FIATAL MŰSZAKIAK TUDOMÁNYOS ÜLÉSSZAKA

Kolozsvár, 2004. március 26-27.

MEGBÍZHATÓSÁGI MODELLEK AUTOMATIKUS GENERÁLÁSA GRÁFTRANSZFORMÁCIÓVAL

Domokos Péter, Majzik István, Varró Dániel

Abstract

In the paper, we introduce a transformation to derive a stochastic dependability model from an UML class diagram representing the system architecture. The UML model is extended with the dependability attributes of the components. The system level dependability attributes are computed using the model derived by the transformation.

Összefoglaló

A dolgozatban bemutatunk egy transzformációt és annak implementációját, amelynek segítségével a rendszerarchitektúrát leíró, a komponensek megbízhatósági jellemzőivel kiegészített UML osztálydiagramokból automatikusan egy időzített Petri hálót hozunk létre. A Petri háló segítségével származtathatóak a rendszerszintű megbízhatósági jellemzők.

Bevezetés

Nagy megbízhatóságú rendszerek tervezése során fontos, hogy a megbízhatósági jellemzők már a tervezési folyamat során megbecsülhetőek, meghatározhatóak legyenek, hiszen ennek függvénye lehet, hogy egy adott architektúra megfelel-e a rendszerrel szemben támasztott követelményeknek. Ilyen megbízhatósági jellemzők például a rendelkezésre állás (annak valószínűsége, hogy a rendszer az adott időpillanatban helyes szolgáltatást nyújt), ill. a megbízhatóság (annak valószínűsége, hogy a rendszer az adott időpillanatban helyes szolgáltatást nyújt, feltéve, hogy nem javítjuk a meghibásodásokat).

A megbízhatósági modellek segítségével a tervezési fázis során összevethetőek különböző architektúrák és megoldások, és így kiválasztható a tervező céljainak leginkább megfelelő változat. A megbízhatósági modell egy matematikai modell, amely leírja az egyes komponensek meghibásodási és javítási folyamatait, valamint a komponensek közötti hibaterjedést. A modell alapján így meghatározható, hogy a komponens hibák hatására mikor adódik rendszerszintű hibajelenség.

Célkitűzésünk a rendszer architektúra modelljéből a megbízhatósági modell előállítás. A rendszer architektúrát UML-ben írjuk le. Az UML (Unified Modeling Language) modellezési nyelv a teljes rendszerfejlesztési folyamatot támogatja: mind a követelményelemzést segítő viselkedési modellek (pl. aktivitás diagramok, állapotgépek), mind a rendszertervezést segítő strukturális modellek (pl. osztálydiagramok, objektumdiagramok), mind pedig az implementációt segítő modellek (pl. állapotgépek) leírhatóak benne.

Ezen modellek közül mi az architektúrát leíró strukturális modellekre, azon belül is az osztálydiagramokra koncentrálunk. Az osztálydiagramok általánosságban jelenítik meg a rendszerben megjelenő objektumokat és a köztük lévő lehetséges kapcsolatokat.

A mi megközelítésünk szerint mind a hardvert, mint a szoftvert UML-ben modellezzük. A tervezési folyamat korai fázisaiban tipikusan az architektúrális modellek készülnek el (mint például osztály- és objektumdiagramok), az implementációs modellek (mint például állapotgépek) csak később készülnek el. Ezen architektúrális modelleket kiegészítjük ún. címkézett értékekkel, amelyek az adott komponens megbízhatósági jellemzőit adják meg, ha azok ismertek (ilyen jellemzők pl. a hiba előfordulási valószínűsége, a hiba megjelenésének késleltetése, az állandósult hibák aránya és a javítás késleltetése).

Többféle megbízhatósági modell formalizmus használatos, ilyenek a Petri hálók, a Markov láncok és a sztochasztikus processz algebra. Mi a Petri hálókat használjuk, amelyeknek létezik grafikus reprezentációjuk és alkalmasak konkurens rendszerek modellezésére is.

Az UML osztálydiagramokon szerepelnek a rendszert alkotó komponensek, az ezek közötti kapcsolatok, továbbá azon komponenseknél, amelyeknél ez rendelkezésre áll, a megbízhatósági paraméterek. Az osztályok közötti kapcsolatokat, mint hibaterjesztési utakat figyelembe véve, egy időzített Petri hálót állítunk elő a hibaterjedés modellezésére, amelynek segítségével a rendszer szintű megbízhatósági paraméterek származtathatóak.

Ebben a cikkben az UML osztálydiagramok időzített Petri hálóká történő transzformációjának elméleti hátterét [1], ill. egy implementációját mutatjuk be.

UML osztálydiagramok transzformációja időzített Petri hálóká

Az osztálydiagramokat két lépésben transzformáljuk Petri hálóká. A rendszerszintű megbízhatósági paraméterek kiszámítására csak az egyes komponensek jellemzőire, illetve a köztük lévő hibaterjedési útra van szükségünk. Mivel a kiindulási modellünk ennél lényegesen több információt tartalmaz, az első lépés egy *absztrakciós lépés*, amelynek során előállítunk egy *köztes modellt*, amely csak a komponensek paramétereit, ill. a köztük lévő hibaterjesztési utakat tartalmazza. Második lépésként ezt a köztes modellt transzformáljuk *időzített Petri hálóvá*.

A köztes modell a rendszer komponenseit reprezentáló *csomópontokból*, az ezekhez csatolt *megbízhatósági paraméterekből*, és a csomópontok közti hibaterjesztést leíró „*használja ...*” (U) relációkból áll. A köztes modell előállítása során a következő leképezéseket hajtjuk végre:

- Az *osztályokat csomópontokba* képezzük le, amely tartalmazza az osztályhoz rendelt megbízhatósági jellemzőket.
- Egy *asszociáció* két osztály között kétirányú hibaterjesztési utat definiál, ezért ezt *két U relációra* képezzük le, amelyek ellentétes irányban mutatnak a megfelelő két csomópont között.
- Egy *aggregáció* egy egyirányú hibaterjesztési utat ír le, ezért ezt egyetlen *U relációra* képezzük le.

- Az öröklés nem definiál explicit módon hibaterjesztési utat, ezért ezt nem képezzük le a köztes modellbe, a tartalmát azonban figyelembe vesszük (azaz, ha két osztály ősei között lehet egy asszociáció, akkor köztük is lehet).

A köztes modell előállítás után abból egy időzített Petri hálót állítunk elő. Ennek során minden egyes csomópontnak, és minden egyes U relációnak egy-egy alhálót feleltetünk meg, majd ezeket a megfelelő módon összekötjük. Az alhálók definíciója megtalálható az [1] irodalomban.

Az 1. (a) ábrán példaként egy egyszerű osztálydiagram látható, amely mindössze két osztályt tartalmaz, és az egyik aggregálja a másikat. Az 1. (b) ábrán látható a fentiek alapján származtatott köztes modell, míg az 1. (c) ábrán a megfelelő Petri háló. A Petri háló három alhálóból áll: egy-egy alháló felel meg az A és a B csomópontoknak, egy harmadik pedig a köztük lévő U relációnak. Az alhálók és a többi modell elemei közti megfeleltetést a pontozott vonal mutatja.

A csomópontoknak megfelelő alháló modellezi a meghibásodási és a hibajavítási folyamatot. A H helyen lévő token jelképezi a rendszer hibátlan működését, míg az E helyen lévő token a rendszer hibás állapotát jelöli. Az F helyen lévő token jelenti, hogy a rendszer hibás állapotának hatására kívülről is látható hibás szolgáltatás állt elő. Az erre a helyre a tokeneket eljuttató forrás tranzíciót egy *őrfeltétel* védi, hogy csak akkor kerülhessen ide token, ha a rendszer valóban hibás állapotban van.

1. ábra: UML osztálydiagram (a), köztes modell (b), időzített Petri háló (c)

Az alháló többi része a hibás állapotjavítását célozza, míg a harmadik alháló a két csomópont közti hibaterjesztést modellezi.

Implementáció

Mindkét transzformáció (osztálydiagramból köztes modell, ill. köztes modellből Petri háló) implementációjára a *gráfranzformációt* választottuk, mint formalizmust. A gráfranzformáció a Chomsky-féle nyelvtanok gráfmintákra történő kiterjesztésének tekinthető, amely a szabály bal oldalán található gráfminta egy illeszkedését cseréli le a szabály jobb oldalán található gráfmintára.

Az osztálydiagramból a köztes modell előállítását végző transzformáció 16 gráfranzformációs szabályból áll. Az első lépés az eredeti modellben megtalálható *implicit* információk *explicit* információkká alakítása: az öröklések kezelése. Azaz, ha a B osztály az A osztály leszármazottja, továbbá a C osztály a B leszármazottja, akkor C is leszármazottja. Hasonlóan járunk el az attribútumok és asszociációk esetén is: ha egy attribútum vagy egy asszociáció hozzátartozik az A osztályhoz, akkor azt föl vesszük a B és C osztályhoz is.

A következő lépés az osztályok példányosítása, azaz a köztes modell csomópontjainak a létrehozása. Ennél a lépésnél különös figyelmet kell fordítani azon osztályokra, amelyekből valamely asszociáció több példány létrehozását írja elő. Éppen ezért ezt a lépést két allépésre bontjuk: először példányosítjuk azokat az osztályokat, amelyekből csak egyetlen példányt kell létrehoznunk, majd azokat, amelyekből többet is. Utóbbiak esetén először létrehozunk egy példányt az adott osztályból, majd minden, hozzá tartozó asszociációnál csökkentjük a multiplicitást. Ily módon pontosan annyi példányt hozunk létre az adott osztályból, amennyire minimálisan szükségünk van.

Miután létrejöttek a köztes modell csomópontjai, a köztük lévő kapcsolatoknak megfelelő U relációkat kell létrehoznunk, és összekötnünk a csomópontokkal.

A köztes modellből a Petri hálót előállító transzformációt 5 szabály valósítja meg, azonban ezek már lényegesen komplexebb mintákra illeszkednek. A transzformáció során minden egyes csomópontnak, és minden egyes U relációnak egy alhálót feleltetünk meg, majd ezek között létrehozuk a megfelelő összeköttetéseket (éleket).

A Petri háló megoldása, azaz a rendszerszintű H ill. F helyeken a token tartózkodás valószínűségének számítása adja meg a rendelkezésre állási jellemzőket. Ha a modellben nincs javítás, akkor megbízhatósági értéket kapunk.

Összegzés

A fent leírt transzformációt a VIATRA (VIsual Automated model TRAnsformation) [2] modelltranszformációs rendszerben implementáltam és sikeresen alkalmaztam kisebb mintamodellekre.

Hivatkozások

- [1] I. Majzik, A. Pataricza, and A. Bondavalli. Stochastic dependability analysis of system architecture based on UML models. In Rogerio de Lemos, Cristina Gacek, and Alexander Romanovsky, editors, *Architecting Dependable Systems*, volume LNCS-2677, pages 219-244. Springer, 2003.
- [2] Gy. Csértán, G. Huszerl, I. Majzik, Zs. Pap, A. Pataricza and D. Varró. VIATRA: Visual automated model transformations for formal verification and validation of UML models. In Proc. ASE 2002: 17th IEEE International Conference on Automated Software Engineering, Edinburgh, UK, September 23-27 2002.

Domokos Péter
doktorandusz
e-mail: pdomokos@mit.bme.hu

Méréstechnika és Információs Rendszerek Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem
H-1521, Budapest,
Magyar tudósok körútja 2.
fax: +36 1 463 2667