le. Megadtam továbbá egy konkrét atomi kifejezés (relációsablon-feltétel) szintaktikáját és szemantikáját relációsablonok segítségével. Bebizonyítottam, hogy izomorf relációsablonok segítségével felépített feltételek egymással ekvivalensek. Beláttam, hogy a TPDL nyelv szemantikája alapján a propozicionális logikai következtetési szabályok érvényesek a TPDL következtetőrendszerben.*

II.2 Altézis *Következtetési szabályokat dolgoztam ki a TPDL nyelven megfogalmazott implikációk vizsgálatára, és megmutattam, hogy ezek a szabályok helyesek. Elégséges feltételt adtam a következtetési szabályok alkalmazhatóságára oly módon, hogy ezek a feltételek algoritmikusan hatékonyan ellenőrizhetők legyenek. Megmutattam, hogy a TPDL következtetések helyessége általánosságban algoritmikusan eldönthetetlen feladat. Megmutattam, hogy létezik a következtetéseknek egy részhalmaza, mely algoritmikusan eldönthető.*

II.3 Altézis *Bevezettem a korlátfüggvények fogalmát, és megadtam egy érvényes korlátfüggvényt. Algoritmust dolgoztam ki TPDL kifejezésekből épített következtetések helyességének elemzésére. Megmutattam, hogy érvényes korlátfüggvény esetén az algoritmus mindig terminál. Megmutattam, hogy az általam definiált korlátfüggvény alkalmazása esetén az algoritmus mindig megoldást ad a II.3. altézisben meghatározott algoritmikusan eldönthető problémák részhalmazára.*

III. Tézis – Újraírási szabályok és részleges transzformációk formális elemzése

A harmadik tézis a következő kérdésekkel foglalkozik: (i) különálló újraírási szabályok verifikációja, (ii) összetettebb transzformációrészletek verifikációja, ill. (iii) ezen elemzések során belátott tulajdonságok propagálása egy modelltranszformáció vezérlő szerkezetén keresztül, vagyis azon tulajdonságok megállapítása, amik nem egy-egy transzformáció részlet után, hanem a teljes transzformáció végén teljesülnek.

Módszert adok újraírási szabályok és azok modelleken történő alkalmazása deklaratív interfészének a leírására. Ezen interfész alapján történik az újraírási szabályok elemzése, amelynek célja, hogy levezessünk olyan tulajdonságokat, melyek biztosan igazak lesznek a szabály alkalmazása után függetlenül a bemeneti modelltől.

Formalizálom modelltranszformációk irányított gráf alapú vezérlési szerkezetét, amely segítésével módszert adok az újraírási szabályok elemzése során belátott tulajdonságok propagálására (végigvezetésére) a teljes vezérlésfolyamgráfon. Így tudjuk levezetni azokat a tulajdonságokat, amelyek biztosan igazak lesznek a teljes transzformáció végén. Ezeket a tulajdonságokat az ún. *végső formula* határozza meg, amelynek leírásához a TPDL nyelvet használjuk. Amennyiben egy bizonyítandó kifejezés levezethető a végső formulából, akkor sikerült bebizonyítani, hogy az adott tulajdonság

minden lehetséges kimenet esetén teljesül. Amennyiben a bebizonyítandó kifejezés negáltját sikerül levezetni, akkor beláttuk, hogy nincs olyan lehetséges kimenet, amelyre igaz lenne a bebizonyítandó tulajdonság. Egyéb esetben a verifikációs rendszerünk nem tud semmit sem mondani az adott tulajdonság teljesüléséről.

A harmadik tézis egy másik fontos eredménye az MTA (Model Transformation Analysis) módszerek koncepciójának bemutatása.

- Modelltranszformációk formális vizsgálata során gyakran használunk ismétlődően előkerülő, formálisan leírható, intuitív módszereket, amelyeket nem tudnánk egy különálló transzformációrészlettel definiálni. Ezeket a módszereket összefoglaló néven *MTA technikáknak* nevezem. Az ilyen technikák formális dokumentálásának több előnye is van: (i) hasznos, ha az intuitívan használt módszereket mások is megismerik, és (ii) a formális lehetővé teszi, hogy a különböző verifikációs rendszerekben automatizáljuk ezek alkalmazását. Ebben a tézisben bevezetem az *Intakt Element* MTA technikát, amely segítségével meghatározhatók azok a szabályok egy transzformációban, melyek bizonyos sablonok példányait érintetlenül hagyják. E módszer automatizálása lehetővé teszi, hogy a verifikációt végző felhasználónak kevesebb szabályt kelljen manuálisan elemeznie. Egy másik bemutatott módszer a *Composite Rule* MTA technika, amely azt határozza meg, hogyan kell két szekvenciálisan végrehajtott újraírási szabályt egyetlen szabályként leírni. Az így előállított komponált szabály használatának több előnye is van. Például sokkal könnyebb egyetlen szabályt elemezni, mint több egymás utáni szabály sorozatát.
- Az *MTA minták* hasonlóak az objektum-orientált fejlesztés során megismert hagyományos tervezési mintákhoz [GHJV95]. Egy MTA minta pontosan leír egy olyan transzformáció részletet, mely egy gyakran előkerülő transzformációs feladatot old meg. A mintát alkalmazva beépíthetjük annak egy példányát egy konkrét transzformációba, ahol az adott probléma előkerül. A legfontosabb tulajdonsága ezeknek a transzformáció részleteknek, hogy önmagukban – a konkrét transzformációtól függetlenül – is elemezhetők, és ennek az elemezésnek az eredménye beépíthető annak a transzformációnak az elemzésébe, amelyben felhasználjuk az adott mintát. Más szóval, amikor egy mintát alkalmazunk, akkor az adott transzformáció részletre automatikusan teljesülnek bizonyos tulajdonságok. Ez a módszer lehetővé teszi komplex transzformációk automatizált elemzését. A harmadik tézisben bevezetek egy konkrét MTA mintát, amelyet *Traverser* mintának nevezek. Ez a minta megoldást javasol arra, hogyan tudjuk iteratívan feldolgozni egy adott sablon összes előfordulását a bementi modellben.

Az III. Tézis a disszertáció 7. és 8. fejezetében található.
Az III. Tézishez kapcsolódó publikációk: [5, 23, 21, 22, 4, 27, 1, 17, 11, 2, 3, 7, 8, 34, 33, 26, 13].

III.1 Altézis *Formalizmust dolgoztam ki újraírási szabályok deklaratív specifikálására. Megmutattam, hogy ezek alapján és a* $\mathbf{GraphsPM}_{\mathfrak{M}}$ *kategória felhasználásával leírható az elemi transzformációs lépések interfésze. Az újraírási szabályok interfésze alapján elégséges feltételt adtam arra, hogy egy szabály biztosan alkalmazható a bementi modellek egy halmazán. Az újraírási szabályok interfésze alapján elégséges feltételt adtam arra, hogy egy szabály biztosan nem alkalmazható a bemeneti modellek egy halmazán.*

III.2 Altézis *Formalizmust dolgoztam ki modelltranszformációk vezérlésfolyamgráfjának leírására. Elégséges feltételeket adtam egy újraírási szabály interfésze alapján olyan tulajdonságok levezethetőségére, melyek biztosan igazak lesznek a szabály sikeres alkalmazása után. Szintén elégséges feltételeket adtam egy újraírási szabály interfésze alapján olyan tulajdonságokat levezethetőségére, melyek biztosan igazak lesznek a szabály sikertelen alkalmazása után. Megmutattam, hogy a fenti elégséges feltételek algoritmikusan ellenőrizhetők.*

III.3 Altézis *Bevezettem az MTA (Model Transformation Analysis) technikák fogalmát, melyek olyan módszerek formális leírásai, melyeket gyakran használunk gráfújraírás alapú modelltranszformációk ellenőrzése során. Definiáltam az „Intact Element" MTA technikát. Elégséges feltételt adtam annak megállapítására, hogy egy újraírási szabály gyengén, vagy egyáltalán nem módosít sablonokat.*

III.4 Altézis *Formalizáltam a „Traverser" MTA mintát, amely a bemeneti modell elemei egy halmazának iteratív feldolgozását írja le. A mintával szakterület-független módon definiáltam egy résztranszformációt, és formális módszerek segítségével belátok róla tulajdonságokat. Elégséges feltételt adtam a „Traverser" mintában leírt transzformációrészlet terminálására. Zárt formulát adtam az adott részlet pontos lépésszámának meghatározására abban az esetben, ha az elégséges feltétel teljesül. Bebizonyítottam azokat a tulajdonságokat, melyek biztosan teljesülnek az adott részlet végrehajtása után. Megmutattam, hogy tetszőleges transzformációra, mely megvalósítja ezt a mintát, a fenti tulajdonságok valóban érvényesek lesznek.*

IV. Tézis – Új tudományos eredmények alkalmazása

Elkészítettem az első három tézisben bemutatott elméleti keretrendszer egy megvalósítását, melyet a kutatócsoportunk által fejlesztett többszintű modellező és modelltranszformációs keretrendszer (VMTS – Visual Modelig and Transformation System) részeként implementáltam. A szoftver elkészítése során az volt a célom, hogy az elméleti eredmények gyakorlati alkalmazhatóságát igazoljam.

Az MTV programozási nyelv

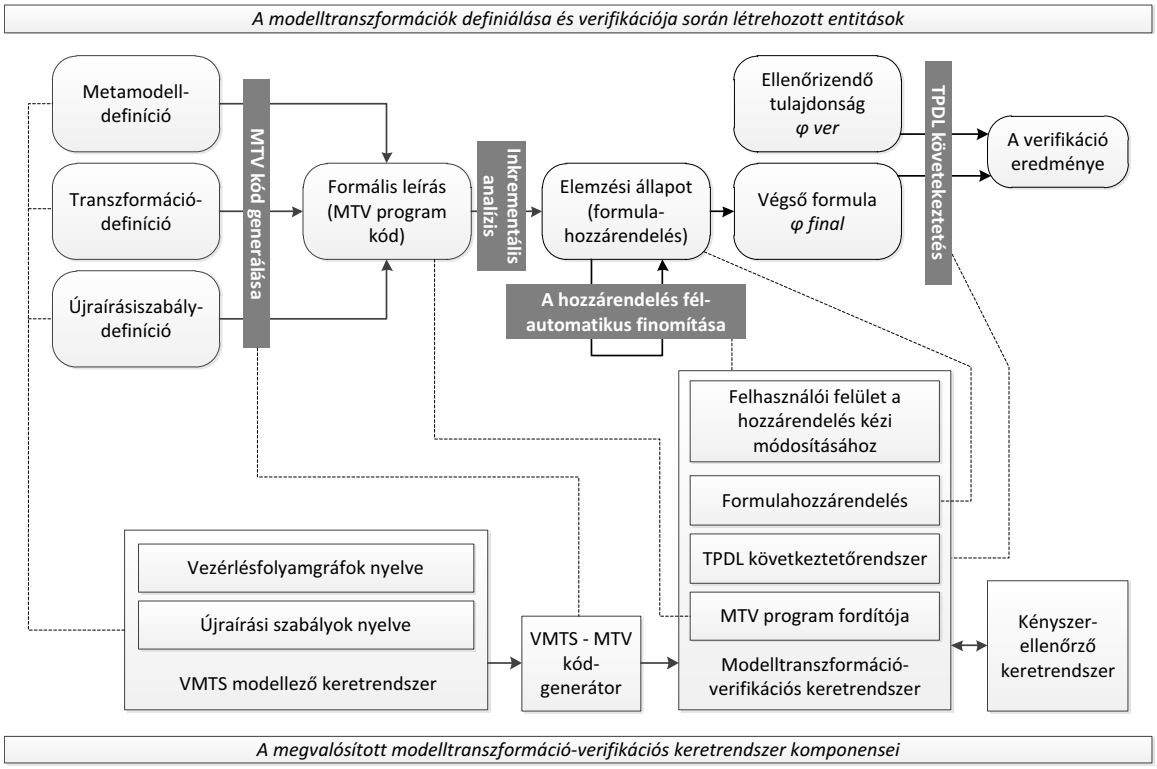
Kifejlesztettem egy szöveges programozási nyelvet (MTV – Model Transformation Verification), amelynek a segítségével le lehet írni metamodell-interfészeket, (reláció)gráfokat, (reláció)modelleket, (reláció)sablonokat, újraírásiszabály-interfészeket, vezérlésfolyamgráfokat és TPDL formulákat. Azaz, egy MTV program egy adott metamodell-interfész felett definiált modelltranszformációkat és azok verifikálandó tulajdonságait specifikálja az I. Tézisben bemutatott formalizmushoz igazodva. A megvalósított verifikációs keretrendszer beolvassa és értelmezi az MTV programokat, és elvégzi a bennük leírt modelltranszformációk ellenőrzését.

VMTS verifikációs keretrendszer

A továbbiakban a VMTS-ben megvalósított verifikációs keretrendszert mutatom be. Az 2. ábra felső részén látható egy modelltranszformáció verifikációjának tipikus folyamata. A modelltranszformáció definícióját először MTV programmá kell fordítani. A VMTS keretrendszerhez készítettem egy komponenst, mely ezt a feladatot automatizáltan elvégzi, azaz egy VMTS-ben definiált modelltranszformációból automatikusan képes vele ekvivalens MTV kódot generálni. Ez a komponens képes az újraírási szabályokhoz csatolt kényszerek és imperatív kód elemzésére is, amelyekből az I. Tézisben bemutatott absztrakt attribútumkényszereket származtat mind az LHS, mind az RHS gráfokhoz. Természetesen a fejlesztő ellenőrizheti és kiegészítheti az automatikusan generált kódot.

Mint említettem, a keretrendszer beolvassa és értelmezi az MTV nyelvű programokat, felismeri bennük a modelltranszformáció-leírásokat. Minden ilyen modelltranszformáció-leíráshoz automatikusan generál egy testre szabott felhasználó felületet, ahol az adott vezérlésfolyamgráfhoz tartozó hozzárendeléseket lehet áttekinteni és manuálisan módosítani. Ezen kívül, a felhasználó kiválaszthatja, hogy mely szabályokat elemzi a keretrendszer automatikusan, ezeknek az ellenőrzéseknek az eredményei is bővítik az aktuális hozzárendelést. Az automatizált vizsgálat és a kézi finomítás után a keretrendszer a végső formulát az aktuális hozzárendelés alapján számítja ki, majd ellenőrzi, hogy levezethető-e belőle a verifikálandó tulajdonság. Az 2. ábra alsó részén a keretrendszer felépítése és legfontosabb komponensei láthatók.

Megmutattam, hogy a verifikációs keretrendszer és a VMTS modelltranszformációs keretrendszer két különálló egység, melyek egymással csak MTV kóddal kommunikálnak, ezért nem kötelező, hogy az MTV kód, amelyet vizsgálunk, egy valós VMTS-beli modelltranszformációból származzon. Így a keretrendszer nem korlátozódik a csak VMTS-ben definiált modelltranszformációk ellenőrzésére, hanem bármely eszközbe – mely képes a modelltranszformációit ilyen programmá lefordítani – integrálható.



2. ábra. A modelltranszformáció-verifikációs keretrendszer felépítése

Eszköztámogatás az MTA technikák és minták használatához

A VMTS keretrendszer funkcionalitása kiegészíthető különálló komponensek, ún. *addon*ok fejlesztésével. Ezek a komponensek hozzáférnek a keretrendszer által nyújtott szolgáltatásokhoz egy jól meghatározott API-n keresztül. Így elérhetjük a VMTS modelleket, a memóriába már beolvasott MTV programokat, és a VMTS felhasználói felületét is módosíthatjuk.

A harmadik tézisben bemutatott mindkét MTA technikához egy-egy addont fejlesztettem, amelyek lehetővé teszik az adott technikák automatizált alkalmazását. Szinten külön komponens készült a *Traverser* minta használatához. Ennek segítségével a felhasználónak lehetősége van megjelölni bizonyos szabályokat, ezzel jelezve, hogy ezek a szabályok a *Traverser* minta egy példányát írják le. Az addon segít leellenőrizni, hogy az adott szabályok valóban teljesítik-e a minta leírásában meghatározott követelményeket. Miután ez megtörtént, a *Traverser* mintára belátott tulajdonságokat automatikusan tudjuk használni az adott transzformáció elemzésekor.

A IV. Tézis a disszertáció 9. fejezetében található.
A IV. Tézishez kapcsolódó publikációk: [14, 24, 25, 27, 32, 20, 28, 30, 31, 11, 35, 12, 10, 6, 29].

IV.1 Altézis *Kifejlesztettem egy eszközfüggetlen szöveges programozási nyelvet (MTV – Model*

Transformation Verificiation), melynek a segítségével az I. Tézisben bemutatott formalizmuson alapuló modelltranszformációkat lehet leírni. Megmutattam, hogy mérnöki verifikációs problémák leírhatók MTV programokként. Megmutattam, hogy az MTV nyelv használatával az általam megvalósított keretrendszerrel elemezhető transzformációk nem korlátozódnak a VMTS-ben fejlesztett transzformációkra.

IV.2 Altézis *Algoritmust dolgoztam ki VMTS-ben definiált modelltranszformációk automatikus feldolgozására, melynek során az általam készített eszköz MTV kódot generál, melyet a keretrendszer azonnal képes elemezni. Megmutattam, hogy az előző tézisekben bemutatott elméleti módszer alapján implementálható egy verifikációs keretrendszer, amely képes modelltranszformációk formális elemzésére.*

IV.3 Altézis *Megmutattam, hogy az elméleti keretrendszer megvalósítható egy komponensalapú és kiterjeszthető szoftvercsomagként (VMTS-VF – VMTS Verification Framework), amely képes formálisan elemezni tetszőleges MTV kódot. Megmutattam, hogy a keretrendszer képes MTA mintákat felhasználni az elemzések során.*

A VMTS keretrendszer nyilvánosan elérhető és letölthető a `http://vmts.aut.bme.hu` oldalról.

4. Kapcsolódó publikációk

Folyóiratcikkek

[1] **Asztalos Márk**, Madari István, Lengyel László. Towards formal analysis of multi-paradigm model transformations. *Simulation*, 86(7):429–452, 2010. **I.F.: 0.611**

[2] **Asztalos Márk**, Ekler Péter, Lengyel László, Levendovszky Tihamér, Madari István, Vajk Tamás. Automated formal verification of graph rewriting-based model transformations. *Buletinul Stiintific Al Universitatii Politehnica Din Timisoara-Seria Automatica Si Calculatoare*, 55(69):175–184, 2010.

[3] **Asztalos Márk**, Ekler Péter, Lengyel László, Levendovszky Tihamér, Mészáros Tamás, Mezei Gergely. Automated verification by declarative description of graph rewriting-based model transformations. *Electronic Communications Of The EASST*, 42:12, 2010.

[4] **Asztalos Márk**, Lengyel László, Levendovszky Tihamér. Toward automated verification of model transformations: A case study of analysis of refactoring business process models. *Electronic Communications of the EASST*, 21, 2009.

[5] **Asztalos Márk**, Lengyel László, Levendovszky Tihamér. A formal framework for automated verification of model transformations. *Software Testing Verification & Reliability*, 2012. (elbírálás alatt).

[6] **Asztalos Márk**, Madari István, Mészáros Tamás, Vajk Tamás, Mezei Gergely. Szakterület-specifikus modellezés. *Híradástechnika*, LXV(5-6):25–30, 2010.

[7] **Asztalos Márk**, Eugene Syriani, Manuel Wimmer, and Marouane Kessentini. Simplifying model transformation chains by rule composition. *Lecture Notes in Computer Science*, 6627:293–307, 2011. Proceedings of the 2010 international conference on Models in software engineering.

[8] **Asztalos Márk**, Eugene Syriani, Manuel Wimmer, and Marouane Kessentini. Towards transformation rule composition. *Electronic Communications of the EASST*, 42:13, October 2011. 4th International Workshop on Multi-Paradigm Modeling - MPM'10.

[9] Ekler Péter, **Asztalos Márk**. Performance analysis of maintaining mobile-based social network models. *WSEAS Transactions on Computers*, 9(12):1391–1400, dec 2010.

[10] Mészáros Tamás, **Asztalos Márk**, Mezei Gergely. Overlapped execution of rewriting rules in graph transformation systems. *Buletinul Stiintific Al Universitatii Politehnica Din Timisoara-Seria Automatica Si Calculatoare*, 55(3):143–150, 2010.

[11] Mészáros Tamás, Mezei Gergely, Levendovszky Tihamér, **Asztalos Márk**. Manual and automated performance optimization of model transformation systems. *International Journal of Software Tools for Technology Transfer*, 12(3-4):231–243, July 2010.

[12] Vajk Tamás, **Asztalos Márk**, Mezei Gergely. Formal analysis of incremental code generation in ocl compilers. *Buletinul Stiintific Al Universitatii Politehnica Din Timisoara-Seria Automatica Si Calculatoare*, 55(69):117–122, 2010.

[13] Varró Dániel, **Asztalos Márk**, Bisztray Dénes, Artur Boronat, Duc-Hanh Dang, Rubino Geiss, Joel Greenyer, Pieter Van Gorp, Ole Kniemeyer, Anantha Narayanan, Edgars Rencis, Erhard Weinell. Transformation of UML Models to CSP: A Case Study for Graph Transformation Tools. *Applications of Graph Transformations with Industrial Relevance*, volume 5088 of *Lecture Notes in Computer Science*, pp 540–565. Springer Berlin / Heidelberg, 2008.

Nemzetközi konferencia-kiadványban megjelent, idegen nyelvű előadás

[14] Angyal László, **Asztalos Márk**, Lengyel László, Levendovszky Tihamér, Madari István, Mezei Gergely, Mészaros Tamás, Siroki László, Vajk Tamás. Towards a fast, efficient and customizable domain-specific modeling framework. *Proceedings of the IASTED International Conference*, pp 11–16, Innsbruck, Ausztria, 2009. ACTA Press.

[15] **Asztalos Márk**. Comparison of termination criteria for graph transformation systems. *Proceedings of the Automation and Applied Computer Science Workshop: AACS'07*, Budapest, Magyarország, 2007.

[16] **Asztalos Márk**. Offline analysis of the properties of transformation UML to CSP. *Proceedings of the Automation and Applied Computer Science Workshop: AACS'08*, pp 259–271, Budapest, Magyarország, 2008. ISBN: 978-963-420-955-3.

[17] **Asztalos Márk**, Ekler Péter, Lengyel László, Levendovszky Tihamér. Verification of model transformations to refactoring mobile social networks. *Electronic Communications of the EASST*, 32:12, 2010.

[18] **Asztalos Márk**, Ekler Péter, Lengyel László, Levendovszky Tihamér, Mészáros Tamás. Formalizing models with abstract attribute constraints. *Third International Workshop on Graph Computation Models*, pp 65–80, Enschede, Hollandia, 2010.

[19] **Asztalos Márk**, Ekler Péter, Lengyel László, Levendovszky Tihamér, Vajk Tamás. MCDL: A language for specifying graph conditions with attribute constraints. *Proceedings of Workshop on Model-Driven Engineering, Verification, and Validation*, pp 43–48, Oslo, Norvégia, 2010.

[20] **Asztalos Márk**, Lengyel László. A metamodel-based matching algorithm for model transformations. *6th IEEE International Conference on Computational Cybernetics (ICCC 2008)*, pp 151–155, Stara Lesna, Szlovákia, November 2008.

[21] **Asztalos Márk**, Lengyel László, Levendovszky Tihamér. A formalism for automated verification of model transformations. *International Symposium of Hungarian Researchers on Computational Intelligence and Informatics (CINTI)*, pp 377–390, Budapest, Magyarország, 2009.

[22] **Asztalos Márk**, Lengyel László, Levendovszky Tihamér. A formalism for describing modeling transformations for verification. *MoDeVVa'09 : Proceedings of the 6th International Workshop on Model-Driven Engineering, Verification and Validation*, pp 1–10, New York, NY, USA, 2009. ACM.

[23] **Asztalos Márk**, Lengyel László, Levendovszky Tihamér. Towards automated, formal verification of model transformations. *Third International Conference on Software Testing, Verification and Validation*, pp 15 – 24, Párizs, Franciaország, 2010. IEEE.

[24] **Asztalos Márk**, Lengyel László, Levendovszky Tihamér, Charaf Hassan. Graph transformation contest - UML to CSP transformation. *Applications of Graph Transformation 2007 (AGTIVE) - Graph Transformation Tool Contest*, page 5, Kassel, Németország, 2007.

[25] **Asztalos Márk**, Lengyel László, Levendovszky Tihamér, Charaf Hassan. Termination analysis of the transformation UML to CSP. *Proceedings of Computational Intelligence and Informatics 8th International Symposium of Hungarian Researchers*, pp 611–622, Budapest, Magyarország, 2007.

[26] **Asztalos Márk**, Madari István. VTL: an improved model transformation language. *Proceedings Automation and Applied Computer Science Workshop 2009: AACS'09*, pp 185–195, Budapest, Magyarország, 2009. ISBN: 978-963-420-978-2.

[27] **Asztalos Márk**, Madari István, Vajk Tamás, Lengyel László, Levendovszky Tihamér. Formal verification of model transformations : An automated framework. *2010 IEEE International Joint Conference on Computational Cybernetics and Technical Informatics (ICCC-CONTI)*, pp 493–498, Temesvár, Románia, 2010.

[28] **Asztalos Márk**, Mészáros Tamás, Lengyel László. Generating executable BPEL code from BPMN models. *Grabats 2009 5th international workshop on graph-based tools – graph transformation tool contest*, page 5, Zürich, Svájc, 2009.

[29] **Asztalos Márk**, Vajk Tamás. Automated offline verification of graph rewriting-based model transformations. In *Proceedings of the Automation and Applied Computer Science Workshop 2010 (AACS'10)*, Budapest, Magyarország, 2010. Department of Automation and Applied Informatics, ISBN: 978-963-313-004-9.

[30] Lengyel László, Madari István, **Asztalos Márk**, Levendovszky Tihamér. Validating query/view/transformation relations. *Proceedings of Workshop on Model-Driven Engineering, Verification, and Validation*, pp 7–12, Oslo, Norvégia, 2010.

[31] Levendovszky Tihamér, Mészáros Tamás, Ekler Péter, **Asztalos Márk**. DSML-aided developement for mobile P2P systems. In Matti Rossi, Jonathan Sprinkle, Jeff Gray, and Juha-Pekka Tolvanen, editors, *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM'09)*, pp 51–56, Orlando, USA, 2009. HSE Print.

[32] Madari István, **Asztalos Márk**, Mészáros Tamás, Lengyel László, Charaf Hassan. Implementing QVT in a domain-specific modeling framework. *5th International Conference on Software and Data Technologies (ICSOFT)*, pp 304–307, Athén, Görögország, 2010.

[33] Mészáros Tamás, **Asztalos Márk**, Mezei Gergely. An overlapped execution technique for graph rewriting rules. *IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics (ICCC-CONTI 2010)*, pp 281–286, 2010.

[34] Mészáros Tamás, **Asztalos Márk**, Mezei Gergely, Charaf Hassan. Performance optimization of exhaustive rules in graph rewriting systems. *ICSOFT 2010 - Proceedings of the 5th International Conference on Software and Data Technologies*, volume 2, pp 292–295, 2010.

[35] Vajk Tamás , **Asztalos Márk**. Incremental code generation in Object Constraint Language compilers. *Proceedings of the Automation and Applied Computer Science Workshop 2010 (AACS'10)*, pp 71–84, Budapest, Magyarország, 2010. ISBN: 978-963-313-004-9.

[36] Vajk Tamás, Dávid Zoltán, **Asztalos Márk**, Mezei Gergely, Levendovszky Tihamér. Runtime model validation with parallel object constraint language. *Proceedings of the 8th International Workshop on Model-Driven Engineering, Verification and Validation*, pp 1–7, Wellington, Új-Zéland, 2011. ACM.

5. Idegen hivatkozások

[1] publikációra hivatkozik:

Eugene Syriani, Jeff Gray. Challenges for Addressing Quality Factors in Model Transformation: In 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), pp. 929-937.

[11] publikációra hivatkozik:

Amir Hossein Ghamarian, Maarten de Mol, Arend Rensink, Eduardo Zambon and Maria Zimakova. Modelling and analysis using GROOVE, International Journal on Software Tools for Technology Transfer (STTT) 14(1), pp 15-40, 2012.

Krause, C., Maraikar, Z., Lazovik, A., Arbab, F. Modeling dynamic reconfigurations in Reo using high-level replacement systems. Science of Computer Programming, 76 (1), pp. 23-36, 2011.

Tisi, M., Martínez, S., Jouault, F., Cabot, J. Lazy execution of model-to-model transformations. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6981 LNCS, pp 32-46, 2011.

Louis M. Rose, Markus Herrmannsdoerfer, Steffen Mazanek, Pieter Van Gorp, Sebastian Buchwald, Tassilo Horn, Elina Kalnina, Andreas Koch, Kevin Lano, Bernhard Schätz, et al. Graph and model transformation tools for model migration, Software and Systems Modeling, 2012.

Enrico Biermann, Claudia Ermel, Leen Lambers, Ulrike Prange, Olga Runge and Gabriele Taentzer. Introduction to AGG and EMF Tiger by modeling a Conference Scheduling System, International Journal on Software Tools for Technology Transfer, 12(3-4), pp 245-261 július, 2010.

[13] publikációra hivatkozik:

Federico Banti, Rosario Pugliese Francesco Tiezzi. An accessible verification environment for UML models of services, Journal of Symbolic Computation, Volume 46, Issue 2, February 2011, pp 119-149, ISSN 0747-7171

James Sharp, Helen Treharne. CASE Technical Report: Automated Transformations from VHDL to CSP, 2011.

Kawtar Benghazi, Luis Garrido, Manuel Noguera, Maria V. Hurtado, Lawrence Chung. Extending and Formalizing UML 2.0 Activity Diagrams for the specification of time-constrained business processes. Fourth International Conference on Research Challenges in Information Science (RCIS), pp.93-100, 2010.

Dang, D.-H., Gogolla, M. Precise model-driven transformations based on graphs and metamodels (2009) SEFM 2009 - 7th IEEE International Conference on Software Engineering and Formal Methods, art. no. 5368074, pp. 307-316.

[22] publikációra hivatkozik:

Lasalle Jonathan, Bouquet Fabrice, Legeard Bruno, Peureux Fabien. SysML to UML model transformation for test generation purpose, ACM SIGSOFT Software Engineering Notes, Volume 36 Issue 1, január, 2011. pp 1-8, 2011

Horacio López, Fernando Varesi, Marcelo Vinolo, Daniel Calegari, Carlos Luna. Estado del arte de verificación de transformación de modelos Reporte Técnico RT 10-07 Facultad de Ingeniería, Universidad de la República Montevideo, Uruguay, 2010.

Calegari Daniel, Szasz Nora. Verificacioi de transformaciones de modelos. Una Revision del Estado del Arte (Version Extendida), Reporte Tecnico RT 12-05, Instituto de Computacion - Facultad de Ingenieria, Universidad de la Republica Montevideo, Uruguay, ISSN 0797-6410, 2012.

[23] publikációra hivatkozik:

Martin Strecker. Locality in Reasoning about Graph Transformations, International Symposium on Applications of Graph Transformation With Industrial Relevance (AGTIVE), Budapest, Magyarország 2011.

Márcio Filipe Caetano Mateus. Measuring Data Transfer in Heterogeneous IoT Environments, Dissertacao para obtencao do Grau de Mestre em Engenharia Electrotécnica e de Computadores, Universidade Nova De Lisboa, 2012.

Moussa Amrani, Levi Lucio, Gehan Selim, Benoit Combemale, Jurgen Dingel, Hans Vangheluwe, Yves Le Traon, James R. Cordy. A Tridimensional Approach for Studying the Formal Verification of Model Transformations, In Proceeding of the IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), pp 921-928 IEEE Computer Society, ISBN: 978-0-7695-4670-4, 2012.

Hanh Nhi Tran, Christian Percebois. Towards a Rule-Level Verification Framework for Property-Preserving Graph Transformations ICST (International Conference on Software Testing, Verification and Validation) Workshop on Verification and validation Of model Transformations (VOLT), pp 946-953., Montreal, Canada (2012)

Hanna Schölzel, Hartmut Ehrig, Frank Hermann, Christoph Brandt. Propagation of Constraints along Model Transformations Based on Triple Graph Grammars, Electronic Communications of the EASST, 41, 2011.

Léveque, T., Carlson, J., Sentilles, S., Borde, E. Flexible semantic-preserving flattening of hierarchical component models (2011) Proceedings - 37th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2011, art. no. 6068319, pp. 31-38.

Fabian Büttner, Jordi Cabot, Martin Gogolla. On validation of ATL transformation rules by transformation models, In Proceedings of the 8th International Workshop on Model-Driven Engineering, Verification and Validation, article no. 9., 2011.

Marko Jurisic. Transition between process models (BPMN) and service models (WS-BPEL and other standards): A systematic review, Journal of Information and Organizational Sciences, 35:(2), 163-171, 2011.

Lúcio L, Barroca B, Amaral V. A technique for automatic validation of model transformations, ure Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6394 LNCS:(PART 1), pp 136-150, 2010.

[25] publikációra hivatkozik:

Louis M. Rose, Markus Herrmannsdoerfer, Steffen Mazanek, Pieter Van Gorp, Sebastian Buchwald, Tassilo Horn, Elina Kalnina, Andreas Koch, Kevin Lano and Bernhard Schätz, Manuel Wimmer. Graph and model transformation tools for model migration: Graph and model transformation tools for model migration Software and Systems Modeling, Springer Berlin, 1-37, 2012 (megjelenés alatt)

[27] publikációra hivatkozik:

Calegari Daniel, Szasz Nora. Verificaciói de transformaciones de modelos. Una Revisión del Estado del Arte (Versión Extendida), Reporte Tecnico RT 12-05, Instituto de Computacion - Facultad de Ingenieria, Universidad de la Republica Montevideo, Uruguay, ISSN 0797-6410, 2012.

[28] publikációra hivatkozik:

Espinosa Enrique D, Frausto Juan, Rivera Ernesto J. Markov Decision Processes for Optimizing Human Workflows, Service Science 2: (4) pp 245-269., 2010.

[30] publikációra hivatkozik:

Pejman Salehi. A Model Based Framework for Service Availability Management, PhD thesis, Concordia University, Montreal, Kanada, 2012.

6. Hivatkozások

[ABK07] K. Anastasakis, B. Bordbar, and J. M. Küster. Analysis of Model Transformations via Alloy. In *Model Driven Engineering, Verification, and Validation: Integrating Verification and Validation in MDE (MODEVVA07)*, pages 47–56, October 2007.

[Agr03] Aditya Agrawal. Graph Rewriting And Transformation (GReAT): A Solution For The Model Integrated Computing (MIC) Bottleneck. *International Conference on Automated Software Engineering*, 0:364, 2003.

[ALS+12] Moussa Amrani, Levi Lucio, Gehan Selim, Benoit Combemale, Jurgen Dingel, Hans Vangheluwe, Yves Le Traon, and James R. Cordy. A tridimensional approach for studying the formal verification of model transformations. In *Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, ICST '12, pages 921–928, Washington, DC, USA, 2012. IEEE Computer Society.

[BA93] M. Ben-Ari. *Mathematical logic for computer science*. Prentice-Hall International series in computer science. Prentice-Hall International, Hempel Hempstead England, 1993.

[BBG+06] Jean Bézivin, Fabian Büttner, Martin Gogolla, Frederic Jouault, Ivan Kurtev, and Arne Lindow. Model transformations? transformation models! In Oscar Nierstrasz, Jon Whittle, David Harel, and Gianna Reggio, editors, *Model Driven Engineering Languages and Systems*, volume 4199 of *Lecture Notes in Computer Science*, pages 440–453. Springer Berlin / Heidelberg, 2006.

[BFG96] Dorothea Blostein, Hoda Fahmy, and Ann Grbavec. Issues in the practical use of graph rewriting. In *5th Workshop on Graph Grammars and Their Application To Computer Science, Lecture Notes in Computer Science*, pages 38–55. Springer, 1996.

[BH07] Dénes Bisztray and Reiko Heckel. Rule-level verification of business process transformations using CSP. *ECEASST*, 6, 2007.

[BW90] Michael Barr and Charles Wells. *Category Theory for Computing Science*. Prentice Hall, 1990.

[CHM+02] György Csertán, Gábor Huszerl, István Majzik, Zsigmond Pap, András Pataricza, and Dániel Varró. VIATRA: Visual Automated Transformations for Formal Verification and validation of UML models. In Julian Richardson, Wolfgang Emmerich, and Dave Wile, editors, *Proc. ASE 2002: 17th IEEE International Conference on Automated Software Engineering*, pages 267–270, Edinburgh, UK, September 23–27 2002. IEEE Press.

[dLV02] Juan de Lara and Hans Vangheluwe. AToM3: A Tool for Multi-formalism and Metamodelling. In *FASE '02: Proceedings of the 5th International Conference on Fundamental Approaches to Software Engineering*, pages 174–188, London, UK, 2002. Springer-Verlag.

[dLV10] Juan de Lara and Hans Vangheluwe. Automating the transformation-based analysis of visual languages. *Formal Aspects of Computing*, 22:297–326, 2010. 10.1007/s00165-009-0114-y.

[EEdL+05] H. Ehrig, K. Ehrig, J. de Lara, G. Taentzer, D. Varró, and Sz. Varró-Gyapay. Termination criteria for model transformation. *FASE*, pages 49–63, 2005.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*, volume XIV of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer, 2006.

[FS99] Martin Fowler and Kendall Scott. *UML Distilled: A Brief Guide to the Standard Object Modeling Language (2nd Edition)*. Addison-Wesley Professional, August 1999.

[GH06] Holger Giese and Stefan Henkler. A survey of approaches for the visual model-driven development of next generation software-intensive systems. *J. Vis. Lang. Comput.*, 17(6):528–550, December 2006.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison-Wesley, Boston, MA, 1995.

[GHN10] Holger Giese, Stephan Hildebrandt, and Stefan Neumann. Model synchronization at work: Keeping sysml and autosar models consistent. In Gregor Engels, Claus Lewerentz, Wilhelm Schäfer, Andy Schürr, and Bernhard Westfechtel, editors, *Graph Transformations and Model-Driven Engineering*, volume 5765 of *Lecture Notes in Computer Science*, pages 555–579. Springer Berlin Heidelberg, 2010.

[Hec06] Reiko Heckel. Graph transformation in a nutshell. *Electron. Notes Theor. Comput. Sci.*, 148(1):187–198, February 2006.

[HR04] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, New York, NY, USA, 2004.

[KN08] Gabor Karsai and Anantha Narayanan. Towards Verification of Model Transformations Via Goal-Directed Certification. pages 67–83. 2008.

[KNNZ99] Thomas Klein, Ulrich A. Nickel, Jörg Niere, and Albert Zündorf. From UML to Java And Back Again. Technical Report tr-ri-00-216, University of Paderborn, Paderborn, Germany, September 1999.

[KT08] Steven Kelly and Juha-Pekka Tolvanen. *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley-IEEE Computer Society Pr, March 2008.

[KWB03] Anneke G. Kleppe, Jos Warmer, and Wim Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[LBA10] Levi Lucio, Bruno Barroca, and Vasco Amaral. A technique for automatic validation of model transformations. In *MoDELS (1)*, pages 136–150, 2010.

[Len06] László Lengyel. *Online Validation of Visual Model Transformations.* PhD thesis, Budapest University of Technology and Economics, Department of Automation and Applied Informatics, 2006.

[LPE06] Tihamér Levendovszky, Ulrike Prange, and Hartmut Ehrig. A Termination Criteria for DPO Transformations with Injective Matches. In *Graph Transformation for Verification and Concurrency*, pages 102–116, Bonn, Germany, August 2006.

[MCG05] Tom Mens, Krzysztof Czarnecki, and Pieter Van Gorp. A taxonomy of model transformation. In *Proc. Dagstuhl Seminar on "Language Engineering for Model-Driven Software Development". Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl.* Electronic, 2005.

[Obj07] Object Management Group (OMG). *OMG Unified Modeling Language (OMG UML) Superstructure Specification, V2.1.2*, November 2007.

[Pen09] Karl-Heinz Pennemann. *Development of Correct Graph Transformation Systems.* PhD thesis, Department of Computing Science, University of Oldenburg, Oldenburg, 2009.

[Pie91] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists.* MIT Press, 1991.

[Plu98] Detlef Plump. Termination of graph rewriting is undecidable. *Fundam. Inf.*, 33(2):201–209, 1998.

[Roz97a] G. Rozenberg. Handbook on graph grammars and computing by graph transformation, foundations, vol.1. *World Scientific,*, 1997.

[Roz97b] Grzegorz Rozenberg. *Handbook on Graph Grammars and Computing by Graph Transformation: Foundations*, volume 1. World Scientific, Singapore, 1997.

[SK97] J. Sztipanovits and G. Karsai. Model-integrated computing. *Computer*, 30(4):110 –111, April 1997.

[SK03] Shane Sendall and Wojtek Kozaczynski. Model transformation: The heart and soul of model-driven software development. *IEEE Softw.*, 20(5):42–45, September 2003.

[SV09] Eugene Syriani and Hans Vangheluwe. Matters of model transformation. Technical Report SOCS-TR-2009.2, 2009.

[SVC06] Thomas Stahl, Markus Voelter, and Krzysztof Czarnecki. *Model-Driven Software Development: Technology, Engineering, Management.* John Wiley & Sons, 2006.

[Tae04] Gabrielle Taentzer. AGG: A Graph Transformation Environment for Modeling and Validation of Software. In *Application of Graph Transformations with Industrial Relevance (AGTIVE 2004)*, volume 3062 of *LNCS 3062*, pages 446–453. Springer, 2004.

[VMT10] Visual Modeling and Transformation System. `http://vmts.aut.bme.hu`, 2010.

[WL06] Alan Wassyng and Mark Lawford. Software tools for safety-critical software development. *Int. J. Softw. Tools Technol. Transf.*, 8(4):337–354, August 2006.

[WML10] Alan Wassyng, Tom Maibaum, and Mark Lawford. On software certification: we need product-focused approaches. In *Proceedings of the 15th Monterey conference on Foundations of Computer Software: future Trends and Techniques for Development*, Monterey'08, pages 250–274, Berlin, Heidelberg, 2010. Springer-Verlag.