X. FIATAL MŰSZAKIAK TUDOMÁNYOS ÜLÉSSZAKA

Kolozsvár, 2005. március 18-19.

VIATRA 2: EGY ÁLTALÁNOS MODELL- TRANSZFORMÁCIÓ FEJLESZTŐ KERETRENDSZER

Balogh András

Recent achievements in software engineering aim at the simultaneous increase of software productivity and quality by increasing the level of the automation of the design and implementation process. The most characteristic appearance of this trend is that of the Model Driven Architecture (MDA) [1] initiative of OMG which aims at 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 involve model transformations, mappings between various models or even modeling languages. These transformations require the simultaneous handling of modeling and programming languages. Model transformations can be applied for model analysis and model checking applications. The systems designer works with an expressive, easy-to-use, semi-formal modeling technology, like UML, and the model correctness analysis has to be done in a precise mathematical formalism. The mapping between the two domains requires model transformations that can handle multiple modeling languages at the same time. Currently there are no universal development tools for model transformations that could support the whole transformation development process. We present a general model transformation framework in this paper that can be used in the development of a large scale of model transformations.

A szoftverfejlesztés területén elért legújabb eredmények a fejlesztés produktivitását és minőségét célozzák az automatikus tervezési és implementálási eljárások bevezetésének segítségével. A legerőteljesebb irányzat a modell-vezérelt architektúra (MDA)[1], mely az Object Management Group (OMG) kezdeményezése, mely a szoftverek közvetlen és automatikus implementációját támogatja egy, a rendszer funkcionális specifikációját tartalmazó platform független modellből (PIM). A szoftver komponensek magas szintű modellekből való generálása, valamint a modellezési szintek közötti leképezések elvégzéséhez modell transzformációkra van szükség. Ezeknek egy időben kell kezelni a modellezési és programozási nyelveket.

A modell transzformációk másik jelentős felhasználási területe a modell analízis és modell ellenőrzés területe. A rendszerfejlesztő egy egyszerűen használható és szemléletes modellezési nyelven dolgozik, de az analízis technikák matematikai formalizmust igényelnek. Transzformációk segítségével a modelleket át lehet alakítani a tervező által használt modellezési nyelvről az analízis eszközök nyelvére. Ez olyan transzformációkat igényel, melyek több modellezési nyelvet támogatnak egy időben.

Jelenleg nincs olyan univerzálisan használható transzformáció fejlesztő környezet, mely a fejlesztés teljes életciklusát támogatja. E cikkben bemutatunk egy olyan keretrendszert, mely széles körű támogatást nyújt különféle transzformációk fejlesztéséhez.

1. Bevezetés

A Modell-vezérelt Architektúra (MDA) az Object Management Group (OMG) egyre inkább terjedő koncepciója [1]. Egy lehetséges módszert definiál nagy informatikai rendszerek szabványos modellezésére, ami elkülöníti a rendszer funkcionális specifikációját az implementáció-specifikus részekről.

A fejlesztési folyamat egy platform-független modell (Platform Independent Model – PIM) elkészítésével kezdődik, mely specifikálja a tervezett rendszer funkcionális jellemzőit. Ez a modell nem tartalmaz semmiféle implementációval kapcsolatos információt. A PIM elkészülte után kiválasztják az implementációs platformot vagy platformokat és a PIM-et (automatikusan vagy manuálisan) átalakítják a rendszer platform-specifikus modelljére (Platform Specific Model – PSM). A PSM a rendszer funkcionális és implementáció-specifikus elemeit is tartalmazza, ezáltal egy adott platformhoz tartozik. A kész rendszer forráskódját a PSM modellből lehet generálni.

2. Modell-transzformációk

Mivel az MDA paradigma erősen támaszkodik a modell-transzformációkra és kódgenerálásra, ezek a technológiák fontos szerepet játszanak napjaink szoftver fejlesztési technológiájában. Ez idő szerint néhány speciális modell-transzformációs eszköz létezik már (például [2]), de egyik általunk ismert eszköz sem rendelkezik az összes szükséges funkcióval.

A modell-transzformációk igen fontos szerepet kapnak a modell verifikáció és validáció területén is, melyet elsősorban a megbízhatósági és biztonsági elemzés területén használnak. Ezen alkalmazási területeken a transzformációkat arra használják, hogy a rendszer modelljét egy matematikai modellé alakítsák, melyet a formális analízis eszközök használni tudnak, és az automatikus transzformáció segítségével elrejtsek a matematikai formalizmust a rendszer fejlesztője elől.

A napjainkban elérhető transzformációs eszközök nem különítik el a tervezési és végrehajtási idejű környezeteket, ezért a teljes transzformáció tervező rendszerre szükség van akkor is, ha a felhasználó csak futtatni szeretne egy transzformációt. Ebből következően a rendszerek telepítése, konfigurálása és használata igen bonyolult.

3. Transzformáció fejlesztés a VIATRA 2-ben

A modell-transzformációk előzőekben bemutatott lehetséges alkalmazási körei bizonyos követelményeket támasztanak egy általános modell-transzformáció fejlesztő keretrendszerrel szemben. Az eszköznek támogatnia kell a transzformációk és kódgeneráló modulok definícióját, fejlesztését, tesztelését és javítását.

A modell-transzformációkat sok különböző leírási móddal lehet definiálni [7]. Egyik legelterjedtebb és általunk is használt módszer a gráf transzformációs megadás. Ebben a formalizmusban az elemi modell manipulációs lépéseket gráfminták (részgráfok) formájában adják meg. Egy szabálynak két oldala van, a baloldali mintára illeszkedő részt kell keresni a modellben, és azt a jobboldalival

helyettesíteni. Ez a formalizmus vizuális formában ábrázolja a szabályokat, mely könnyen érthető a transzformációs technológia mélyebb ismerete nélkül is.

Az elemi gráf transzformációs szabályok nem definiálják a végrehajtás vezérlési szerkezetét, ezért rendszerünkben az absztrakt állapotgépek (Abstract State Machines ASM) nyelvét használjuk [3]. Az a nyelv nagyon hasonló formalizmust kínál, mint az általános programozási nyelvek, de matematikailag precíz szemantikája van, ezáltal könnyen használható és könnyen ellenőrizhető is.

A modellek ábrázolásához szükség van egy metamodellezési környezetre is, mely egyszerre több különböző nyelv metamodeljét (definícióját) tudja kezelni, és precíz szemantikával rendelkezik. Rendszerünkben a VPM (Visual and Precise Metamodeling) [4] koncepciót használtuk erre a célra.

1. ábra: A transzformáció fejlesztés menete

Az 1. ábra szemlélteti a transzformáció fejlesztés menetét. A meta-modellek, gráf transzformációs szabályok és vezérlő programok importálásra kerülnek a Viatra keretrendszerbe. A transzformációk nyomkövetése és tesztelése a keretrendszeren belül történik, majd a kész transzformáció egy specifikus transzformációs modul formájában készül el, mely már független a VIATRA rendszertől.

A transzformációs modulok különböző platformokra és programozási nyelvekre generálhatóak, ezáltal az elkészül transzformációk könnyen integrálhatóak összetett fejlesztőrendszerekbe is. Az egyik legfontosabb célplatform az Eclipse Modeling Framework (EMF) [5], ami egy nyílt, általános célú modell ábrázolási környezet, mely egyre több kereskedelmi termék alapját is képezi.

4. A keretrendszer felépítése

A VIATRA 2 keretrendszer fejlesztésekor az egyik fő cél a könnyű használhatóság volt, ezért erősen modularizált architektúrát alakítottunk ki. A teljes keretrendszert egy létező (és igen széles körben használt) nyílt szoftverfejlesztő környezetbe, az Eclipse keretrendszerbe [6], integráltuk.

A VIATRA 2 keretrendszer dinamikusan bővíthető újabb modellezési nyelvek beolvasó moduljaival, modell-értelmezőkkel és forráskód formázó modulokkal. A modellezési nyelvek beolvasó moduljai új modellezési nyelvek kezelését teszik lehetővé, míg a modell-értelmezőkkel különböző modelleket lehet „futtatni” a rendszeren belül. Utóbbi lehetővé teszi például a dinamikus viselkedést leíró modellek (például állaptgépek) egyszerű szimulációját is.

A forráskód formázó modulok a kódgenerátorok által készített forráskódot formázzák a célnyelv speciális igényeinek megfelelően. Például Java nyelv esetén az egyes osztályokat automatikusan külön fájlokba helyezik el, a csomagstruktúrának megfelelő könyvtárakban. Ezen bővítések segítségével a későbbiekben is követni lehet a felhasználói igények esetleges változásait.

5. Összegzés

A VIATRA 2 keretrendszer komplett megoldást kínál modell-transzformációk vizuális tervezéssel támogatott fejlesztésére, tesztelésére és hibakeresésére. A transzformációkból önálló végrehajtott modulokat lehet generálni, ezáltal a VIATRA keretrendszerre nincs szükség futási időben. A VIATRA 2 keretrendszer meta-transzformációs képességeinek segítségével az MDA koncepció kiegészíthető modell-vezérelt megbízhatóság és teljesítmény analízissel, mely az MDA transzformációk elemzésén alapul.

Hivatkozások

- [1] Gy. Csértán, A. Pataricza, P. Harang, O. Dobán, G. Biros, A. Dancsecz, and F. Friedler. BPM based robust e-business application development. Proc. Dependable Computing - EDCC-4, 4th European Dependable Computing Conference, p. 32-43. Springer, Berlin etc., 2002.
- [2] IBM. Model Transformation Framework, 2004. <http://www.alphaworks.ibm.com/tech/mtf>
- [3] Egon Börger, Robert Stark, “Abstract State Machines A Method for High-Level System Design and Analysis”, ISBN 3-540-00702-4, Springer-Verlag Heidelberg, 2003
- [4] D. Varró, A. Pataricza, “VPM: Mathematics of Metamodeling is Metamodeling Mathematics”, in Journal of Software and Systems Modelling, Special Issue on UML 2002, Springer.
- [5] The Eclipse Community, EMF Resources URL: <http://download.eclipse.org/tools/emf/scripts/home.php>
- [6] The Eclipse Community, Eclipse portal. <http://www.eclipse.org>
- [7] Varró Dániel. „Automated Model Transformations for the Analysis of IT Systems”, PhD Thesis, Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2004.

Balogh András, doktorandusz

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: abalogh@mit.bme.hu