BPEL 2.0 MUNKAFOLYAMATOK FORMÁLIS VERIFIKÁCIÓJA[1]

Hegedüs Ábel
Budapesti Müszaki és Gazdaságtudományi Egyetem
Méréstechnikai és Információs Rendszerek Tanszék

Absztrakt. *A cikkben BPEL 2.0 munkafolyamatok modellellenőrzésen alapuló formális verifikációjára dolgoztam ki egy módszert. A folyamatokból tervezési időben, automatikus modelltranszformációk segítségével precíz leírás készül tranzíciós rendszerek formájában, amelyen kiterjedt vizsgálatok végezhetők az állapottér szisztematikus és kimerítő bejárását végző modellellenőrző eszközökkel.*

Bevezetés

Napjainkban a szolgáltatás orientált architektúra (SOA) elképzelés központi szerepet tölt be a komplex, nagyvállalati elosztott rendszerek tervezésében. Ezen rendszerek olyan autonóm szolgáltatásokból épülnek fel, amelyek precíz interfészekkel rendelkeznek. Az egyedi szolgáltatásokra épülő komplex üzleti folyamatok modellezésére szabványos munkafolyamat-leíró nyelvek léteznek, például a Business Process Execution Language for Web Services (BPEL) [1].

Motiváció. A BPEL segítségével létrehozott üzleti folyamatokat gyakran használják üzleti partnerek közötti automatikus adatcserében (Business-to-Business) és vállalati alkalmazásintegrációban (Enterprise Application Integration). Mivel ezek a folyamatok valósítják meg az együttműködést a szereplők között, minőségük kritikus fontosságú, hiszen bármilyen rendellenes működésnek jelentős, negatív pénzügyi hatása lehet. A hibák előfordulási esélyeinek minimalizálásához a tervezőknek és elemzőknek olyan eszközökre van szükségük, amelyek garantálják az üzleti folyamatok megfelelő viselkedését.

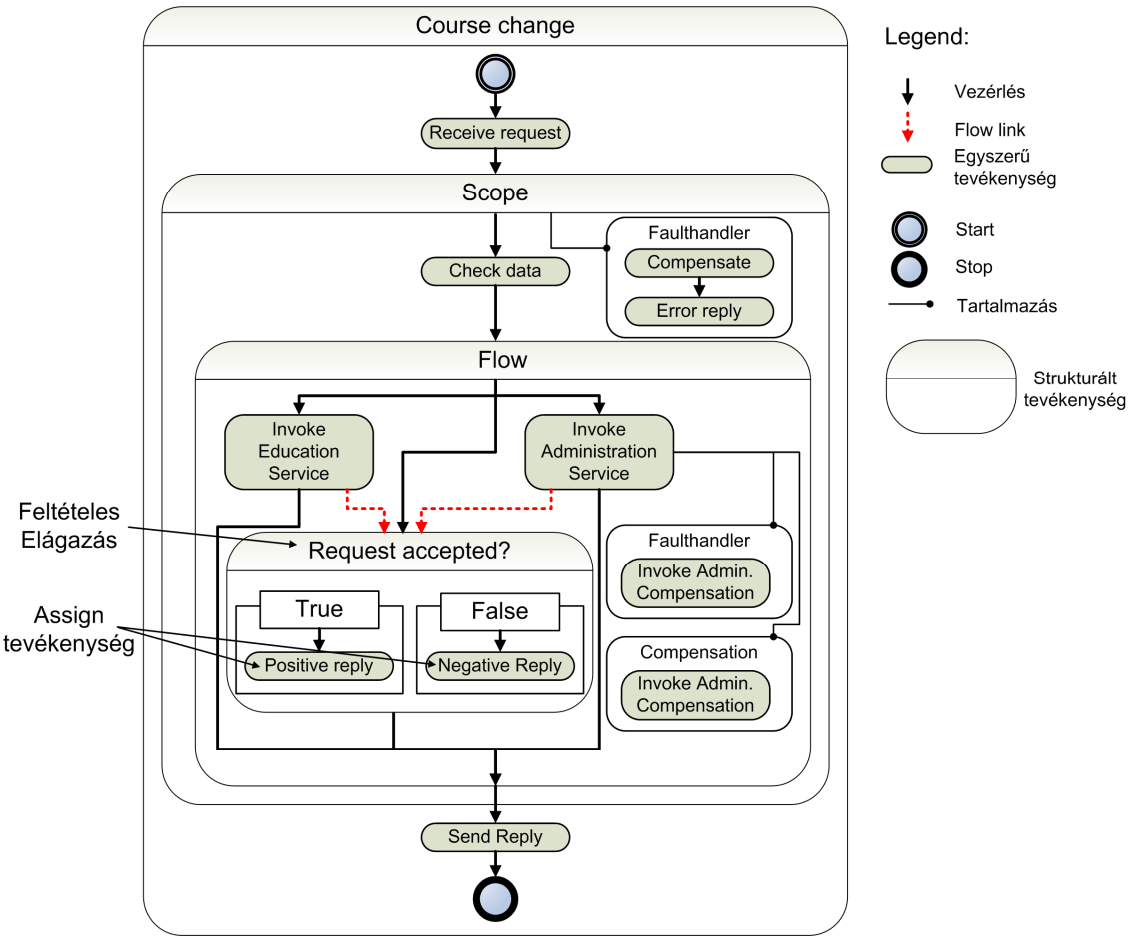
Célkitűzés. Kiterjedt múltbeli kutatások bizonyították a modellellenőrzés és más formális módszerek használhatóságát az üzleti folyamatok minőségének javításában. A létező módszerek azonban csak a munkafolyamatok elemkészletének egy részhalmazát képesek vizsgálni. Ezért szükség van olyan módszerekre, amelyek a *kiforrottabb BPEL 2.0 szabvány* alapján készített munkafolyamatokat lehetőleg minden nyelvi elemében támogatják, kiterjesztve így a [2]–ban a BPEL szabvány 1.1-es változatára korábban elvégzett vizsgálatokat többek között a párhuzamos részfeladatok közötti *részleges sorrendezés* működésének modellezésével.

Megközelítés. A javasolt módszer alkalmas BPEL 2.0 munkafolyamatok működésének formális modellezésére és helyességellenőrzésére. A vizsgált folyamatból egy modelltranszformációs keretrendszer tranzíciós rendszer leírást hoz létre. Ezen a matematikailag precíz struktúrán automatikusan végezhető el a temporális logikai kifejezések alakjában megadott kritériumok szisztematikus, kimerítő állapottérbejáráson alapuló ellenőrzése. A modellellenőrző keretrendszer által szolgáltatott ellenpélda visszavetíthető az eredeti BPEL munkafolyamatra, így megadva a kiemelt lefutást.

[1] Ezt a munkát részben a SENSORIA Európai Uniós kutatási projekt (IST-3-016004) és a ResilTech támogatta.

Esettanulmány

A dolgozatban módszerünk bemutatásához egy esettanulmányt fogok felhasználni. A mintapélda a SENSORIA projekt elosztott tanulmányi és kurzus menedzsment rendszert leíró esettanulmányának (eLearning Case Study [3]) egy részfeladatát használja fel. A SENSORIA projekt célja a modellalapú fejlesztésre, ellenőrzésre használható módszerek kifejlesztése.



1. ábra: Az esettanulmány főfolyamata

Egy tanulmányi rendszer egyik fontos feladata a hallgatók kurzusjelentkezéseinek kezelése, amely összetett feladatot öt együttműködő komponens végez el. Abban az esetben, ha egy hallgató egy kurzust nem a saját egyetemén kíván teljesíteni, akkor a két egyetem tanulmányi rendszereinek kooperációjára van szükség, melynek során a kurzus áthallgatási kérelmet elbírálják és tárolják. A folyamat a kérelem fogadása után ellenőrzi az adatokat, majd párhuzamosan kapcsolatba lép a tanulmányi és adminisztrációs rendszerekkel. Az adminisztrációs rendszer válaszának függvényében állítja össze a válaszüzenetet, továbbá e rendszer hibája esetén biztosítja az elvégzett műveletek visszagörgetését kompenzáció segítségével. A folyamat működése a végrehajtáshoz szükséges részleteket elfedő illusztráción látható az 1. ábrán.

A cikkben részletezett módszer felhasználásával verifikálom a mintapélda folyamatát. Egyrészt az adatok tárolására használt változók helyes használatát vizsgálom, másrészt ellenőrzöm, hogy a folyamatok minden esetben véget érnek-e.

BPEL 2.0 munkafolyamatok modellezése

A munkafolyamatokat gyakran ábrázolják olyan véges automata formalizmus használatával, amelyet modellellenőrző rendszerek bemeneti nyelvére lehet fordítani, majd az állapottér bejárásával bizonyíthatók a folyamatra megadott tulajdonságok. A következőkben a [2] által bevezetett módszer BPEL 2.0 munkafolyamatokra történő kiterjesztése és az összeköttetések támogatásának bevezetése kerül részletezésre, melynek technikai részleteit [4] tartalmazza.

A BPEL 2.0 szabványban bevezetésre került a változó inicializáció, és az integrált adatmanipuláció lehetősége. Lehetőség van a változók definiálásakor egy kezdőérték megadására, ezzel csökkenthető a változóolvasáskor előforduló hibák száma. Az esettanulmányban például a tanulmányi rendszer szolgáltatásának meghívásakor (**Invoke Education Service**) az üzenet egyes részei már a folyamat kezdetekor megadhatók. Szintén újdonság, hogy a változók közötti adatmásoláshoz nem szükséges külön *Assign* típusú tevékenység használata (ilyen például a **Positive Reply** tevékenység), hanem lehetőség van az adatmanipuláció elvégzésére az adatot használó tevékenységeken belül is (például **Check Data** bemeneti paramétereinek és eredményének elérése). Ezen lehetőségek modellezéséhez az eredeti módszer [2] adatmanipulációjának kiterjesztésére volt szükség.

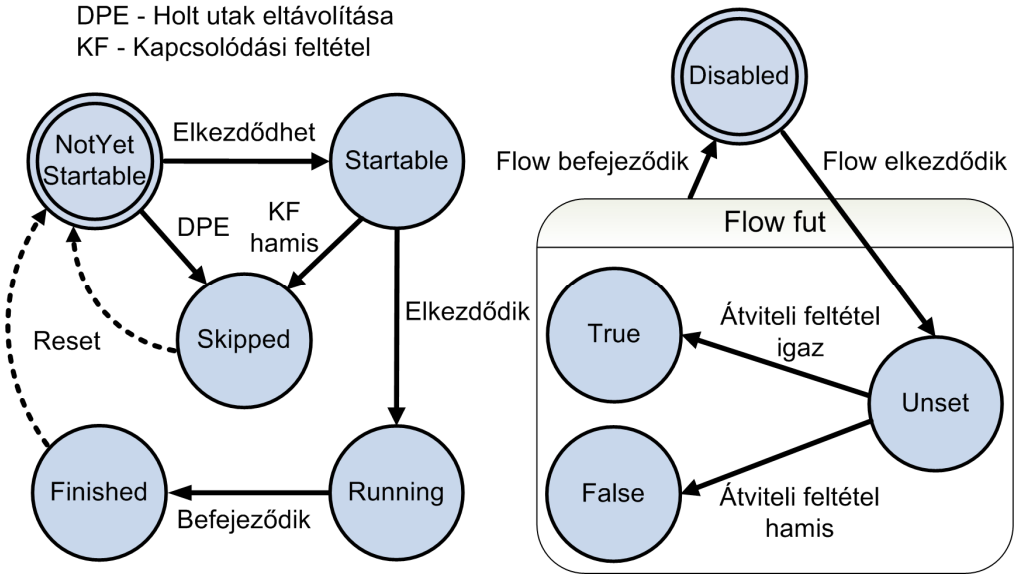
Újdonság a BPEL 2.0 szabványban a hibaterjesztés lehetősége, és az explicit megszakításkezelő megjelenése. Az előbbi használható olyan esetekben, ha egy lekezelt hibát változtatás nélkül a felsőbb szintű vezérlési folyamat alapegység (**Scope)** felé tovább kell terjeszteni, ennek modellezése egy egyszerű tevékenységhez hasonló, kiegészítve a hiba paramétereinek továbbadásával. Az utóbbi alkalmazható **Scope**-ok futásának *szabályozott* leállítására, olyan esetekben, amikor explicit megszakításra van szükség. A megszakításkezelőt, a hiba- és eseménykezelőkhöz hasonlóan, egy strukturált tevékenységként modellezzem, amelynek lefutása a **Scope** egy adott állapota esetén lehetséges.

Flow linkek modellezése

A BPEL lehetőséget ad arra, hogy a párhuzamos végrehajtású ágak tevékenységei között sorrendi összeköttetéseket adhassunk meg. Az 1. ábrán szaggatott nyilak jelölik az esettanulmányban definiált linkeket. Minden linkhez megadható egy átviteli feltétel és minden cél tevékenységhez egy kapcsolódási feltétel. Ennek következtében a linkek a sorrendi összeköttetésen túl engedélyezési információt is hordozhatnak. Ezért a linkek alkalmazása esetén előfordulhat, hogy egy tevékenység nem hajtódik végre. Ennek a működésnek a modellezéséhez szükség van a tevékenységekhez tartozó állapotváltozók értékkészletének kibővítésére. Felveszem a kihagyott (*Skipped*) állapotot, amely azt jelzi, hogy az adott tevékenység nem hajtódott végre. A 2. ábrán látható a kibővített állapottér egy példa tevékenység esetén.

Linkek állapottere

Az összeköttetések modellezésekor figyelembe kell venni azokat az állapotokat és állapotátmeneteket, amelyek a végrehajtás során előfordulhatnak. A link állapota a munkafolyamat futásának kezdetén *letiltott* (*disabled*). A linket definiáló flow tevékenység elindulásakor az állapot *határozatlan* (*unset*) lesz. A link kezdőpontját képző tevékenység lefutása után (például az adminisztrációs rendszer válaszüzenete megérkezésekor, **Invoke Administration Service**) a link *igaz* (*true*) vagy *hamis* (*false*) értéket vesz fel. A flow tevékenység befejeződésekor a definiált linkek állapota ismét letiltottra vált. A 2. ábra illusztrálja a link állapotterét.



2. ábra: Tevékenységek és összeköttetések állapottere

Holt utak eltávolításának modellezése

A folyamatokban előfordulhatnak olyan tevékenységek, amelyek bizonyos lefutások esetén nem kerülnek végrehajtásra. A kihagyott tevékenységek kimenő linkjei holtpontot okozhatnak a folyamatban, mert nem kapnak határozott értéket. Ennek elkerülésére definiálja a szabvány a *holt utak eltávolítása* algoritmust, melynek működése két fázisra osztható: az egyik a nem végrehajtható tevékenységek megtalálása, a másik a kimenő linkek kezelése.

A következő helyzetekben lehetséges, hogy egy tevékenység nem kerül végrehajtásra, ezeket az eseteket kezelő tranzíciókat kell modellezni:

(1) kapcsolódási feltétel hamis értékű és a hiba elnyomásra kerül, (2) feltételes elágazás nem végrehajtott ágaiban helyezkedik el, (3) **pick** tevékenység nem végrehajtott ágaiban helyezkedik el, (4) **scope** tevékenységen belül egy hiba keletkezési helye után vannak, (5) nem végrehajtott hibakezelők és megszakítás-kezelő tartalmazza.

A felsorolt esetekben a kihagyott tevékenységek kimenő linkjei hamis értéket kapnak, így megakadályozva a holtpont kialakulását.

BPEL 2.0 munkafolyamatok analízise

A bemutatott modellezési módszer használata lehetőséget nyújt önálló folyamatok modellellenőrzésére. Az ellenőrzés segítségével mind általános, mind folyamat-specifikus kritériumok teljesülése vizsgálható egy adott munkafolyamat esetén. A modellellenőrzéshez a vizsgált folyamat működését modellező tranzíciós rendszer leírást kiegészítjük a követelmények formális definíciójával. Ezek a definíciók lineáris temporális logikai (LTL) kifejezések, amelyek igazságtartalmát bizonyítja a modellellenőrző keretrendszer. Az általam használt rendszer a Symbolic Analysis Laboratory (SAL) [5], amely tranzíciós rendszer leírással megadott modellek ellenőrzésére használható. A SAL szimbolikus modellellenőrző eszköze a definiált LTL kifejezéseket kísérli meg bizonyítani. Ha a kifejezés hamis, akkor egy, az illegális állapotba vezető (kifejezést sértő) tranzíciósorozatot is szolgáltat az ellenőrző (ezt nevezzük ellenpéldának). Az ellenpélda a folyamat egy konkrét lefutását adja meg, így a meghibásodást előidéző körülmények visszavezethetők az eredeti folyamatleírásra.

Az önálló folyamatokra megadott néhány általános kritérium a következő:

- **Nincs inicializálatlan változó olvasás**. $G(var_{reply} \neq read)$ A vizsgált munkafolyamat végrehajtásakor nem következhet be olyan helyzet, amikor egy változó tartalmát felhasználja, mielőtt a változót írta volna (például a **Send Reply** végrehajtásakor). A folyamat teljesíti ezt a kritériumot, tehát a *read* változót soha nem olvassa a folyamat előbb, mint írná.

- **Nincs soha nem olvasott változó**. $\neg(F(var_{check} = writtenAndRead))$ Teljesítése esetén kimondható, hogy nincs felesleges változó a rendszerben (ebben az esetben a **Check Data** eredményét tároló változót vizsgálom). Ez azt jelenti, hogy minden változót módosít a folyamat és a változó tartalma felhasználásra kerül. Az a munkafolyamat, amelyben a tulajdonság nem teljesül, tartalmaz felesleges változót, amely eltávolítható. A kritérium a konkrét folyamaton igaz.

- **A folyamat futása mindig befejeződik**. $G(P = startable \rightarrow F(P = finished))$ Kritérium fennállása biztosítja, hogy a folyamat minden esetben véget ér. Azt az állapotot, ahonnan egy folyamat nem képes továbblépni, holtpontnak (deadlock) nevezzük. Az ellenőrzés a holtpont létezésének ténye mellett a folyamat holtpontra jutását okozó lefutást is megadja. Az esettanulmány teljesíti ezt a kritériumot.

Kapcsolódó munkák

Van der Aalst és Verbeek megoldása [6] le tudja képezni a lehetséges tevékenységek nagy részét, így a **Linket** is, de nincs **Scope** és hibakezelés. A BPEL szemantikát szinte teljes egészében lefedi [7]. Egy másik figyelemreméltó megközelítés [8], amely a folyamatokat hierarchikus, színezett Petri-hálókra képezi le. A Petri-hálókkal történő modellezésre kialakított eszköz [9] jelenleg mindössze háromfajta, meglehetősen korlátozott ellenőrzést támogat.

Az eddig ismertetett munkák mindegyike a BPEL 1.1-es szabványának megfelelő BPEL folyamatok modellezését tűzte ki célul. A közelmúltban megjelent 2.0-ás szabványban található újdonságokat is lefedő megoldást nyújt [10]. Ez a módszer [7] egyfajta kiegészítésének tekinthető, így a modellezési eljárás azonos.

Összefoglalás és kitekintés

A munkafolyamatok – ebben az esetben a BPEL 2.0 szabványnak megfelelő munkafolyamatok – leírása nem alkalmas formális verifikáció elvégzésére. Az általam kidolgozott módszer a munkafolyamatok formális verifikációját teszi lehetővé, a BPEL legújabb 2.0-s szabványban felhasználható elemek jelentős részét képes modellezni. Kiemelt jelentőségű az összeköttetések és a holt utak eltávolítása algoritmus támogatása. A dolgozatban használt esettanulmány folyamatain ellenőrzéseket végeztem és a módszer a SENSORIA project Sensoria Development Environmentbe történő integráltam.

A megközelítés kiterjesztésére többféle lehetőség van. A részletezett módszert felhasználva lehetőség nyílik kooperáló üzleti folyamatok ellenőrzésére is. Az ellenőrzés során ellenpélda-vezérelt iteratív modellfinomítást alkalmaztunk. Ennek használhatóságát és jó hatásfokát bizonyítottuk a [4,11] publikációkban. Terveim között szerepel egy grafikus felület létrehozása a követelményspecifikációk megadására és az eredmények megjelenítésére. Továbbá doménspecifikus hibamodellek használata is növelné a módszer használhatóságát.

Irodalomjegyzék

[1] Alves, A. és mások: Web services business process execution language version 2.0 (OASIS standard). WS-BPEL TC OASIS, 2007.

[2] Kovács, M., Varró, D. és Gönczy, L.: Formal Analysis of BPEL Workflows with Compensation by Model Checking. International Journal of Computer Systems Science and Engineering, 2008

[3] Hölzl, M.: SENSORIA Deliverable D8.4.a: University management and e-learning case study: Requirements modelling and analysis of selected scenarios. Tech. rep., 2007.

[4] Bende, T. és Hegedüs, Á.: BPEL 2.0 alapú Munkafolyamatok Kooperációjának Formális Verifikációja, Budapest, TDK dolgozat, 2008.

[5] Shankar, N.: Symbolic Analysis of Transition Systems. Monte Verità, Computer Science, Springer-Verlag, 2000. 287–302.

[6] Verbeek, H. M. W. és Aalst, W.: Analyzing BPEL processes using petri nets, Miami, Florida International University, 2005. 59–78.

[7] Stahl, C.: A Petri Net Semantics for BPEL. Berlin, Informatik-Berichte 188, Humboldt-Universität zu Berlin, 2005.

[8] Yang, Y. és mások: Verifying web services composition: A transformation-based approach. PDCAT, IEEE Computer Society, 2005. 546–548.

[9] Ouyang, C. és mások: WofBPEL: A Tool for Automated Analysis of BPEL Processes. ICSOC, 2005. 484–489.

[10] Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0 and its compiler BPEL2oWFN. Berlin, Informatik-Berichte 212, Humboldt-Universität zu Berlin, 2007.

[11] Bende, T., Hegedüs, Á., Kovács, M., Varró, D. és Gönczy, L.: Verification of BPEL Process Cooperations by Iteratively Refined Abstraction. Business Process Management Conference, 2009. (bírálat alatt)