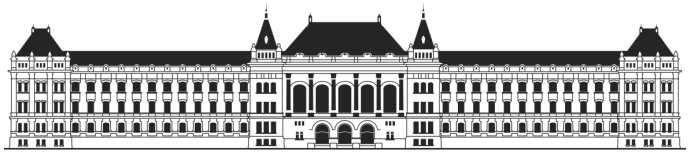

MŰEGYETEM 1782
Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Analízis technikák modell-lekérdezések és transzformációk elemzésére

PhD tézisfüzet

Ujhelyi Zoltán
Okl. Mérnök-informatikus

Témavezető:
Dr. Dániel Varró, DSc
egyetemi tanár

Budapest, 2016

1. A kutatás előzményei

Modell-alapú rendszertervezés A szoftverrendszerek növekvő komplexitásának kezelésére különböző fejlesztési módszertanokat használnak, mint például a *modell-alapú rendszertervezést* (model-driven engineering, MDE) [BG05; WHR14], amely magas szintű modellek tervezése és analízise segítségével teszi lehetővé a rendszer különböző tulajdonságainak vizsgálatát akár többféle absztrakciós szinten is. A modellezés megfelelő alkalmazása érthetőbbé teszi a rendszert, ezáltal gyorsítja a fejlesztést és például a hibák számának csökkentésével [HWR14] javítja a minőséget. Ezek a tulajdonságok különösen előnyösek kritikus rendszerek fejlesztésekor, ahol rendszerhibák emberi életet vagy egészséget veszélyeztethetnek, esetleg számottevő anyagi károkat okozhatnak [Bro+14].

A minőség és a produktivitás párhuzamos javulása nagyrészt a rendszermodellek korai ellenőrzésének, valamint a különböző tervezési modellek és dokumentumok, illetve forráskód automatikus generálásának köszönhető. A modellek ellenőrzése már azelőtt lehetővé teszi a tervezési hibák kimutatását, mielőtt a hagyományos tesztelési technikákat alkalmazni lehetne. Az automatikus forráskódgenerálás pedig mind a tervezési, mind az ellenőrzési időt csökkenti. Ráadásul a magas szintű modellek lehetővé teszik tesztesetek, konfigurációs fájlok és dokumentáció automatikus előállítását is.

Többféle szakterületen építenek a modellezésre, mint elsődleges tervezési eszközre [Béz05], különféle modellezési nyelvek felhasználásával. Ezen nyelvek alkalmasak szoftver- és hardverarchitektúrák leírására (pl. UML [Obj11c], SysML [Obj10] vagy AADL [SAE09]), a komponensek külső vagy belső viselkedések specifikációjára (pl. UML állapottérképek [Obj11c], BMPN [Obj11a] vagy BPEL [OAS07]), komponensek megbízhatóságának és teljesítményének vizsgálatára (pl. MARTE [Obj11b]) vagy formális ellenőrzésekre, mint elosztott rendszerek modellezésénél a Petri hálókat [Mur89] vagy a modellellenőrzés során használt Kripke struktúrákat [Cla08]. Mindezeken felül, gyakran használnak iparág-specifikus platformleíró nyelveket, mint amilyen az AUTOSAR [AUT12] az autóiparban vagy az ARINC-653 [Pri08] a repülőiparban.

Modell-lekérdezések és -transzformációk A *modell-lekérdezések* sokféle modellezési feladat, beleértve jólformáltsági kényszerek ellenőrzésének vagy modellmetrikák kiszámításának a részproblémáját alkotják [Aut+16; Bar+15]. Ezen lekérdezések eredményei azok a modellelemek, amelyekre teljesül a lekérdezés által megfogalmazott tulajdonság, amely megkövetelheti több elemi tulajdonság együttes teljesülését. Ezenfelül a modell-lekérdezések használata megkönnyítheti a magas szintű nézeti modellek származtatását.

A *modelltranszformációk (MT)* célja különböző modellek automatikus létrehozása vagy módosítása egy vagy több létező modell alapján, támogatva akár többféle modellezési nyelv együttes alkalmazását. A modelltranszformációk kimenetei lehetnek tesztesetek, tesztadatok, dokumentáció vagy tetszőleges egyéb modell [Bon+99; Rus08]. Ráadásul formális modellek generálásával lehetséges a mérnöki modellek formális ellenőrzése anélkül, hogy a modellezőnek értenie kellene az elméleti, matematikai részleteket, lehetővé téve az ismert, magas szintű modellek használatát a rendszerek definiálására [Bon+99].

A modelltranszformációk egy speciális fajtájának tekinthetőek a *kódgenerátorok*, amelyek kimenetei szövegfájlok, mint például forráskód vagy konfigurációs leíró, vagy a tartalmat leíró absztrakt szintaxisfák. A modellező eszközök többféle kódgenerátort használnak, mint például az Acceleo [Acc], az Epsilon Generation Language (EGL) [Epsilon; Ros+08] vagy az Xtend [Xtend] eszközöket. A kódgenerátorokat gyakran tartják az MDE egyik fontos, gyors fejlesztést támogató technikájának [HWR14].

Az transzformációk felhasználásához kritikus, hogy a felhasznált modell-lekérdezések és -transzformációk helyességéről meggyőződhessünk, ugyanis a transzformáció azonosítatlan hibái az elvégzett ellenőrzéseket érvényteleníthetik, vagy belekerülhetve a fejlesztett alkalmazásba működés közben okozhatnak problémákat. Ugyanakkor a lekérdezések és transzformációk bonyolultságának növekedésével egyre bonyolultabb biztosítani azok helyességét.

1.1. Modell-lekérdezések és -transzformációk ellenőrzése

A modelltranszformációk leírására a hagyományos szoftverrendszerekhez specifikációjához hasonló, ún. modelltranszformációs programok használhatóak. Az elemi transzformációs lépések megfogalmazásához gyakran használnak deklaratív, adatvezérelt szabályokat, míg imperatív vezérlési szerkezetek segítségével lehet ezen elemi lépéseket komplex transzformációkká összefűzni, melynek be- és kimeneti modelljei bonyolult adatszerkezetekkel írhatóak le. Komplex, ipari környezetben, ahol a hibakeresés és a validáció kulcsfontosságú, szükséges a transzformációs programok ellenőrzéséhez [Bau+06; Küs06] a hagyományos szoftvertervezésnél használt módszereket kiterjeszteni.

A transzformációk ellenőrzésére sokféle algoritmus és módszer érhető el. Követve egy áttekintő cikk [AW13] besorolását, ezek tesztesetek és orákulumok generálásától, tételbizonyításon és modellellenőrzésen keresztül gráfelméleten alapuló bizonyításokig terjedhetnek.

- *Modelltranszformációk tesztelésekor* a fő kihívások a bonyolult, de strukturált bemeneti és kimeneti modellek létrehozása [Bau+10] a modellező rendszer számára megfelelő formátumban, valamint a transzformációs nyelvek és módszerek széles köre. Gyakori módszer ekvivalencia-partíciók alkalmazása a programok bemeneteinek megfelelő fedést biztosító modellek generálására [Stu+07; WKC08], de alternatív, például mutáció alapú [FSB04] módszerek is elérhetőek. Tesztorákulumok előállítására [MBL08] is többféle módszer érhető el: ismerten helyes referencia-transzformáció biztosítása és az eredmények összevetése; célmodellek kézi előállítása; vagy a célmodellek kényszer alapú megadása, például OCL kényszerek [Bau+06] vagy gráfminták [OW09] segítségével.

- A *tételbizonyító* alapú megközelítéseknek is sok fajtája érhető el. Az egyik csoportjuk *közvetlenül* a transzformációs szabályokat vizsgálja, vagy a szabályok ekvivalensének előállításával tételbizonyító eszközök számára [Cal+11] vagy a szabályokból invariánsok kinyerésével [Cab+10; ALL09]. Más, *közvetett* módszerek a célmodellek tulajdonságainak ellenőrzésére hagyatkoznak, például jólformáltsági kényszerek [ER10] vagy szemantikus megfelelőség vizsgálatával [Bar+08]. Egyéb megközelítések, beleértve a NASA által használt Amphion eszközt [Van+98] vagy az AutoBayes kódgenerátort [Sch+03], a modellek ellenőrzését megkönnyítő bizonyítékot nyújtanak. Általában a közvetett módszerek egyszerűbbek, mert csak a transzformáció kimenetével foglalkoznak, a transzformáció helyességével azonban nem, ezért a transzformáció újrafuttatása után az ellenőrzést újra végre kell hajtani.

- *Modellellenőrzés* segítségével a transzformációk a transzformációk dinamikus tulajdonságait lehet vizsgálni. A felhasznált módszerek közé tartozik az állapottér kimerítő bejárása [LBA10; Var04], a biszimuláció [KN07] vagy színezett Petri hálók analízise [Wim+09]. A módszerek fő előnye, hogy hiba esetén egy ellenpéldát is előállítanak, amelyek egy végrehajtási napló segítségével megmutatják a hiba bekövetkezésének a lépéseit; ugyanakkor gyakran az állapottér robbanása korlátozza az alkalmazhatóságukat. A gráfok változásai (modellelemek létrejötte vagy törlődése) végtelen állapotteret is eredményezhet-

nek, amelynek kezeléséhez a modell felett valamilyen alkalmas absztrakcióra van szükség [BCK08; RD06].

- Mivel a modellek gyakran gráfként vannak kifejezve, a transzformációk, különösen a gráftranszformációk ellenőrzése gráfelméleti bizonyításokon is alapulhat. Ilyen módszerrel többféle tulajdonság vizsgálata megtörtént, mint a terminálás [BKZ14; Var+06] vagy a konfluencia [Küs06; TG15]. Ezeken felül, mivel a gráftranszformációk segítségével kétirányú transzformációkat is lehet definiálni, a megfordíthatóság ellenőrzése [Her+11] is fontos.

Összevetve a különböző módszereket, jellemző, hogy a használatuk közepesen vagy akár nagy bonyolult lehet [AW13]: a módszereket leíró cikkek inkább a fő algoritmusokra, mintsem az automatizációra vagy az integrációra fókuszálnak. Ezenfelül gyakran az ellenőrzés bonyolultsága számításigényessé teszi az elvégzését.

A módszerek kiegészítésére hasznosak lennének egyszerűbb, „könnyűsúlyú" ellenőrzési technikák, mint például statikus típusellenőrzés vagy (minta-alapú) statikus ellenőrző eszközök (mint a Java programok fejlesztésénél használt FindBugs [Fin] vagy PMD [PMD14] eszközök) használata. Ugyanakkor, modelltranszformációk ellenőrzéséhez kevesebb ilyen eszköz érhető el, például teljes egészében hiányoznak az [AW13] áttekintéséből, noha a gyakorlati hasznuk jól látható más szoftvertervezési feladatok során.

1.2. Hibakeresés modell-lekérdezésekben és -transzformációkban

Hagyományos szoftverfejlesztési feladatoknál, amennyiben egy (állandósult) szoftverhibát *észlelnek*, a fejlesztők egy hibakereső (debugging) [STJ83] folyamatba kezdenek, amelynek a fő célja a *hibaok azonosítása*. A folyamat akkor ér véget, ha a megtalált hibaok *javításra kerül*.

A hiba azonosítása történhet automatizált vagy kézi teszteléssel, vagy kezdődhet egy felhasználói hibajelzés hatására. Általában a hibaok azonosítását tartják a hibakeresés legbonyolultabb szakaszának; a megtalált hibaokot többnyire már egyszerű javítani. A hibakereső eszközök képesek a program végrehajtását akármikor megszakítani *töréspontok* felvételével, és ezután a *program aktuális állapotát*, mint a beállított változók értékeit, megmutatják a fejlesztőnek, lehetővé téve akár a beavatkozást is.

A modelltranszformációk hibakeresése [MV11] hasonló elvek mentén működik. A hibák azonosítása többnyire nem megfelelő kimeneti modell észlelésével kezdődik, ahol egyszerű kérdésekre keressük a választ [HLR07], mint például „miért nincsen adott típusú modellelem a kimeneti modellben?" vagy „mi lehet az oka, hogy ez a metamodell kényszer megsérült a célmodellben?".

Ugyanakkor a hibaok megtalálása hasonló okokból nehézkes, mint a modelltranszformációk tesztelése: a be- és kimenet bonyolult, strukturált modellek segítségével adható meg a modellkezelő környezetekben. Amikor egy transzformációs programot egy *töréspont* mentén megállítunk, a program aktuális állapota magába foglalja a program változóinak az értékét és a modelleket egyaránt. Ezenfelül nagy modellek esetén a modellek vizualizációjához modell- és kontextus-függő szűréseket is kell alkalmazni.

Ráadásul magas szintű transzformációs nyelvek esetén, különösen deklaratív és/vagy grafikus szintaxissal rendelkező esetben, komoly logikai különbségek lehetnek a transzformáció definíciója és végrehajtása között. Például a legtöbb transzformációs nyelv primitív műveletként teszi lehetővé komplex modellfeltételek kiértékelését, például OCL kifejezések vagy gráfminták segítségével. Hasonló a helyzet akkor, ha a keretrendszer egy hagyományos, imperatív nyelven megírt kódot állít elő a transzformáció specifikációjából [Wag+11].

Az irodalomban számos módszer található a modelltranszformációk hibakeresésére: a forrásmodellek elrontása (input model tainting) [Dho+10] meglevő modellek mutációjával próbál

problémás eseteket azonosítani, ahol a transzformáció nem viselkedik helyesen; az ún. mindentudó (omniscient vagy forensic) hibakeresők [HLR07; Cor14] elmentett végrehajtási naplókat használnak, és lehetővé teszik a futás előre- és hátrafelé történő követését.

Mindent egybevetve, a transzformációs fejlesztőkörnyezetek egyelőre csak korlátozott hibakeresési támogatással rendelkeznek, az integrált modellvizualizáció vagy a szűrés támogatása további kutatást igényel.

2. Kutatási kérdések és kihívások

A kutatásom gyakorlati motivációját könnyűsúlyú ellenőrzési eszközök alkalmazása adta modell-lekérdezések és -transzformációk ellenőrzéséhez. Szemben a bonyolultabb ellenőrzési módszerekkel, ezeknek a technikáknak a fő célja a közvetlen, integrált visszajelzés biztosítása a fejlesztőknek, gyors válaszidővel.

Ezek a könnyűsúlyú módszerek gyakran hasznosak, ugyanis elég gyorsak ahhoz, hogy azonnal jelezzék a hibát, ahogy a fejlesztő elköveti őket, jelentősen megkönnyítve a javításukat. Hasonlóképpen, hibakeresés közben ezek a módszerek súgást adhatnak a hibaok lehetséges helyéről. Ugyanakkor ezek a módszerek kevésbé precízek, mint a formális megközelítések, így nem tudják garantálni, hogy a lekérdezés vagy transzformáció hibamentes. Ennek ellenére a gyors, integrált megvalósításuk kiegészítheti a bonyolultabb módszereket korai hibajelzésekkel.

A kutatásom során három fő kihívással foglalkoztam modell-lekérdezések és -transzformációk könnyűsúlyú analízise kapcsán.

1. kihívás: Hogyan azonosítsunk gyakori hibákat modell-lekérdezések és -transzformációk specifikációjában?

Modell-lekérdezések és -transzformációk fejlesztése közben viszonylag egyszerű olyan hibákat elkövetni, amelyek szintaktikusan helyes programot eredményeznek, de futás közben mindig nem-kívánatos eredményt adnak vissza. Ezeket a hibákat gyakran nehéz hibakeresés közben megtalálni a magas szintű, deklaratív specifikációk miatt. Másrészt, az ilyen hibák egyes, gyakori okait hatékonyan lehet azonosítani, mint például egy hívás paramétereinek felcserélése, téves változóhivatkozások használata vagy akár másolási hibák, például hiányzó vagy felesleges kényszerek egy lekérdezés belsejében.

Jellemzően ezeket a hibákat triviális javítani az azonosításuk után, az automatikus azonosításuk akkor a leghatékonyabb, ha rögtön a lekérdezés vagy transzformáció szerkesztése közben végrehajtható. Ennek feltétele, hogy a megközelítés hatékony legyen, de ne igényeljen túl sok erőforrást. Ezenfelül fontos, hogy az ellenőrzés belső adatszerkezeteiből számított eredményt visszavezessük az eredeti specifikációba, és vizuálisan megjelenítsük a fejlesztő számára.

Ilyen lehetséges statikus analízis módszerek a típusellenőrzés és a gráfminták gráfstruktúrájának ellenőrzése: ezek gyorsan végrehajthatóak, ugyanakkor alkalmasak a korábban említett gyakori problémák azonosítására. A disszertációmban ilyen ellenőrzéseket javasolok és értékelek ki modell-lekérdezések és -transzformációk statikus ellenőrzésére.

2. kihívás: Hogyan figyeljük meg a modell változásait hibakeresés közben?

Modelltranszformációk hibakeresése közben gyakran szükséges a modell bizonyos részeinek a vizsgálata a hiba kapcsán, ugyanakkor nem létezik közvetlen vizualizáció, amely *megjeleníti* ezt a fejlesztő által megadott részmodellt, és *frissíti* azt a transzformáció futása közben. Ezek a megjelenítők ugyanakkor hasznosak lennének egyes (belső) állapotgépek aktív állapotainak megje-

lenítéséhez vagy a nyomonkövethetőségi kapcsolatok (traceability link) létrejöttének követése a transzformáció során.

A grafikus modellnézetek készítéséhez többféle eszköz és módszer érhető el [God+07; Bul08; WEK04]. Ugyanakkor ezek nem alkalmasak a hibakeresés közben szükséges ad-hoc vizualizációk támogatására, ugyanis vagy kézzel kell megvalósítani a modellek szűrését a megfelelő információ megjelenítéséhez, vagy kézi kódolást igényelnek a vizualizáció megalkotásához, költségesebbé téve a használatát.

Ráadásul ahhoz, hogy a modellt meg lehessen jeleníteni, vagy exportálni kell a modellt, vagy további fejlesztést igényel a megjelenítés. A modellek exportálása jellemzően egyszerűen megvalósítható - egyes esetekben, mint például a GrGen.net eszközben [Gei+06] ez a keretrendszer szintjén elérhető szolgáltatás, - de az exportált modell frissítése a modell változásaikor nehézkes. A vizualizációs eszközök kézi integrációja lehetséges, de a transzformációk hibakezelése szempontjából ennek a költsége gyakran meghaladja a hibaok megtalálásának költségét bármilyen vizualizáció használata nélkül.

Egyéb környezetek, mint például a Sirius [Sirius] vagy Papyrus [Papyrus], beépítve támogatják a nézeti modellek specifikációját, de továbbra is jelentős fejlesztési munkát igényel a nézetek megalkotása (nagyrészt az eszköz bonyolult funkciókészlete miatt), és a nézetek automatikus frissítése is alapos tervezést igényel.

Ebben a disszertációban egy megközelítést javasolok grafikus nézetek definiálására annotált gráfminták alapján. A megközelítés inkrementális gráfmintaillesztésre és eseményvezérelt modelltranszformációkra építve állandóan frissen tart egy nézeti modellt, amelyet többféle vizualizációs eszközzel is meg lehet jeleníteni.

3. kihívás: Hogyan keressük meg bonyolult transzformációs hibák okát?

A hibakeresés során a fejlesztő gyakran a programnak azon a pontján kezdi az ellenőrzést, ahol a hiba először láthatóvá válik, és megpróbálja kitalálni azon végrehajtott műveletek sorozatát, amelyek a hibás állapothoz vezettek. Ez a fajta következtetés modell-lekérdezések és -transzformációk hibakeresése közben is hasznos, de figyelembe kell venni mind a transzformációs programot, mind a modellt amelyen a program futott.

Ezt a fajta következtetést meg lehet könnyíteni azon modellelemek és program utasítások megjelölésével, amelyeknek hatása volt az azonosított (hibás) állapotra. Így a fejlesztőnek nem kell foglalkoznia a kizárt (független) elemekkel és program utasításokkal, megkönnyítve a hibaok azonosítását.

Hagyományos programozási nyelvek esetén programszeletelési (program slicing) [Wei82] technikákat használnak efféle függőségi analízisre [BGG06; GL91; RH07; SGM02]. A disszertációmban definiálom a program szeletés problémáját modell-lekérdezések és -transzformációk környezetében, és két algoritmust javaslok a szeletek kiszámítására.

3. Új tudományos eredmények

3.1. Analízis technikák

Egyes hibákat, mint például a paraméterek felcserélését vagy érvénytelen változó értékadásokat hatékonyan lehet keresni a program futtatása nélkül típusellenőrzéssel, vagy egy változó felhasználásainak leszámlálásával vagy elérhetőségi analízissel. Mivel ezeket a hibákat könnyű elkövetni, a statikus felderítésük megkönnyíti a hibamentes lekérdezések és transzformációk fejlesztését. Az 1. kihíváshoz kapcsolódó új tudományos eredményeim a következőképp foglalhatók össze:

1. tézis. *Háromféle analízis technikát javasoltam gyakori tévedések azonosítására modell-lekérdezések és -transzformációk specifikációjában: típusellenőrzést, változó felhasználásának leszámlálását és elérhetőségi analízist. Az eredményeket integráltam az EMF-IncQuery és Viatra nyílt forráskódú fejlesztőkörnyezetébe.*

1. ***Típusellenőrzés és típuskövetkeztetés kényszerkielégítési problémákkal:*** *Definiáltam egy leképezést a típushelyesség ellenőrzése és típuskövetkeztetés feladatáról kényszerkielégítési problémákra. A leképezés lehetővé teszi a hibák behatárolását kielégíthetetlen típuskényszerek esetén [4, 14, 25, 26].*
2. ***Modell lekérdezések típuskényszerei:*** *Definiáltam a típusellenőrzés szabályait az EMF-IncQuery és Viatra keretrendszerekben definiálható gráfminták összes támogatott kényszertípusára [4, 13, 26].*
3. ***Modelltranszformációs programok típuskényszerei:*** *Meghatároztam a típusellenőrzés szabályait gráftranszformációs és absztrakt állapotgép szabályok számára, és kiértékeltem a szabályok alkalmazásának a teljesítményét többféle stratégia segítségével [4, 14, 26].*
4. ***Statikus analízis módszerek modell-lekérdezésekhez:*** *Változó felhasználásának leszámlálását és elérhetőségi analízist javasoltam olyan gyakori hibák felderítésére gráfmintákban, mint nem használt változók és független kényszerek keresésére [3].*

Az eredmények a Viatra és a EMF-IncQuery keretrendszerek lekérdező és transzformációs nyelvéhez kapcsolódnak. A Viatra transzformációs nyelve [VB07] Balogh András és Varró Dániel munkája, míg az EMF-IncQuery lekérdező nyelve [13] Bergmann Gábor disszertációjának a része [Ber13].

3.2. Inkrementális grafikus nézetek

A modelltranszformációkhoz tartozó lekérdezések a felhasznált modellnek egyes aspektusait vizsgálják, amelyek gyakran érdekesek hibakeresés közben. Ezek a lekérdezések felhasználhatóak modellmegjelenítő paraméterézésére, lehetővé téve a modell érdekes részeinek a gyors megjelenítését. Ezenfelül az inkrementális lekérdezéskiértékelési technikák használata lehetővé teszi hatékony modellfrissítési mechanizmusok használatát, a modell változásait is megjelenítve, akár egy transzformáció futása közben is.

A 2. kihíváshoz kapcsolódó új tudományos eredményeim a következőképp foglalhatók össze:

2. tézis. *Javasoltam egy algoritmust nézeti modellek származtatására és inkrementális karbantartására modell-lekérdezések alapján. A megközelítés működőképességét egy Viatra keretrendszerre épülő implementáció segítségével demonstráltam.*

1. ***Grafikus nézeti modellek definiálása annotált lekérdezésekkel:*** *Kiterjesztettem az EMF-IncQuery gráfmintanyelvét egy annotációkészlettel, amelynek segítségével grafikus nézeti modelleket lehet megadni. Az annotációk egyértelműen meghatározzák, hogyan jelenítendőek meg az adatok különböző (gráf, fa vagy lista alapú) nézetekben [13, 16]*

2. ***Nézeti modell karbantartás inkrementális lekérdezések segítségével:*** *Definiáltam egy inkrementális modelltranszformációt grafikus modellek származtatására és karbantartására a lekérdezések eredményeinek megjelenítési elemekre leképezésével és az eredményhalmazok változásának nézeti modell módosítási műveletekre való leképezésével. A leképezési szabályok automatikusan származtathatóak az annotált lekérdezésekből.*

A grafikus nézetgenerálás implementációja az EMF-IncQuery inkrementális gráfmintaillesztő technikájára [Ber+10] épülnek, amely Bergmann Gábor disszertációjának része [Ber13], valamint a Viatra eseményvezérelt virtuális gépére, amelynek fő közreműködője Hegedüs Ábel. Debreceni Csaba általánosította a megközelítést tetszőleges nézeti modellek építésére és inkrementális karbantartására egy közös cikkünkben, valamint megvizsgálta a teljesítményét [16]. Az MSc diplomatervében Lunk Péter (az én konzultációm mellett) készített egy prototipikus hibakeresőt a Viatra keretrendszer számára, amely felhasználja az annotáció-alapú nézeti modell definíciókat.

3.3. Modelltranszformációk szeletelése

Programszeletelő [Wei82] technikákat széles körben használnak függőségi analízisre hagyományos programozási nyelvekhez [Xu+05; Tip95]; hasonló módszerek elérhetőek modellező környezetekben modellek szeletelésére [LK10; SP08; Sha+10]. Ugyanakkor ezeket az eredményeket nem lehetséges közvetlenül felhasználni modelltranszformációk szeletelésére, ugyanis itt a modellek és transzformációs programok *együttes* szeletelése határozza meg a függőségi viszonyokat. A 3. kihíváshoz kapcsolódó új tudományos eredményeim a következőképp foglalhatók össze:

3. tézis. *Definiáltam a modelltranszformációk szeletelésének fogalmát függőségi viszonyok azonosítására modellelemek és transzformációs program utasítások között. Dinamikus hátrafelé irányuló szeletelési algoritmusokat adaptáltam a modelltranszformációk szeleteléséhez, és demonstráltam a módszer alkalmazhatóságát modell-lekérdező és -transzformációs keretrendszerben.*

1. ***Modelltranszformációk szeletelése:*** *Definiáltam a szeletelési problémát modelltranszformációs programokra azon modellelemek és transzformációs utasítások azonosítása, amelyek függőségi viszonyban vannak a szeletelési kritériummal. Azonosítottam a fő kihívásokat, amelyek a szeletelő eljárások adaptálása kapcsán előkerülnek [11].*

2. ***Modell-lekérdezések dinamikus visszafele szeletelése:*** *Bevezettem egy algoritmust, amely dinamikus hátrafelé irányuló szeleteket számol modell-lekérdezésekhez Rete hálókra építve [6].*

3. ***Modelltranszformációk dinamikus visszafele szeletelése:*** *Készítettem egy algoritmust, amely dinamikus hátrafelé irányuló szeleteket számol modelltranszformációkhoz egy elmentett futási napló elemzésével [10].*

4. ***Dinamikus visszafele haladó szeletelés kiértékelése:*** *Elvégeztem a javasolt algoritmus skálázhatóságának kiértékelését többféle transzformációs program segítségével [10].*

4. Az új eredmények alkalmazásai

A disszertáció új eredményeinek gyakorlati felhasználhatóságát a következőkben néhány friss alkalmazás segítségével demonstrálom.

I. eszköz: Az EMF-IncQuery fejlesztői környezete

Az EMF-IncQuery nyílt forráskódú modell-lekérdező keretrendszer [EIQ] a Rete algoritmus segítségével inkrementális lekérdező technikát valósít meg az EMF, az egyik leggyakrabban használt modellező környezet felett. Lekérdezések kiértékelésén alapul jó néhány további funkció, mint például automatikus modellvalidáció, adatkötés, inkrementális grafikus nézetek és modelltranszformációk. Az EMF-IncQuery keretrendszer jelentős részét adja Bergmann Gábor doktori disszertációjának.

A Viatra transzformációs nyelv típusellenőrzője (1. tézis) fejlesztése közben szerzett tapasztalatok befolyásolták az EMF-IncQuery mintanyelv tervezését és implementálását: egyfelől a kikövetkeztethető típuskényszerek elhagyásával a minták leírására tömörebbé válhat; másfelől a minták paramétereinek opcionális deklarációja csökkenti a típusok kiértékelésének költségét.

Az EMF-IncQuery fejlesztői környezete egyéb ellenőrzések mellett tartalmazza a disszertációban ismertetett statikus analízis technikákat (1. tézis) és az ismertetett modellmegjelenítő eszközt (2. tézis).

II. eszköz: A Viatra modelltranszformációs keretrendszer

A Viatra keretrendszer [1][VIA] egy nyílt forrású Eclipse projekt, egy teljes fejlesztő- és futtatókörnyezet biztosít modelltranszformációk számára. A rendszer egy saját metamodellező magra épül, tartalmaz egy nyelvet transzformációk specifikációjára, és beépítve támogat keresés alapú és inkrementális lekérdezés kiértékelő stratégiákat. A disszertáció írásakor három éve társvezetője (co-lead) vagyok a projektnek.

Az eredményeim közül a típusellenőrző a Viatra keretrendszer transzformációs nyelvéhez készült el (1. tézis), míg a szeletelés (3. tézis) a keretrendszer hibakereső funkcióit egészíti ki.

A keretrendszer új, Viatra3 verziója egy magas szintű rendszert nyújt transzformációk kötegelt és eseményvezérelt végrehajtásához. A lekérdezések végrehajtásához az EMF-IncQuery által definiált gráfmintakiértékelőt használja, ezáltal beépített támogatást nyújt EMF alapú modellek feldolgozásához.

Mind az EMF-IncQuery, mind a Viatra moduláris, plug-in alapú architektúrával rendelkezik, ami megkönnyíti az integrációjukat létező EMF alapú modellező eszközökhöz és alkalmazásokhoz, mint például a Papyrus UML [Papyrus], Capella [Capella], Sirius [Sirius] vagy az Artop [Artop]. Ezen felül, noha mi EMF alapú modellezőeszközök kontextusában mutattuk be, a projekt lekérdező és transzformációs komponensei sikeresen alkalmazva lettek egyéb, Eclipse-független technológiák felett is, mint például a Matlab [Hor+14; HRS15] vagy az MPS [TV16] rendszerekhez.

A keretrendszereket ezen felül sikeresen használtuk fel különböző ipari projektekben, beleértve egy inkrementális kódgenerátor készítését végrehajtható UML modellekhez [Hor+15] vagy AutoSAR modellek jólformáltsági ellenőrzését autóipari felhasználásra [Ber+10]. Ezek a projektek részben motivációt szolgáltattak a disszertációban bemutatott kutatásokhoz, részben pedig felhasználták az integrált statikus ellenőrzési vagy grafikus nézetek generálásának lehetőségeit.

I. alkalmazás: Inkrementális grafikus nézetek repülőgépipari modellekhez

A 2. tézis eredményeit sikeresen alkalmaztuk a Trans-IMA projektben, egy kutatási együttműködés keretében az Embraer (brazil repülőgépgyártó) és a kutatócsoportunk között. A Trans-IMA projekt célja [Hor+14] egy modellező eszköz fejlesztése volt, amely lehetővé teszi mérnökök számára, hogy magas szintű allokációs folyamat segítségével megalkotott funkció-számítási egység összerendelésekből automatikusan generáljanak Matlab Simulink modelleket a rendszer teljes hardver-szoftver architektúrájának az analíziséhez.

A javasolt megközelítés többféle modellből indul ki: a funkcionális architektúra modellek (FAM) a végrehajtandó funkciókat írják le, a platform leíró modellek (PDM) a hardver architektúrát, amelyeket egy hozzárendelés szerkesztő segítségével felhasználói döntések segítségével félautomatikusan kombinálhatóak. Ezen modellekből a hozzárendelő folyamat generált egy integrált architektúra modellt (IAM) amely eltárolja az összes Simulink-specifikus beállítást, mint függvénykönyvtár és modell hivatkozásokat. Az összes említett modell megtekinthető grafikus szerkesztőkben, amelyek a megfelelő modellek különböző aspektusaira fókuszálnak.

Ezen nézőpontokhoz a 2. tézisben definiált nézeti modell transzformáció segítségével alkottunk nézeti modelleket. A megjelenítéshez a yFiles for Java [WEK04] gráfvizualizációs programkönyvtárt használtuk, amely (1) inkrementális gráfrajzoló algoritmusok segítségével több ezer csomópont és él gyors elrendezésére képes, (2) támogatja a hierarchikus gráfokat és bezárható/kinyitható csomópontokat, és (3) részletes beállítási lehetőségeket az elrendezés és a vizualizáció testreszabásához.

A nézetek deklaratív definíciója megkönnyítette a gyors prototipizálást: lehetséges volt sokféle vizualizációt gyorsan előállítani (az alapértelmezett megjelenítési opciókra építve), majd kipróbálás után kiválasztani az ígéreteseket. Miután a prototípus nézetek elkészültek, a megjelenítést a yFiles Java programozási felületén lehetett testre szabni, például színezést vagy elrendező algoritmust választani, stb.

II. alkalmazás: Modell-lekérdezések használata szoftverkarbantartáshoz

Együttműködésben a Refactoring 2011 Kft.-vel egy projektben vizsgáltuk a modell-lekérdező és -transzformációs eszköz alkalmazhatóságát szoftver refaktorizációra. A projekt célja az volt, hogy csökkentse a szoftvererózió hatását rossz kódolási minták azonosításával és automatikus javítások felajánlásával, amelyek megoldják a hibát, de a fejlesztőre bízva az azonosított hibák tényleges kezelésének menetét. A folyamat a Columbus kódelemző eszköz képességeire épített [Fer+02].

A javasolt refaktorizációs folyamat négy lépésből állt: (1) létre kell hozni egy absztrakt szintaxis gráfot (ASG) a forráskódból a Columbus segítségével, (2) megkeresni az előre definiált ellenpéldák előfordulását az ASG modellben, (3) végrehajtani a kívánt átalakításokat az ASG modellen, és végül (4) visszaírni az ASG modell módosításait az eredeti forrásmodellbe. Mivel a (2) és (3) lépések számára lehet hasznos magas szintű specifikációs nyelvek használata, az itt elvégzendő modell-lekérdezési és -transzformációs problémák megvalósításánál vetettük össze a hagyományos, Java-alapú megvalósítást egy újabb, EMF-IncQuery és Viatra alapúval.

Az elvégzett benchmark mérések azt mutatták, hogy a különböző megközelítések másképp viselkednek: a Columbus kézzel optimalizált modellreprezentációja kevesebb memóriát fogyaszt, így megengedi nagyobb modellek memóriába töltését, míg indexelésre építő technikák, mint az EMF-IncQuery keretrendszerben használt Rete algoritmus jobban teljesít, ha szükséges a hibaminták ismételt kiértékelése a modell részleges módosítása után.

Az EMF-IncQuery keretrendszer számára az ellenpéldák gráfmintaként voltak implementálva, ahol a fejlesztőkörnyezet általam ismertetett bővítései megkönnyítették a helyes minták

megalkotását. Ezen felül az elkészült teljesítménymérés lehetővé tette az EMF-IncQuery által használt különböző mintaillesztő stratégiák kiértékelést valós adatkészleteken.

A teljesítményméréseket ismertető konferenciacikk [8] legjobb cikk díjat kapott az IEEE CSMR-WCRE 2014 Software Evolution Week konferencián, és egy bővített verzió megjelent a Journal of Information and Software Technology folyóiratban [2].

5. Kapcsolódó Publikációk

Publikációk száma: 26
Lektorált publikációk száma: 20
Független hivatkozások száma: 43

Külföldön megjelent, idegen nyelvű folyóiratcikkek

[1] D. Varró, G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, and **Z. Ujhelyi**. "Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework". In: *Software & Systems Modeling* 15.3 (2016), pp. 609–629. DOI: 10.1007/s10270-016-0530-4

[2] **Z. Ujhelyi**, G. Szőke, Á. Horváth, N. I. Csiszár, L. Vidács, D. Varró, and R. Ferenc. "Performance comparison of query-based techniques for anti-pattern detection". In: *Information and Software Technology* 65 (2015), pp. 147–165. DOI: 10.1016/j.infsof.2015.01.003

[3] **Z. Ujhelyi**, Á. Hegedüs, G. Bergmann, Á. Horváth, I. Ráth, and D. Varró. "EMF-IncQuery: An Integrated Development Environment for Live Model Queries". In: *Science of Computer Programming* 98, Part 1 (2015). Fifth issue of Experimental Software and Toolkits (EST): A special issue on Academics Modelling with Eclipse (ACME2012), pp. 80–99. DOI: 10.1016/j.scico.2014.01.004

[4] **Z. Ujhelyi**, Á. Horváth, and D. Varró. "Static Type Checking of Model Transformation Programs". In: *ECEASST* 38 (2010). Ed. by A. Corradini, T. Margaria, J. Padberg, and G. Taentzer. URL: http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/567

[5] Á. Hegedüs, **Z. Ujhelyi**, I. Ráth, and Á. Horváth. "Visualization of Traceability Models with Domain-specific Layouting". In: *ECEASST* 32 (2010). Ed. by T. Margaria, J. Padberg, and G. Taentzer. URL: http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/510

Nemzetközi konferencia-kiadványban megjelent idegen nyelű előadások

[6] **Z. Ujhelyi**, G. Bergmann, and D. Varró. "Rete network slicing for model queries". English. In: *Graph Transformation*. Ed. by R. Echahed and M. Minas. Vol. 9761. Lecture Notes in Computer Science. Springer International Publishing, 2016, pp. 137–152. DOI: 10.1007/978-3-319-40530-8_9

[7] G. Bergmann, I. Dávid, Á. Hegedüs, Á. Horváth, I. Ráth, **Z. Ujhelyi**, and D. Varró. "Viatra 3: A Reactive Model Transformation Platform". English. In: *Theory and Practice of Model Transformations*. Ed. by D. Kolovos and M. Wimmer. Vol. 9152. Lecture Notes in Computer

Science. Springer International Publishing, 2015, pp. 101–110. DOI: 10.1007/978-3-319-21155-8_8

[8] **Z. Ujhelyi**, Á. Horváth, D. Varró, N. I. Csiszár, G. Szőke, L. Vidács, and R. Ferenc. "Anti-pattern detection with model queries: A comparison of approaches". In: *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*. Best paper award. Acceptance rate: 31%. Feb. 2014, pp. 293–302. DOI: 10.1109/CSMR-WCRE.2014.6747181

[9] G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, **Z. Ujhelyi**, and D. Varró. "Integrating Efficient Model Queries in State-of-the-Art EMF Tools". In: *Objects, Models, Components, Patterns*. Ed. by C. Furia and S. Nanz. Vol. 7304. Lecture Notes in Computer Science. Acceptance rate: 31%. Springer Berlin / Heidelberg, 2012, pp. 1–8. DOI: 10.1007/978-3-642-30561-0_1

[10] **Z. Ujhelyi**, Á. Horváth, and D. Varró. "Dynamic Backward Slicing of Model Transformations". In: *Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. ICST '12. Acceptance rate: 27%. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1–10. DOI: 10.1109/ICST.2012.80

[11] **Z. Ujhelyi**, Á. Horváth, and D. Varró. "Towards Dynamic Backwards Slicing of Model Transformations". In: *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*. 2011, pp. 404–407. DOI: 10.1109/ASE.2011.6100084

[12] G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, **Z. Ujhelyi**, and D. Varró. "Implementing efficient model validation in EMF tools". In: *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Nov. 2011, pp. 580–583. DOI: 10.1109/ASE.2011.6100130

[13] G. Bergmann, **Z. Ujhelyi**, I. Ráth, and D. Varró. "A Graph Query Language for EMF Models". In: *Theory and Practice of Model Transformations*. Ed. by J. Cabot and E. Visser. Vol. 6707. Lecture Notes in Computer Science. Acceptance rate: 27%. Springer Berlin / Heidelberg, 2011, pp. 167–182. DOI: 10.1007/978-3-642-21732-6_12

[14] **Z. Ujhelyi**. "Static Type Checking of Model Transformation Programs". In: *Graph Transformations*. Ed. by H. Ehrig, A. Rensink, G. Rozenberg, and A. Schürr. Vol. 6372. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, pp. 413–415. DOI: 10.1007/978-3-642-15928-2_36

Nemzetközi workshop-kiadványban megjelent idegen nyelű előadások

[15] M. Búr, **Z. Ujhelyi**, Á. Horváth, and D. Varró. "Local Search-Based Pattern Matching Features in EMF-IncQuery". English. In: *Graph Transformation*. Ed. by F. Parisi-Presicce and B. Westfechtel. Vol. 9151. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 275–282. DOI: 10.1007/978-3-319-21145-9_18

[16] C. Debreceni, Á. Horváth, Á. Hegedüs, **Z. Ujhelyi**, I. Ráth, and D. Varró. "Query-driven Incremental Synchronization of View Models". In: *Proceedings of the 2nd Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. VAO '14. York, United Kingdom: ACM, 2014, 31:31–31:38. DOI: 10.1145/2631675.2631677

[17] **Z. Ujhelyi**, T. Szabó, I. Ráth, and D. Varró. "Developing and Visualizing Live Model Queries". In: *Proceedings of the 1st Workshop on the Analysis of Model Transformations (AMT) @ MoDELS'12*. AMT '12. Innsbruck, Austria: ACM, 2012. DOI: 10.1145/2432497.2432505

[18] Á. Hegedüs, **Z. Ujhelyi**, and G. Bergmann. "Solving the TTC 2011 Reengineering Case with VIATRA2". In: *TTC 2011: Fifth Transformation Tool Contest, Post-Proceedings*. Ed. by P. V. Gorp, S. Mazanek, and L. Rose. Vol. 74. EPTCS. Zürich, Switzerland: Open Publishing Association, 2011, pp. 136–148. DOI: 10.4204/EPTCS.74.13

[19] Á. Hegedüs, **Z. Ujhelyi**, and G. Bergmann. "Saying Hello World with VIATRA2 - A Solution to the TTC 2011 Instructive Case". In: *TTC 2011: Fifth Transformation Tool Contest, Post-Proceedings*. Ed. by P. V. Gorp, S. Mazanek, and L. Rose. Vol. 74. EPTCS. Zürich, Switzerland: Open Publishing Association, 2011, pp. 302–324. DOI: 10.4204/EPTCS.74.25

[20] Á. Hegedüs, **Z. Ujhelyi**, G. Bergmann, and Á. Horváth. "Ecore to Genmodel case study solution using the VIATRA2 framework". In: *Transformation Tool Contest (TTC '10)*. Ed. by P. V. Gorp, S. Mazanek, and A. Rensink. Malaga, Spain, July 2010

Magyar nyelvű konferenciaelőadások

[21] **Z. Ujhelyi**. "Def-Use Analysis of Model Transformation Programs with Program Slicing". In: *Proceedings of the 18th PhD Minisymposium*. Budapest University of Technology, Economics, Department of Measurement, and Information Systems. Budapest, 2011, pp. 46–49

[22] **Z. Ujhelyi**. "Static Type Checking of Model Transformation Programs". In: *Conference of PhD Students in Computer Science*. 2010, pp. 413–415

[23] **Z. Ujhelyi**. "Static Analysis of Model Transformations". In: *Proceedings of the 17th PhD Minisymposium*. Budapest University of Technology, Economics, Department of Measurement, and Information Systems. 2010, pp. 26–27

[24] **Z. Ujhelyi**. "Modelltranszformációk statikus analízise". In: *Tavaszi Szél Konferenciakiadvány*. In hungarian. 2009

Kutatási jelentések

[25] **Z. Ujhelyi**, Á. Horváth, and D. Varró. *A Generic Static Analysis Framework for Model Transformation Programs*. Technical Report TUB-TR-09-EE19. Budapest University of Technology and Economics, June 2009

[26] **Z. Ujhelyi**, Á. Horváth, and D. Varró. *Static Type Checking of Model Transformations by Constraint Satisfaction Programming*. Technical Report TUB-TR-09-EE20. Budapest University of Technology and Economics, June 2009

Köszönetnyilvánítás

Kutatásaim nagy mértékben támogatta a CONCERTO (ART-2012-333053), DIANA (AERO1-030985), MONDO (ICT-611125) and SECURECHANGE (ICT-FET-231101) EU projektek, a magyar CERT-IMOT (ERC_HU-09-1-2010-0003) projekt, a TÁMOP (4.2.2.B-10/1–2010-0009) és GOP (1.2.1-11-2011-0002) pályázatok és egy közös projektünk az Embraerrel.

Hivatkozások

[Acc] The Eclipse Project. *Acceleo*. http://www.eclipse.org/acceleo.

[ALL09] M. Asztalos, L. Lengyel, and T. Levendovszky. "A Formalism for Describing Modeling Transformations for Verification". In: *Proceedings of the 6th International Workshop on Model-Driven Engineering, Verification and Validation*. MoDeVVa '09. 2009, 2:1–2:10. DOI: 10.1145/1656485.1656487.

[Artop] *Artop: The AUTOSAR Tool Platform*. https://www.artop.org/.

[Aut+16] M. Autili, A. Bertolino, G. D. Angelis, D. D. Ruscio, and A. D. Sandro. "A Tool-Supported Methodology for Validation and Refinement of Early-Stage Domain Models". In: *IEEE Transactions on Software Engineering* 42.1 (2016), pp. 2–25. DOI: 10.1109/TSE.2015.2449319.

[AUT12] AUTOSAR Consortium. *The AUTOSAR Standard*. http://www.autosar.org/. 2012.

[AW13] L. Ab. Rahim and J. Whittle. "A survey of approaches for verifying model transformations". In: *Software & Systems Modeling* (2013). DOI: 10.1007/s10270-013-0358-0.

[Bar+08] P. Barbosa, F. Ramalho, J. de Figueiredo, and A. dos S.Junior. "An Extended MDA Architecture for Ensuring Semantics-Preserving Transformations". In: *32nd Annual IEEE Software Engineering Workshop, 2008. SEW '08.* 32nd Annual IEEE Software Engineering Workshop, 2008. SEW '08. 2008, pp. 33–42. DOI: 10.1109/SEW.2008.8.

[Bar+15] L. Baresi, G. Blohm, D. S. Kolovos, N. Matragkas, A. Motta, R. F. Paige, A. Radjenovic, and M. Rossi. "Formal verification and validation of embedded systems: the UML-based MADES approach". In: *Software & Systems Modeling* 14.1 (2015), pp. 343–363. DOI: 10.1007/s10270-013-0330-z.

[Bau+06] B. Baudry, T. Dinh-Trong, J.-M. Mottu, D. Simmonds, R. France, S. Ghosh, F. Fleurey, and Y. Le Traon. "Model Transformation Testing Challenges". English. In: *ECMDA workshop on Integration of Model Driven Development and Model Driven Testing*. 2006.

[Bau+10] B. Baudry, S. Ghosh, F. Fleurey, R. France, Y. Le Traon, and J.-M. Mottu. "Barriers to Systematic Model Transformation Testing". In: *Commununications of the ACM* 53.6 (2010), pp. 139–143. DOI: 10.1145/1743546.1743583.

[BCK08] P. Baldan, A. Corradini, and B. König. "A framework for the verification of infinite-state graph transformation systems". In: *Information and Computation* 206.7 (2008), pp. 869–907. DOI: 10.1016/j.ic.2008.04.002.

[Ber+10] G. Bergmann, Á. Horváth, I. Ráth, D. Varró, A. Balogh, Z. Balogh, and A. Ökrös. "Incremental Evaluation of Model Queries over EMF Models". In: *Model Driven Engineering Languages and Systems*. Vol. 6394. Lecture Notes in Computer Science. 2010, pp. 76–90. DOI: 10.1007/978-3-642-16145-2_6.

[Ber13] G. Bergmann. "Incremental Model Queries in Model-Driven Design". PhD dissertation. Budapest University of Technology and Economics, 2013.

[Béz05] J. Bézivin. "On the unification power of models". In: *Software & Systems Modeling* 4.2 (2005), pp. 171–188. DOI: 10.1007/s10270-005-0079-0.

[BG05] S. Beydeda and V. Gruhn. *Model-Driven Software Development*. Springer-Verlag New York, Inc., 2005.

[BGG06] A. Beszedes, T. Gergely, and T. Gyimothy. "Graph-Less Dynamic Dependence-Based Dynamic Slicing Algorithms". In: *2006 Sixth IEEE International Workshop on Source Code Analysis and Manipulation*. 2006, pp. 21–30. DOI: 10.1109/SCAM.2006.17.

[BKZ14] H. J. S. Bruggink, B. König, and H. Zantema. "Termination Analysis for Graph Transformation Systems". In: *Theoretical Computer Science: 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*. Ed. by J. Diaz, I. Lanese, and D. Sangiorgi. Springer Berlin Heidelberg, 2014, pp. 179–194. DOI: 10.1007/978-3-662-44602-7_15.

[Bon+99] A. Bondavalli, M. Dal Cin, D. Latella, and A. Pataricza. "High-level Integrated Design Environment for Dependability". WORDS'99, 1999 Workshop on Real-Time Dependable System. 1999.

[Bro+14] M. Broy, S. Kirstan, H. Krcmar, and B. Schätz. "What is the Benefit of a Model-Based Design of Embedded Software Systems in the Car Industry?" In: *Software Design and Development: Concepts, Methodologies, Tools, and Applications*. 2014, pp. 310–334. DOI: 10.4018/978-1-4666-4301-7.ch017.

[Bul08] I. Bull. "Model Driven Visualization: Towards A Model Driven Engineering Approach For Information Visualization". Ph.D. Thesis. University of Victoria, BC, Canada, 2008.

[Cab+10] J. Cabot, R. Clarisó, E. Guerra, and J. de Lara. "Verification and validation of declarative model-to-model transformations through invariants". In: *Journal of Systems and Software*. Computer Software and Applications 83.2 (2010), pp. 283–302. DOI: 10.1016/j.jss.2009.08.012.

[Cal+11] D. Calegari, C. Luna, N. Szasz, and Á. Tasistro. "A Type-Theoretic Framework for Certified Model Transformations". In: *Formal Methods: Foundations and Applications*. Ed. by J. Davies, L. Silva, and A. Simao. Lecture Notes in Computer Science 6527. Springer Berlin Heidelberg, 2011, pp. 112–127.

[Capella] PolarSys. *Capella*. http://www.polarsys.org/capella.

[Cla08] E. M. Clarke. "The Birth of Model Checking". In: *25 Years of Model Checking*. Ed. by O. Grumberg and H. Veith. Lecture Notes in Computer Science 5000. Springer Berlin Heidelberg, 2008, pp. 1–26.

[Cor14] J. Corley. "Exploring Omniscient Debugging for Model Transformations". In: *Joint Proceedings of MODELS 2014 Poster Session and the ACM Student Research Competition (SRC) co-located with the 17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014), Valencia, Spain, September 28 - October 3, 2014*. Vol. 1258. CEUR Workshop Proceedings. 2014, pp. 63–68. URL: http://ceur-ws.org/Vol-1258/src3.pdf.

[Dho+10] P. Dhoolia, S. Mani, V. S. Sinha, and S. Sinha. "Debugging model-transformation failures using dynamic tainting". In: *Proceedings of the 24th European conference on Object-oriented programming*. ECOOP'10. ACM ID: 1883983. 2010, pp. 26–51.

[EIQ] The Eclipse Project. *EMF-IncQuery*. http://www.eclipse.org/incquery.

[Epsilon] The Eclipse Project. *Epsilon*. https://www.eclipse.org/epsilon/.

[ER10] M. Egea and V. Rusu. “Formal executable semantics for conformance in the MDE framework”. In: *Innovations in Systems and Software Engineering* 6.1-2 (2010), pp. 73–81. DOI: 10.1007/s11334-009-0108-1.

[Fer+02] R. Ferenc, Á. Beszédes, M. Tarkiainen, and T. Gyimóthy. “Columbus – Reverse Engineering Tool and Schema for C++”. In: *Software Maintenance, 2002. Proceedings. International Conference on*. 2002, pp. 172–181. DOI: 10.1109/ICSM.2002.1167764.

[Fin] FindBugs. *FindBugs – Find Bugs in Java Programs*. http://findbugs.sourceforge.net/. (Visited on 05/16/2013).

[FSB04] F. Fleurey, J. Steel, and B. Baudry. “Validation in model-driven engineering: testing model transformations”. In: *2004 First International Workshop on Model, Design and Validation, 2004. Proceedings*. 2004 First International Workshop on Model, Design and Validation, 2004. Proceedings. 2004, pp. 29–40. DOI: 10.1109/MODEVA.2004.1425846.

[Gei+06] R. Geiß, G. Batz, D. Grund, S. Hack, and A. Szalkowski. “GrGen: A Fast SPO-Based Graph Rewriting Tool”. English. In: *Graph Transformations*. Vol. 4178. Lecture Notes in Computer Science. 2006, pp. 383–397. DOI: 10.1007/11841883_27.

[GL91] K. Gallagher and J. Lyle. “Using Program Slicing in Software Maintenance”. In: *IEEE Transactions on Software Engineering* 17.8 (1991), pp. 751–761. DOI: 10.1109/32.83912.

[God+07] P. Godinho, B. Meiguins, A. Goncalves Meiguins, R. Casseb do Carmo, M. de Brito Garcia, L. Almeida, and R. Lourenco. “PRISMA - A Multidimensional Information Visualization Tool Using Multiple Coordinated Views”. In: *Information Visualization, 2007. IV ’07. 11th International Conference*. 2007, pp. 23–32. DOI: 10.1109/IV.2007.90.

[Her+11] F. Hermann, H. Ehrig, F. Orejas, K. Czarnecki, Z. Diskin, and Y. Xiong. “Correctness of Model Synchronization Based on Triple Graph Grammars”. In: *Model Driven Engineering Languages and Systems*. Ed. by J. Whittle, T. Clark, and T. Kühne. Lecture Notes in Computer Science 6981. Springer Berlin Heidelberg, 2011, pp. 668–682.

[HLR07] M. Hibberd, M. Lawley, and K. Raymond. “Forensic Debugging of Model Transformations”. In: *Model Driven Engineering Languages and Systems*. Ed. by G. Engels, B. Opdyke, D. C. Schmidt, and F. Weil. Lecture Notes in Computer Science 4735. Springer Berlin Heidelberg, 2007, pp. 589–604.

[Hor+14] Á. Horváth, Á. Hegedüs, M. Búr, V. Varró, R. R. Starr, and S. Mirachi. “Hardware-software allocation specification of IMA systems for early simulation”. In: *2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*. 2014, pp. 4D3-1–4D3-15. DOI: 10.1109/DASC.2014.6979474.

[Hor+15] Á. Horváth, I. Ráth, Á. Hegedüs, and Á. Balogh. “Decreasing your Coffee Consumption with Incremental Code Regeneration”. In: *EclipseCon France*. https://www.eclipsecon.org/france2015/session/decreasing-your-coffee-consumption-incremental-code-regeneration. 2015.

[HRS15] Á. Horváth, I. Ráth, and R. R. Starr. “Massif – the love child of Matlab Simulink and Eclipse”. In: *EclipseCon North America*. https://www.eclipsecon.org/na2015/session/massif-love-child-matlab-simulink-and-eclipse. 2015.

[HWR14] J. Hutchinson, J. Whittle, and M. Rouncefield. “Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure”. In: *Science of Computer Programming* 89, Part B (2014). Special issue on Success Stories in Model Driven Engineering, pp. 144–161. DOI: 10.1016/j.scico.2013.03.017.

[KN07] G. Karsai and A. Narayanan. “On the Correctness of Model Transformations in the Development of Embedded Systems”. In: *Composition of Embedded Systems. Scientific and Industrial Issues*. Ed. by F. Kordon and O. Sokolsky. Lecture Notes in Computer Science 4888. Springer Berlin Heidelberg, 2007, pp. 1–18.

[Küs06] J. M. Küster. “Definition and validation of model transformations”. In: *Software & Systems Modeling* 5.3 (2006), pp. 233–259. DOI: 10.1007/s10270-006-0018-8.

[LBA10] L. Lúcio, B. Barroca, and V. Amaral. “A Technique for Automatic Validation of Model Transformations”. In: *Model Driven Engineering Languages and Systems*. Ed. by D. C. Petriu, N. Rouquette, and Ø. Haugen. Lecture Notes in Computer Science 6394. Springer Berlin Heidelberg, 2010, pp. 136–150. DOI: 10.1007/978-3-642-16145-2_10.

[LK10] K. Lano and S. Kolahdouz-Rahimi. “Slicing of UML Models Using Model Transformations”. In: *Model Driven Engineering Languages and Systems*. Vol. 6395. Lecture Notes in Computer Science. 2010, pp. 228–242. DOI: 10.1007/978-3-642-16129-2_17.

[MBL08] J.-M. Mottu, B. Baudry, and Y. Le Traon. “Model transformation testing: oracle issue”. In: *IEEE International Conference on Software Testing Verification and Validation Workshop, 2008. ICSTW ’08.* 2008, pp. 105–112. DOI: 10.1109/ICSTW.2008.27.

[Mur89] T. Murata. “Petri nets: Properties, analysis and applications”. In: *Proceedings of the IEEE* 77.4 (1989), pp. 541–580. DOI: 10.1109/5.24143.

[MV11] R. Mannadiar and H. Vangheluwe. “Debugging in Domain-Specific Modelling”. In: *Software Language Engineering*. Ed. by B. Malloy, S. Staab, and M. v. d. Brand. Lecture Notes in Computer Science 6563. Springer Berlin Heidelberg, 2011, pp. 276–285.

[OAS07] OASIS. *Web Services Business Process Execution Language Version 2.0 (OASIS Standard)*. "http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html". 2007.

[Obj10] Object Management Group. *OMG System Modeling Language (SysML)*. http://www.omg.org/spec/SysML/index.htm. 2010.

[Obj11a] Object Management Group. *Business Process Model And Notation (BPMN)*. http://www.omg.org/spec/BPMN/index.htm. 2011.

[Obj11b] Object Management Group. *UML Profile For MARTE: Modeling And Analysis Of Real-Time Embedded Systems*. http://www.omg.org/spec/MARTE/index.htm. 2011.

[Obj11c] Object Management Group. *Unified Modeling Language (UML)*. http://www.omg.org/spec/UML/index.htm. 2011.

[OW09] F. Orejas and M. Wirsing. “On the Specification and Verification of Model Transformations”. In: *Semantics and Algebraic Specification*. Ed. by J. Palsberg. Lecture Notes in Computer Science 5700. Springer Berlin Heidelberg, 2009, pp. 140–161.

[Papyrus] The Eclipse Project. *Papyrus*. http://www.eclipse.org/papyrus.

[PMD14] PMD. *PMD checker*. http://pmd.sourceforge.net/. 2014. (Visited on 2014).

[Pri08] P. Prisaznuk. "ARINC 653 role in Integrated Modular Avionics (IMA)". In: *Digital Avionics Systems Conference, 2008. IEEE/AIAA 27th*. 2008. DOI: 10.1109/DASC.2008.4702770.

[RD06] A. Rensink and D. Distefano. "Abstract Graph Transformation". In: *Electronic Notes in Theoretical Computer Science* 157.1 (2006), pp. 39–59. DOI: 10.1016/j.entcs.2006.01.022.

[RH07] V. P. Ranganath and J. Hatcliff. "Slicing concurrent Java programs using Indus and Kaveri". In: *International Journal on Software Tools for Technology Transfer* 9.5 (2007), pp. 489–504. DOI: 10.1007/s10009-007-0043-0.

[Ros+08] L. M. Rose, R. F. Paige, D. S. Kolovos, and F. A. C. Polack. "The Epsilon Generation Language". In: *Model Driven Architecture – Foundations and Applications: 4th European Conference, ECMDA-FA 2008, Berlin, Germany, June 9-13, 2008. Proceedings*. Ed. by I. Schieferdecker and A. Hartman. Springer Berlin Heidelberg, 2008, pp. 1–16. DOI: 10.1007/978-3-540-69100-6_1.

[Rus08] J. Rushby. "Automated Test Generation and Verified Software". In: *Verified Software: Theories, Tools, Experiments*. 2008, pp. 161–172. DOI: 10.1007/978-3-540-69149-5_18.

[SAE09] SAE International. *Architecture Analysis & Design Language (AADL)*. http://standards.sae.org/as5506a/. 2009.

[Sch+03] J. Schumann, B. Fischer, M. Whalen, and J. Whittle. "Certification support for automatically generated programs". In: *Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003*. 2003. DOI: 10.1109/HICSS.2003.1174914.

[SGM02] G. Szilágyi, T. Gyimóthy, and J. Małuszyński. "Static and Dynamic Slicing of Constraint Logic Programs". In: *Automated Software Engineering* 9 (1 2002), pp. 41–65.

[Sha+10] A. Shaikh, R. Clarisó, U. K. Wiil, and N. Memon. "Verification-driven slicing of UML/OCL models". In: *25th IEEE/ACM Int. Conf. on Automated Software Engineering*. 2010, pp. 185–194.

[Sirius] The Eclipse Project. *Sirius*. http://www.eclipse.org/sirius.

[SP08] I. Schaefer and A. Poetzsch-Heffter. "Slicing for model reduction in adaptive embedded systems development". In: *Proceedings of the 2008 International Workshop on Software Engineering for Adaptive and Self-managing Systems*. SEAMS '08. 2008, 25–32. DOI: 10.1145/1370018.1370024.

[STJ83] R. L. Sedlmeyer, W. B. Thompson, and P. E. Johnson. "Knowledge-based fault localization in debugging". In: *Journal of Systems and Software* 3.4 (1983), pp. 301–307. DOI: 10.1016/0164-1212(83)90016-X.

[Stu+07] I. Stuermer, M. Conrad, H. Doerr, and P. Pepper. "Systematic Testing of Model-Based Code Generators". In: *IEEE Transactions on Software Engineering* 33.9 (2007), pp. 622–634.

[TG15] G. Taentzer and U. Golas. "Towards Local Confluence Analysis for Amalgamated Graph Transformation". In: *Graph Transformation: 8th International Conference, ICGT 2015, Held as Part of STAF 2015, L'Aquila, Italy, July 21-23, 2015. Proceedings*. Ed. by F. Parisi-Presicce and B. Westfechtel. Springer International Publishing, 2015, pp. 69–86. DOI: 10.1007/978-3-319-21145-9_5.

[Tip95] F. Tip. “A survey of program slicing techniques”. In: *Journal of Programming Languages 3(3)* (1995), pp. 121–189.

[TV16] S. E. Tamás Szabó and M. Voelter. “IncA: A DSL for the Definition of Incremental Program Analyses”. In: *Automated Software Engineering, 31st IEEE/ACM International Conference on*. Accepted. 2016.

[Van+98] J. Van Baalen, P. Robinson, M. Lowry, and T. Pressburger. “Explaining synthesized software”. In: *13th IEEE International Conference on Automated Software Engineering, 1998. Proceedings*. 13th IEEE International Conference on Automated Software Engineering, 1998. Proceedings. 1998, pp. 240–248. DOI: 10.1109/ASE.1998.732661.

[Var+06] D. Varró, S. Varró–Gyapay, H. Ehrig, U. Prange, and G. Taentzer. “Termination Analysis of Model Transformations by Petri Nets”. In: *Graph Transformations*. Ed. by A. Corradini, H. Ehrig, U. Montanari, L. Ribeiro, and G. Rozenberg. Lecture Notes in Computer Science 4178. Springer Berlin Heidelberg, 2006, pp. 260–274. DOI: 10.1007/11841883_19.

[Var04] D. Varró. “Automated formal verification of visual modeling languages by model checking”. English. In: *Software and Systems Modeling* 3.2 (2004), pp. 85–113. DOI: 10.1007/s10270-003-0050-x.

[VB07] D. Varró and A. Balogh. “The Model Transformation Language of the VIATRA2 Framework”. In: *Science of Computer Programming* 68.3 (2007), pp. 214–234.

[VIA] The Eclipse Project. *VIATRA Model Transformation Framework*. http://www.eclipse.org/viatra.

[Wag+11] D. Wagelaar, M. Tisi, J. Cabot, and F. Jouault. “Model Driven Engineering Languages and Systems: 14th International Conference, MODELS 2011, Wellington, New Zealand, October 16-21, 2011. Proceedings”. In: ed. by J. Whittle, T. Clark, and T. Kühne. Springer Berlin Heidelberg, 2011. Chap. Towards a General Composition Semantics for Rule-Based Model Transformation, pp. 623–637. DOI: 10.1007/978-3-642-24485-8_46.

[Wei82] M. Weiser. “Programmers use slices when debugging”. In: *Communications of the ACM* 25.7 (1982), pp. 446–452. DOI: 10.1145/358557.358577.

[WEK04] R. Wiese, M. Eiglsperger, and M. Kaufmann. “yFiles — Visualization and Automatic Layout of Graphs”. In: *Graph Drawing Software*. Ed. by M. Jünger and P. Mutzel. Mathematics and Visualization. Springer Berlin Heidelberg, 2004, pp. 173–191. DOI: 10.1007/978-3-642-18638-7_8.

[WHR14] J. Whittle, J. Hutchinson, and M. Rouncefield. “The State of Practice in Model-Driven Engineering”. In: *IEEE Software* 31.3 (2014), pp. 79–85. DOI: 10.1109/MS.2013.65.

[Wim+09] M. Wimmer, G. Kappel, A. Kusel, W. Retschitzegger, J. Schönböck, and W. Schwinger. “Right or Wrong? Verification of Model Transformations using Colored Petri Nets”. In: *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM’09)*. 2009.

[WKC08] J. Wang, S.-K. Kim, and D. Carrington. “Automatic Generation of Test Models for Model Transformations”. In: *19th Australian Conference on Software Engineering, 2008. ASWEC 2008*. 19th Australian Conference on Software Engineering, 2008. ASWEC 2008. 2008, pp. 432–440. DOI: 10.1109/ASWEC.2008.4483232.

[Xtend] The Eclipse Project. *Xtend - Modernized Java*. http://www.eclipse.org/xtend.

[Xu+05] B. Xu, J. Qian, X. Zhang, Z. Wu, and L. Chen. “A brief survey of program slicing”. In: *ACM SIGSOFT Software Engineering Notes* 30.2 (2005), pp. 1–36.