Új formalizmus eseményvezérelt gráftranszformációhoz

Bergmann Gábor

Abstract
Continuously maintaining the mapping between constantly changing software engineering models requires event-driven transformations. To address the deficiencies of previous solutions, the paper presents a new approach based on graph transformation for the definition of event-driven transformation rules.

Key words:
model based software engineering, event-driven model transformation, graph transformation

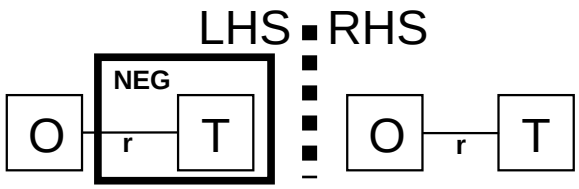
Összefoglalás
Az állandóan változó szoftvermérnöki modellek közötti leképezési viszony folyamatos fenntartásához eseményvezérelt transzformációk szükségesek. A korábbi ötletek hiányosságainak pótlására a dolgozat bemutat egy új, gráftranszformáció alapú megközelítést eseményvezérelt transzformációs szabályok definiálására.

Kulcsszavak:
modellvezérelt szoftvertervezés, eseményvezérelt modelltranszformáció, gráftranszformáció

1. Modelltranszformáció és gráftranszformáció

Az automatikus modelltranszformációk szerepe folyamatosan nő a modell alapú szoftvermérnöki folyamatokban (tervezés, eszközintegráció, formális verifikáció, tesztgenerálás, stb.), így igény van nagyméretű, ipari modellek hatékony lekérdezésére, manipulációjára, valamint egyik modell másikra történő leképezésére, egymásból való származtatására. [1]

A modelltranszformációk deklaratív, szabály alapú specifikációját teszi lehetővé a gráftranszformáció (GT) formalizmus, amely a modelleket típusos *gráf*ként ábrázolja, és a modell manipulációját *gráftranszformációs szabály*okkal írja le. A GT alapeleme a *gráfminta*, amely a gráfmodell bizonyos részeit azonosítja a struktúra és egyéb feltételek alapján. A két *gráfmintá*val megadott GT szabályok azt a lépést írják le, mely során az egyik mintára (LHS) illeszkedő gráfrészletet a másik minta (RHS) képével helyettesítjük. [2]



1. ábra. *Egyszerűsített objektum-relációs leképezés, gráftranszformációs szabályként*

Gráftranszformációs szabályokkal írható le például az úgynevezett objektum-relációs leképezés is, melynek egy szabályát mutatja be az 1. ábra. Az ábrán a csomópontok típusát nem, csak a nevüket és viszonyukat tüntettem fel; a „NEG” feliratú keret a tartalmának a nemlétezését írja elő (ún. negatív alkalmazási feltétel). A bal oldali (LHS) minta olyan osztály típusú O csomópontokra illeszkedik, amelyekhez nem tartozik adatbázistábla típusú T csomópont. A jobb oldalon (RHS) már tartozik hozzá egy T csomópont; a szabály végrehajtása egy T nélküli O-hoz létrehoz egy csatlakozó T csomópontot.

2. Eseményvezérelt gráftranszformáció

A folyamatosan változó, fejlődő modellek transzformációja újabb kérdéseket vet fel. A célmodell folyamatos szinkronban tartása a forrásmodell változásaival eseményvezérelt szabályvégrehajtást követel meg, melynek során a változási események idézik elő a transzformációs lépések végrehajtását. A legkézenfekvőbb ilyen esemény a modell elemeinek létrehozása vagy törlése. Ennél absztraktabb esemény fogalmak, magasabb szintű eseményvezérelt megoldások azonban nagymértékben támogathatják transzformációk készítését.

Egy ilyen magas szintű formalizmus a [3] cikkben bemutatott *gráftrigger*, amely a GT szabályok eseményvezérelt végrehajtását teszi lehetővé. A gráftrigger olyan GT szabály, amely akkor hajtódik végre, ha a bal oldali (LHS) gráfmintájának egy illeszkedése tetszőleges változtatás hatására megjelenik (vagy eltűnik) a gráfban. A gráfminták illeszkedéseinek megjelenése és eltűnése hatékonyan érzékelhető ún. inkrementális mintaillesztés [4] megoldásokkal.

Az 1. ábrán példaként bemutatott GT szabályt gráftriggerként értelmezve minden újonnan létrejövő O osztályt leképez egy T táblává, és a táblájukat valamilyen okból elvesztő, de régebb óta létező osztályok esetén is pótolja azt. Látható, hogy a magas szintű formalizmus a gráfminta megjelenését előidéző elemi változtatástól függetlenül biztosítja a szabály végrehajtását; ez bonyolultabb szabályok esetén természetesen még nagyobb előny a primitívebb eseményvezérelt rendszerekhez képest.

Az eseményvezérelt transzformációk tervezése során felgyűlt tapasztalatok szerint azonban a [3] által bevezetett formalizmus még mindig túlságosan korlátozott. Az esemény szűrőfeltételeként csak egyetlen minta megjelenése vagy eltűnése adható meg; se több illeszkedéshalmaz-változás együttes be(nem)következése, sem pedig a változás statikus környezete nem fejezhető ki a gráftriggerben. Jelen cikk célja egy általánosabb formalizmus kidolgozása, amely ezekre a hiányosságokra megoldást ad.

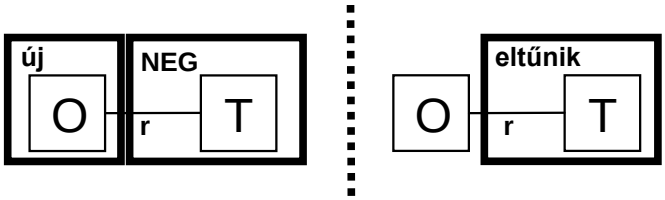
A gráftrigger formalizmus gyenge kifejezőereje az objektum-relációs leképezésen is szemléltethető: ha kétirányú transzformáció az elvárás, akkor például egy tábla törlésének hatására a forrásmodellbeli osztály törlendő. Az imént bemutatott gráftrigger ezt nem különbözteti meg attól az esettől, amikor egy új osztályt hozunk létre: mindkétszer tábla nélküli osztály áll elő, így kénytelen egyformán reagálni.

3. A változásminta formalizmus

A tárgyalt problémák megoldására a *változásminta* formalizmust javaslom. Az egyszerű gráfmintákkal ellentétben a változásminta egy változáson átesett gráf részeit azonosítja, a változások hatásait is figyelembe véve. A változásminta egyrészt a gráfmintáktól megszokott mintaelemeket (csomópontok és élek, negáció) tartalmaz, amelyek a változás *utáni* állapotra vonatkoznak. Másrészt hozzájuk kapcsolódóan előírhatja gráfminták illeszkedésének megjelenését és eltűnését, az eredeti gráftrigger fogalom végrehajtási feltételéhez hasonlóan.

A gráfminta-illeszkedések változásának detektálása ugyanolyan hatékonyan megvalósítható inkrementális mintaillesztés segítségével, mint az eredeti gráftriggerek esetén. Nincs szükség az elemi változások hatását külön kezelni; a magas szintű formalizmus továbbra is képes bonyolult minták illeszkedésváltozását egy eseményként kezelni. Ugyanakkor a változásminta a gráftriggernél nagyobb kifejezőerőt biztosít, hiszen egyszerre több gráfminta megjelenése és eltűnése is detektálható, és emellett a végső gráf elemeire is szabható feltétel (az esetleges változásaiktól függetlenül). Ennek a nagyobb kifejezőerőnek egy bizonyos értelemben vett teljessége is igazolható. A bizonyítást és a formalizmus precíz definícióját [5] mutatja be.

Példaként nézzük meg a kétirányú objektum-relációs leképezés megvalósítását a változásminták segítségével. Ha létrejön egy osztály és nincs hozzá tábla, akkor létrehozunk egy táblát reprezentáló csomópontot és összekötjük az osztállyal (1. szabály). Ha megszűnik egy osztály, akkor a vele összekötött táblát is törölni kell (2. szabály). Ha létrejön egy új tábla és még nincs osztályhoz kapcsolva, akkor létrehozunk egy hozzá kötött osztályt (3. szabály). Ha pedig törlődik egy tábla, akkor a vele összekötött osztályt is meg kell szűntetni (4. szabály). Látható, hogy mind az első, mind a negyedik szabály olyan helyzetben hajtandó végre, amikor megjelenik egy illeszkedése a táblával össze nem kötött osztályból álló gráfmintának; az eredeti gráftrigger formalizmussal nem lehetett (segédstruktúrák bevezetése nélkül) megkülönböztetni a két esetet, holott eltérően kell reagálni. Hasonló viszonyban van egymással a másik két szabály is. A 2. ábra bemutatja, hogy változásmintákkal miként választható szét a két eset. Az ábra első fele az 1. szabály feltételét jeleníti meg, a második fele a 4.szabályét; az „új” feliratú keret egy megjelenő gráfminta-illeszkedést, az „eltűnik” keret egy megszűnő gráfminta-illeszkedést jelöl.



2. ábra. *Két változásminta kétirányú objektum-relációs leképezéshez (1. szabály, 4. szabály)*

A változásmintákkal vezérelt szabályvégrehajtás a gráftriggerek általánosításának tekinthető; az egyetlen mintailleszkedés-megjelenést vagy eltűnést tartalmazó, statikus elem nélküli változásminta

esetén megfelel a gráftrigger formalizmusnak. Ugyanakkor illeszkedéshalmaz-változás nélkül, csupán statikus elemekkel hagyományos GT szabállyá fajul el. Így látható, hogy két korábbi megközelítés közös általánosításáról van szó.

7. Összefoglaló

A szoftvertervezésben használt modellek közötti transzformációkat deklaratív, magas szintű, szabály alapú transzformációs nyelven célszerű definiálni. A fejlődő modellek változásai alapján a leképezés állandó karbantartásához eseményvezérelt transzformáció szükséges. Az egyszerűbb esemény formalizmusok és a gráftranszformáció közös általánosításaként a dolgozat bemutatta a változásminták nyelvét. A változásminta egy új megközelítés eseményvezérelt transzformációs szabályok definiálására, és a korábban publikált gráftrigger formalizmusnál nagyobb kifejezőerejű, ám hozzá hasonlóan magas szintű és hatékonyan megvalósítható.

Jövőbeli kutatási feladat, hogy milyen módokon kezelhető az az eset, amikor egyszerre több eseményvezérelt szabály aktiválódik; megoldás lehet valamilyen prioritásos rendszer, az egyidejű végrehajtás konfliktuskezeléssel, vagy a kézi beavatkozás. Nem esett szó a modellbéli attribútumok változásairól sem. Szintén jövőbeli feladat a kidolgozott megoldás megvalósítása.

Irodalom

[1] The Object Management Group: *Technical Guide to Model Driven Architecture: The MDA Guide v1.0.1*, http://www.omg.org, 2003, 3-2—3-13 oldal.

[2] H.Ehrig, G.Engels, H.-J.Kreowski, and G.Rozenberg, Eds.: *Handbook of Graph Grammars and Computing by Graph Transformation*,vol.2, World Scientific, River Edge, 1999.

[3] Ráth I., Bergmann G., Ökrös A., Varró D.: *Live model transformations driven by incremental pattern matching*, ICMT 2008, LNCS Vol. 5063. Springer, Berlin, 2008, 107-121. oldal.

[4] Bergmann G., Ökrös A., Ráth I., Varró D., Varró G.: *Incremental pattern matching in the VIATRA model transformation system*, GraMoT 2008. ACM, New York, 2008, 25-32. oldal.

[5] Bergmann G.: *Contextual Graph Triggers*, Proceedings of the 17th PhD Mini-Symposium of the Department of Measurement and Information Systems. BME, Budapest, 2010, 22-25. oldal.

Bergmann Gábor, doktorandusz

Budapesti Műszaki és Gazdaságtudományi Egyetem, Villamosmérnöki és Informatikai Kar,

Méréstechnika és Információs Rendszerek Tanszék

H-1117, Magyarország, Budapest, Magyar tudósok krt. 2.

Tel: +36-1-463-3579, E-mail: bergmann@mit.bme.hu