XI. FIATAL MŰSZAKIAK TUDOMÁNYOS ÜLÉSSZAKA

Kolozsvár, 2006. március 24-25.

MODELLTRANSZFOMRÁCIÓK IMPLEMENTÁCIÓJAINAK AUTOMATIKUS, TRANSZFOMRÁCIÓ ALAPÚ GENERÁLÁSA¹

Horváth Ákos

MDA aims a direct implementation starting from a Platform Independent Model (PIM) describing the business logic. The construction of software components from high level models and the mappings between various levels of modeling highly relies on model transformations (MT). As model transformations are growing in huge numbers, a new demand is formed to separate the design of a transformation from the running by using high- level platform-independent models in design time and to derive the running platform source code from these high level models.

The current paper presents a new approach using generic and meta-transformations for generating platform-specific transformations from model transformation specifications defined by a combination of graph transformation and abstract state machine rules. The essence of the approach is to store transformation rules as ordinary models in the model space, which can be processed later by the meta-transformations, which generates the platform-specific (Java, C++, C#, etc.) transformations.

Az MDA a szoftverek közvetlen és automatikus implementációját támogatja egy, a rendszer funkcionális specifikációját tartalmazó platform független modellből (PIM) kiindulva. A szoftver komponensek magas szintű modellekből való generálásához, és a modellezési szintek közötti leképezések elvégzéséhez nagymértékben modelltranszformációkra van szükség. Ahogyan a modelltranszformációk száma egyre növekszik új igényként jelent meg, hogy szétválasszák a modelltranszformációk tervezését a konkrét futtatásuktól: magas szintű matematikai modelleket használva a tervezés alatt, a futási időben használt transzformációs programot pedig ebből a modellből származtassák le.

Ez a cikk egy olyan új, gráfranzformációk és absztrakt állapotgépek kombinációjával megadott modelltranszformációk specifikációjából történő automatikus platform specifikus transzformációk generálását mutatja be, ami generikus és meta transzformációkat használ. A módszer lényege, hogy a transzformációs szabályokat is, mint egyszerű modell elemeket jelentetünk meg a modell térben, így lehetőség nyílik rá, hogy a későbbiekben feldolgozhassuk őket és legeneráljuk belőlük a platform specifikus (Java, C++, C#, stb.) transzformációkat.

Bevezetés:

Manapság az egyik leggyorsabban terjedő szoftverfejlesztési módszertan az OMG által létrehozott Modell-vezérelt Architektúra (MDA). Az MDA egy lehetséges módszert definiál nagy informatikai rendszerek fejlesztéséhez, oly módon, hogy elkülöníti az üzleti funkciók specifikációját a konkrét implementáció-specifikus részekről.

A fejlesztés kezdő lépéseként egy platform-független modellt (Platform Independent Model - PIM) kell elkészítenünk, amely magában foglalja a megvalósítandó rendszer egy magas-

¹Ezt a kutatást a SENSORIA európai uniós projekt támogatja (IST-3-016004).

szintű, konkrét implementációtól független, teljes funkcionalitásának leírását. A PIM validálása után kiválasztjuk az implementációs platformot, és a PIM-et modelltranszformációs lépések sorozatával átalakítjuk a rendszer platform-specifikus modelljévé (Platform Specific Model - PSM). Ezt követően a PSM modellből kiindulva egy kódgenerálási lépésben a rendszer forráskódját állítjuk elő. Mint látható az MDA sikerének egyik kulcskérdése vált a modelltranszformációk minél nagyobb fokú automatizálhatósága.

Modelltranszformációk.

Az MDA egyre szélesebb körben való terjedésével párhuzamosan egyre nagyobb és komplexebb modelltranszformációk (MT) jelentek meg, amelyek már megkövetelték a teljes életcikluson keresztüli szoftvertámogatást. Azonban a modelltranszformációk életciklusának egyes fázisai egymásnak ellentmondó igényeket támasztanak a szoftver eszközökkel szemben.

Például, a transzformációk tervezési és specifikációs fázisaiban jobban megfelel egy alapvetően magas absztrakciós szinten működő, a nagyfokú újrafelhasználhatóságot támogató és interpretált futtatást biztosító tervezőeszköz. Amennyiben azonban a transzformációkat a végleges célplatformon kívánjuk futtatni, ahol a teljesítmény a legfontosabb szempont, jobban megfelel inkább a konkrét platformra (Java, C++, C#, stb.) optimalizált, lefordított - a transzformáció tervezőeszközétől akár függetlenül is futtatható – transzformációs programok. Fontos azonban, hogy nem szabad különválasztani ezt a két eltérő megvalósítást, és egy fejlett MT keretrendszernek támogatnia kell tudnia mindkét megközelítést [2].

Platform-specifikus transzformációk generálása.

Manapság az ilyenfajta leképezéseket még nagyobb részt kézzel, specifikusan a konkrét feladatokhoz készítik el, ami leképezések később nem vagy csak alig használhatóak újra. Jelenleg a piacon általános jól használható eszközökből is igen szűkös a kínálat.

Erre a problémára dolgoztunk ki egy újfajta megközelítést, ami az általános célú VIATRA2 [6] modelltranszformációs keretrendszert használja a transzformációk magas szintű modelltranszformáció specifikációinak az elkészítéséhez. Ezekből a specifikációkból automatikusan generálja le a konkrét futtatható kódot úgy, hogy magát az automatikus generálást is modelltranszformációs (bővebben ismertetve: [4]) lépések sorozataként valósítja meg -ezáltal a generálás a keretrendszeren belül végezhető el- és a későbbiekben is rendelkezésre állnak a transzformáció egyes elemei, így könnyítve meg az esetleges más platformokra történő portolását.

A VIATRA2 keretrendszerben a modelltranszformációkat gráftranszformációk (GT) [1] és absztrakt állapotgépek (ASM) [3] kombinációjával tudjuk megadni, ahol az elemi modell manipulációs lépéseket a GT szabályok, a vezérlési szerkezeteket pedig az ASM-ek biztosítják. Az ASM szabályok nagyban hasonlítanak a konvencionális programozási nyelvekben használt imperatív vezérlési szerkezetekhez, míg a GT szabályok deklaratív módon gráfok szabály és minta alapú manipulációját végzik. A GT szabályok alapvetően két részből állnak, egy keresendő gráf mintából, és egy akció részből, ami az illeszkedés esetén végbemenő modell manipulációkat írják le.

Az 1. ábra demonstrálja a megvalósított alrendszer egyes lépéseit:

1. ábra PST generátor felépítése

- Első lépésben különválasztjuk a GT és ASM szabályokat, és megkezdjük a feldolgozásukat a bemeneti paraméterek vizsgálatával.
- Ezt követően az ASM szabályok feldolgozása következik, ami nagyban hasonlít egy egyszerű OO programozás feldolgozásával.
- Az ASM szabályok után minden egyes GT szabályt két lépésben dolgozunk fel. Elsőként a keresendő gráfmintákhoz előállítunk egy GT mintahívást (paraméter összerendelés, kimenet meghatározása, stb.), majd pedig az akció részben előírt manipulációs utasítások feldolgozása következik, ami akár ASM szabályokat is tartalmazhat, újra hívva az ASM szabály feldolgozó modult.
- Amikor az összes GT szabály feldolgozása befejeződött akkor, minden egyes GT mintahíváshoz egy új keresési gráfot inicializálunk, és kiértékeljük a keresést szűkítő feltételeket

- Ezt követően optimalizált keresési útvonalat (a gráf minta bejárási útvonala, részletesen ismertetve [5]) generálunk minden egyes keresési gráfhoz, ezzel téve determinisztikussá a bejárásainkat.
- Legvégül pedig a kimeneti transzformációs program forrás file-jait generáljuk le template-k segítségével. Más kimeneti OO nyelvek esetén csak ezt a részt kell kicserélni.

A nagyfokú modularitás és teljesen a VIATRA2-ben implementált modulok következtében a megvalósított alrendszert könnyen tovább lehet bővíteni. Például új platformokra történő kódgenerálás, újfajta keresési útvonal készítő algoritmusok implementálása esetleg más modelltranszformációs rendszerekhez való import kimenet generálása.

Összegzés.

A modelltranszformációk egyre szélesebb körben terjednek a legkülönbözőbb ipari alkalmazásokban, fontos a minél nagyobb fokú eszköztámogatásuk. Ebben a cikkben egy olyan új módszert mutattunk be, amivel lehetőségünk nyílik magas absztrakciós szinten megvalósított modelltranszformációk különböző platformokra történő portolására, ezzel segítve a hatékonyabb hordozhatóságot és a nagyobb modelleken történő hatékony futtatást.

Hivatkozások.

- [1] Ehrig, H., G. Engels, H.-J. Kreowski and G. Rozenberg, editors, 2: Applications, Languages and Tools in: “Handbook of Graph Grammars and Computing by Graph Transformations”, World Scientific, 1999.
- [2] Bézivin, J., N. Farcet, J.-M. Jézéquel, B. Langlois and D. Pollet, Reflective model driven engineering, in: P. Stevens, J. Whittle and G. Booch, editors, Proc. UML 2003: 6th International Conference on the Unified Modeling Language, LNCS 2863 (2003), pp. 175–189.
- [3] Börger, E. and R. Stark, “Abstract State Machines. A method for High-Level System Design and Analysis,” Springer-Verlag, 2003.
- [4] Varró, D. and A. Pataricza, Generic and meta-transformations for model transformation engineering, in: T. Baar, A. Strohmeier, A. Moreira and S. Mellor, editors, Proc. UML 2004: 7th Int. Conf. on the Unified Modeling Language, LNCS 3273 (2004), pp. 290–304.
- [5] Varró, G., D. Varró and K. Friedl, Adaptive graph pattern matching for model transformations using model-sensitive search plans, in: G. Karsai and G. Taentzer, editors, GraMot 2005, Int. Workshop on Graph and Model Transformations, ENTCS, in press.
- [6] VIATRA2 Eclipse GMT subproject, <http://www.eclipse.org/gmt/>

Horváth Ákos, hallgató

Budapesti Műszaki és Gazdaságtudományi Egyetem,

Méréstechnika és Információs Rendszerek Tsz.

H-1117 Budapest, XI. Magyar tudósok körútja 2. Informatika ép. I.B. 414

Tel: +36 (1) 463-3579 Fax: +36 (1) 463-2667

Email: ha442@hszk.bme.hu