Budapesti Műszaki és Gazdaságtudományi Egyetem
Irányítástechnika és Informatika Tanszék

Becsülhető válaszidejű interoperábilis WS-* alkalmazások modellvezérelt kialakítása

---

Model Driven Construction of Interoperable WS-* Applications with Predictable Response Times

A doktori értekezés tézisei

Simon Balázs

Témavezető:

Dr. Kondorosi Károly
címzetes egyetemi tanár

Budapest, 2015

Tartalomjegyzék

Összefoglaló

A webszolgáltatások platformfüggetlen elosztott kommunikációt biztosítanak azáltal, hogy minden kapcsolódó szabvány XML-re épül. Az interfészleíró nyelv (WSDL), a kommunikációs protokoll (SOAP), valamint ennek kiegészítései (WS-* protokollok) is mind XML-en alapulnak. Összetett webszolgáltatások is készíthetők az üzleti folyamatok leírására szolgáló BPEL nyelv segítségével, amely ugyancsak az XML formátumra épít. Az XML legfőbb előnye a platformfüggetlenség. A szabványokat implementáló szoftvergyártók azonban eltérő konfigurációs módszereket biztosítanak a WS-* protokollok számára, és a különböző konfigurációk egymásnak való megfeleltetése nehéz feladat. További problémát jelent, hogy az XML formátumú interfészleíró nem felhasználóbarát, és nehéz biztosítani az interfészleírók megfelelő verziózását is. Az XML üzenetek ezen kívül nagy időbeli terhet rónak a kommunikációra a bináris protokollokhoz (pl. RMI, CORBA) képest. A szoftvergyártók folyamatmotorjai képesek ugyan BPEL folyamatok futtatására, azonban a hosszú ideig futó folyamatok esetén felmerülhet az igény a futó folyamatpéldányok másik folyamatmotorra való átvitelére, hogy a gyártóktól való függést csökkenteni lehessen. Ezek a legtöbbször előkerülő kihívások a webszolgáltatásokkal kapcsolatos fejlesztések során.

A disszertáció ezekre a kihívásokra nyújt megoldást. A disszertáció eredményei három fő részre oszthatók. Az első rész a webszolgáltatásokkal kapcsolatos fejlesztések termelékenységén biztosít jelentős javulást. Ez a rész egy platformfüggetlen szakterület-specifikus modellező keretrendszert mutat be, amely képes együttműködő forráskódokat és konfigurációs fájlokat előállítani a legfontosabb SOA eszközök számára, és ezáltal a webszolgáltatásokra épülő alkalmazások hordozhatósága is javul. A második rész az üzleti folyamatok példányainak folyamatmotorok közötti hordozására nyújt megoldást. A javasolt módszer minden folyamatmotorral működik, bár a hordozható folyamatoknak teljesíteniük kell bizonyos feltételeket. A harmadik rész egy teljesítménymodellt mutat be, amely lehetőséget nyújt a webszolgáltatások kommunikációs időterhének becslésére még tervezési fázisban. Ez nagyon hasznos lehet, ha a tervezett szolgáltatásoknak szigorú QoS követelményeknek kell megfelelniük.

1

A kutatás előzményei és céljai

A Szolgáltatásorientált Architektúra (SOA) elosztott komponensek összekapcsolásának alapelveit definiálja. A SOA egy architekturális paradigma, amely különböző technológiák segítségével implementálható. Ilyen korábbi technológiák például az RMI, a CORBA és a DCOM. Ma a legnépszerűbb implementációs technológiát a webszolgáltatások adják. A webszolgáltatások platformfüggetlen elosztott kommunikációt valósítanak meg azáltal, hogy minden kapcsolódó szabvány XML-re épül. A kommunikációs protokoll (SOAP), az interfészleíró (WSDL) és a köztesrétegbeli aspektusok (WS-* szabványok: WS-Addressing, WS-ReliableMessaging, WS-Security, WS-SecureConversation, stb.) is mind XML-en alapulnak. Ezeket a köztesrétegbeli aspektusokat is WSDL-ben elhelyezett platformfüggetlen XML részletek segítségével lehet konfigurálni, az úgynevezett WS-Policy megoldással. A webszolgáltatások komplexebb folyamatokká is összekombinálhatók a Business Process Execution Language (BPEL) szabvány segítségével. A BPEL folyamatok végrehajthatók, az implementációjuk ugyancsak XML-ben van leírva, és önmaguk is webszolgáltatások, vagyis tovább kombinálhatók komplexebb folyamatokká.

Az XML-nek és a szabványosításnak köszönhetően a webszolgáltatások megoldást nyújtanak a különböző programnyelvek, futtatási környezetek és hardver architektúrák közötti együttműködésre. A webszolgáltatásokhoz kapcsolódó szabványok csak az interfész megadásával és a protokoll üzenetek formátumával foglalkoznak, és nem kötik meg, hogy ezeket hogyan kell a különböző programnyelvekre leképezni. Talán ennek is köszönhető, hogy ilyen széles körben elterjedtek. A legfőbb adaptálók között található a Microsoft, az Oracle, az IBM, valamint a nyílt forráskóddal dolgozó közösségek, mint például a RedHat és az Apache. Mindegyik szereplő alkalmazás-

szervereket, keretrendszereket és fejlesztőeszközöket biztosít a webszolgáltatások fejlesztésének megkönnyítéséhez.

A csupán XML-re épülő együttműködés azért lehet sikeres, mert minden szereplő megválaszthatja, milyen módon implementálja a szabványokat. Egyedül arra kell ügyelniük, hogy a vezetéken a megfelelő üzenetek menjenek át. Az XML használatának azonban vannak árnyoldalai is. A disszertációmban ezek közül a legfontosabbakat veszem górcső alá, és nyújtok ezekre megoldást. Megmutatom, hogy az általam javasolt megoldás nagyobb produktivitásnövekedéssel jár, mint a már létező megoldások. A felsorolt kihívások az irodalomból, a fejlesztők gyakorlati tapasztalataiból, valamint a valós életbeli projektekben szerzett saját tapasztalataimból származnak.

1.1 Webszolgáltatások fejlesztésének legfontosabb kihívásai

Az XML mint interfészleíró formátum és mint üzenetformátum megoldást nyújt a különböző programnyelvek és futtatási környezetek közötti együttműködés problémájára. Azonban az XML használatának ára van. Az XML használata miatt a fejlesztőknek számos kihívással kell szembesülniük, és ezeknek mind a teljesítményre, mind pedig a fejlesztés produktivitására kihatása van.

Egy szoftverfejlesztési folyamat alapvető tevékenységei a követelményelemzés, a specifikáció, a tervezés, az implementáció, a tesztelés, valamint a karbantartás [Som11]. A különböző szoftverfejlesztési életciklusok csak abban különböznek, hogy ezek a tevékenységek mekkora hangsúlyt kapnak, és hogy hogyan követik egymást. A SOA alkalmazások és a webszolgáltatások is valamilyen szoftverfejlesztési életciklus során keletkeznek, és az XML elsősorban a specifikáció, az implementáció és a karbantartás fázisaiban okozhat produktivitási gondokat.

A WSDL interfészleíró egy komplex és redundáns XML struktúra, és ezáltal nem alkalmas emberi szerkesztésre. A szoftvergyártók ezt hamar felismerték, és több megoldást is kidolgoztak erre a problémára. A leggyakoribb megoldás az, hogy a WSDL a szolgáltatás implementációjából automatikusan generálódik. Ez az alulról felfelé típusú fejlesztés azonban csak marketing és prototipizálási célokra alkalmas. Éles környezetben ezt a fajta fejlesztést célszerű mellőzni, mert az implementáció bármilyen kis változása az interfész megváltozását is maga után vonhatja. Elképzelhető egy olyan fejlesztési módszer is, hogy az eredeti WSDL-t ezzel a módszerrel generáljuk, majd a továbbiakban ezt a WSDL-t használjuk kiindulópontként. Azonban a WSDL karbantartása hasonló nehézségű feladat, mint a WSDL teljes egészében kézzel történő megírása. Egy következő lehetséges módszer a WSDL előállítására az eszközök által

biztosított grafikus szerkesztők. A probléma ezekkel a szerkesztőkkel az, hogy nem emelik meg eléggé az absztrakciós szintet, így továbbra is meg kell küzdeni a WSDL redundanciáival. További gondot okoz az, hogy ezek a grafikus szerkesztők nem biztosítanak lehetőséget a WS-Policy beállításokra, így a WS-* protokollok konfigurálása nehéz feladat. A gyártók eszközei tehát nem biztosítanak kényelmes WSDL szerkesztési lehetőséget, és ezért a fejlesztés során nehéz követni az éles projektek során ajánlott [FM14] fentről lefelé (WSDL-ből kiinduló) történő fejlesztést.

A WS-* protokollok konfigurálására szolgáló WS-Policy beállítások még nehezebbé teszik a WSDL fájlok kezelését. A WS-Policy XML-ek hosszúak és bonyolultak is lehetnek, ezért nehéz őket kézzel előállítani. A legtöbb SOA eszköz (pl. Oracle JDeveloper, IBM WebSphere) éppen ezért biztosít egy katalógust néhány előre összeállított WS-Policy XML számára. Más eszközök grafikus szerkesztőt is biztosítanak (pl. Netbeans), de vannak olyanok is, amelyek a WS-Policy beállításokat egy kényelmesebb saját konfigurációs formátumra konvertálják (pl. Apache CXF, Microsoft WCF). Ugyan ezek a megoldások megkönnyíthetik a fejlesztők dolgát, de akár az eszközök közötti interoperabilitási problémákhoz is vezethetnek, hiszen a különböző konfigurációs reprezentációk összehangolása mindkét oldali eszköz alapos ismeretét igényli.

A biztonsági protokollok konfigurációja jelenti a legnagyobb kihívást, mivel ezek a protokollok nagyon sok beállítást igényelnek, mint például a tanúsítványok, a titkosítási és digitális aláírási algoritmusok. Az XACML (eXtensible Access Control Markup Language) szabvány az attribútum alapú hozzáférés-szabályozást (Attribute-Based Access Control, ABAC) hivatott megoldani platformfüggetlen módon. Azonban mivel az XACML is XML-en alapul, az XACML feltételek kézzel való megírása ugyancsak kényelmetlen. További problémát okoz, hogy az XACML még nem eléggé elterjedt. Például a Microsoft WCF keretrendszere nem támogatja, és a legtöbb nyílt forráskódú megoldás egy régi Sun könyvtárra épül.

További problémát okoz az, hogy a gyártók a szabványokat eltérő módon implementálják. A WS-* szabványoknak több verziója is létezik, és annyira bonyolultak és többértelműek, hogy a különböző gyártók csak bizonyos részeket implementálnak belőlük, vagy máshogy értelmezik őket. Ezek a problémák pedig interoperabilitási gondokhoz vezetnek. Ezt a gyártók hamar felismerték, és megalapították a Web Services Interoperability (WS-I) szervezetet. Ez a szervezet a követendő gyakorlatokat definiálja, és korlátozza a WS-* szabványok specifikációját azért, hogy azok egyértelművé váljanak, és hogy a WS-I konform eszközök együttműködése megvalósulhasson. Ugyan a WS-I teszteseteket is biztosít az általa hozott szabályok ellenőrzésére, a szervezet maga nem tanúsítja a különböző gyártók eszközeit. A legtöbb szabály ráadásul csak futásidőben ellenőrizhető a szolgáltatások tényleges meghívásával, ami egyáltalán nem garantálja, hogy a gyártók WS-* implementációi minden esetben WS-I

konformak. És még ha az eszközök WS-I konformak is, akkor sem biztosított a teljes interoperabilitás. Ugyanis a protokollok WS-I konform konfigurálása és ezen konfigurációk eszközök közötti megfeleltetése továbbra is kihívást jelent.

Habár a webszolgáltatás szabványok nem adják meg, hogyan kell őket a különböző programnyelvekre leképezni, a szoftvergyártók ezt megteszik a saját módjukon. Ezeket a leképzéseket webszolgáltatás API-knak hívjuk. A .NET világban jelenleg a Windows Communication Foundation (WCF) API, a Java világban pedig a Java API for XML-Based Web Services (JAX-WS) API használatos. A két API ekvivalensnek tekinthető, hiszen hasonló metaadatokat (.NET-ben attribútumok, Java-ban annotációk) definiálnak, amelyek a nyelvi interfészek és a WSDL, valamint a nyelvi értékek és a SOAP üzenetek között biztosítanak leképzést. A WCF nem támogatja közvetlenül a WS-Policy beállításokat, de képes konvertálni a WS-Policy beállítások és a saját konfigurációs megoldása között. A JAX-WS specifikáció egy közös API-t biztosít a JavaEE alkalmazásszerverek számára, azonban a JAX-WS csak az egyszerű WSDL-lel és SOAP-pal foglalkozik, és nem tér ki a WS-Policy beállításokra. Ez azt jelenti, hogy a különböző szoftvergyártóknak saját megoldást kell adniuk a WS-Policy beállítások kezelésére. Ez pedig a WS-* protokollok használata esetén a különböző Java alkalmazásszerverek közötti hordozhatósági problémákhoz vezet.

A JavaEE specifikációk célja vállalati Java alkalmazások hordozhatóságának biztosítása különböző JavaEE szerverimplementációk között. Habár megengedettek a gyártóspecifikus beállítások, a forráskódok és konfigurációk nagy részének hordozhatónak kéne lennie. A WS-* protokollok beállításai éppen ezt nehezítik meg. Egy másik szerverre való áttérés során meg kell tanulni az új szerver speciális beállításait, hogy az alkalmazást portolni lehessen. Ha egy alkalmazást más szerverekre kell átültetni, az alkalmazás adatbázisa általában továbbra is használható, de akár az is átmigrálható egy másik gyártó eszközére. Azonban vannak olyan gyártóspecifikus adatbázisok, amelyek nem hordozhatók, és ezeknek a migrálása ugyanazon eszköz különböző verziói között is problémás lehet. Ilyen például a BPEL folyamatok példányainak állapotát tároló adatbázis. A BPEL folyamatpéldányok állapotának migrációja tehát egy újabb kihívást jelent.

A karbantartási fázisban a szolgáltatások interfészeinek verziózása [EKW+09] jelent nagyobb feladatot. A fentről lefelé történő fejlesztés a WSDL és XSD fájlok verziózását igényli, azonban ezeket nehéz karbantartani és frissíteni. Az ilyen XML formátumú fájlok követése és összefésülése még egy verziókövető rendszerrel sem egyszerű. Ugyan a grafikus WSDL szerkesztők elrejtik az XML kényelmetlenségeit, azonban ezek sem biztosítanak verziózási és összefésülési lehetőségeket.

A webszolgáltatások további problémája, hogy az XML formátumú üzenetek so-

rosítása megterheli a kommunikációt, különösen a bináris protokollokhoz képest. A sorosításból származó időveszteség akár az alkalmazás logikájának futásidejével is összemérhető lehet. Ez a teljesítményveszteség a webszolgáltatások ellen irányuló legfőbb kritikák egyike. Ha az alkalmazás szolgáltatási minőségének (Quality of Service, QoS) szerződésben megfogalmazott elvárásokat (Service Level Agreement, SLA) kell teljesítenie, akkor fontos, hogy az XML sorosításból származó teljesítményvesztést még tervezési időben meg tudjuk becsülni, még mielőtt a szolgáltatást implementálnánk.

A fenti kihívások hátráltatják leginkább a webszolgáltatásokkal kapcsolatos fejlesztéseket. Ezeket az alábbi lista foglalja össze:

- nincs felhasználóbarát módja a WSDL fájlok létrehozásának, karbantartásának és verziózásának
- a különböző eszközök közötti interoperabilitás megvalósítása nehéz feladat, mert a különböző eszközök különböző módszereket használnak a WS-* protokollok konfigurálására, és a szabványoknak is különböző részeit implementálják
- a JAX-WS API nem foglalkozik a WS-* protokollokkal, ezért a WS-* protokollokat használó szolgáltatások különböző JavaEE szerverek közötti hordozhatósága megoldatlan
- az XACML kényelmetlen API a hozzáférés-kezelés leírására
- a BPEL folyamatok példányainak állapotát nehéz migrálni különböző folyamatmotorok között
- az XML sorosítás jelentős terhet ró a SOAP kommunikációra, azonban ez az overhead tervezési időben nem becsülhető előre

Amennyiben ezeken a kihívásokon sikerülne túllendülni, a webszolgáltatások fejlesztési folyamatának produktivitása jelentősen javítható lenne, különösen a tervezési, implementációs és karbantartási fázisban. A disszertációban megoldást nyújtok a fent felsorolt kihívások mindegyikére, és megmutatom, hogy a megoldásom nagyobb produktivitás növekedést eredményez, mint a korábban elérhető megoldások.

1.2 Korábbi megoldások

A korábbi kutatások próbáltak megoldást adni az előző szakaszban felsorolt kihívásokra, azonban egyetlen korábbi megoldás sem teljesíti egyszerre a felhasználóbarátság, a platformfüggetlenség és a magas fokú produktivitás követelményeit.

A webszolgáltatások modellezésének egyik legegyszerűbb módja az UML használata. Vannak sztereotípiák nélküli [GSSO04] és sztereotípiákkal dolgozó megoldások [CMV04, DNsmGW08, EES10] is, azonban ezek nem alkalmasak a WS-* protokollok kényelmes konfigurálására.

H. Wada et. al. [WSO08, WSO06] egy nagyon részletes UML profilt adott, amely sok köztesrétegbeli aspektust képes leírni, azonban a bonyolultabb protokollok (pl. WS-SecureConversation, WS-Trust és WS-Federation) nem támogatottak, és hiányzik a modell leképzése a fontosabb szoftvergyártók SOA eszközeire.

A legtöbb megoldásból [OMG09, JB08, PWH08, CBZ+04, QN10] hiányzik a köztesrétegbeli aspektusok modellezése, így a WS-* protokollok konfigurálására alkalmatlanok.

A webszolgáltatások egy másik lehetséges leírási módja a szemantikus webes technológiák [W3C12b, W3C12a, ESS12, KPKH05, VAG05, SSVS04] használata. A szemantikus webszolgáltatások (Semantic Web Services, SWS) célja, hogy intelligens szoftver ügynököket biztosítsanak az automatizált együttműködés megvalósításához [Klu08]. A szemantikus webszolgáltatásokat nem modellezésre tervezték, és mivel saját futási környezetet igényelnek, így nem használhatók az ipari SOA eszközökkel.

A korábbi megoldások problémája, hogy nem fedik le a legfontosabb WS-* szabványokat, nem támogatják a legfontosabb ipari SOA eszközöket és nem a megfelelő absztrakciós szinten írják le a webszolgáltatásokat.

Az üzleti folyamatpéldányok migrálására is vannak már létező megoldások. Zaplata et. al. [ZHKL10] egy általános megoldást adott, ami nem igényli a folyamat módosítását és nem támaszt vele szemben plusz követelményeket, hanem a migrációs adatok automatikusan számítódnak a folyamat leírása és a folyamatpéldány futása alapján. A megközelítés hátránya azonban az, hogy a folyamatmotor módosítását igényli, ami zárt forráskódú motorok esetén nem mindig járható út.

Más megközelítések [CAB08, KG99, DVRS03, PSSvdA07, RRKD05, RMR10] a folyamatleírás evolúcióját követik. Ezek a megoldások azonban vagy a folyamatmotor

módosítását igénylik, vagy pedig egy saját folyamatmotort használnak, így nem használhatók az ipari SOA termékkel. Néhány megoldás nem teljes, hiányzik az időzített feladatok és a párhuzamos ágak kezelése, valamint a nem migrálható folyamatokkal egyáltalán nem foglalkoznak.

A webszolgáltatások kommunikációjának teljesítményét is többen vizsgálták. A legtöbb ilyen vizsgálat [NSP+10, REB11, JRB+06, JRH00, SSFG04, vEZ08] csak a korábbi (RMI, DCOM, CORBA) elosztott technológiákkal hasonlítja össze a SOAP alapú kommunikációt. A vizsgálatok eredménye az, amit várnánk: a webszolgáltatások nagy kommunikációs teherrel rendelkeznek a bináris protokollokhoz képest, különösen ha a titkosítás is engedélyezett. Azonban egyik vizsgálat sem megy ennél tovább, egyik sem biztosít megoldást a teljesítmény megbecslésére.

Az összetett szolgáltatásoknak is gazdag irodalma van [AdL08, ESBS07, AGM08, KKK08, MM07]. Ezek a cikkek azonban a szolgáltatások összekombinálására fókuszálnak, azonban a legalsó szinten az egyes elemi szolgáltatások válaszidejére is szükség lenne, hogy ezek a módszerek használhatók legyenek. Erre azonban már nem térnek ki ezek a cikkek.

G. Imre et. al. [IKLC10] egy XML sorosításon alapuló költségmodellt dolgozott ki Java és .NET környezetekhez. Az XML sorosítás a SOAP üzenetküldés alapja, ezért ez egy fontos eredmény, azonban a cikk csak három primitív típussal (string, int és double) foglalkozik. A többi primitív típus vizsgálata, valamint a WS-* protokollok elemzése is fontos lenne az előrelépéshez.

Jelenleg nem létezik olyan korábbi munka, amely a webszolgáltatások válaszidejének alapos vizsgálatával foglalkozna, és egyetlen korábbi munka sem ad olyan teljesítménymodellt, amellyel ez a válaszidő a szolgáltatás interfésze alapján becsülhető lenne.

1.3 Módszertani összefoglaló

A legtöbb felsorolt fejlesztési kihívás éles projektekből származik, vagyis a legtöbbjük a gyakorlatban fellépő igények alapján keletkezett. A fejlesztéssel kapcsolatos produktivitási feladatokat a magyar E-Közigazgatási Keretrendszer kialakítása című projektben való részvétel során hamar felismertem, ahol a különböző SOA eszközök közötti interoperabilitás megteremtése volt az egyik legfontosabb feladat. Ezeknek a feladatoknak a megoldása határozta meg a kutatásaim fő irányát.

Először a létező megoldásokat vizsgáltam meg, azonban egyetlen korábbi megoldás sem fedte le a szükséges WS-* protokollokat, egyik sem a megfelelő absztrakciós szinten dolgozott, egyik sem adott konfigurációs leképzést a legfontosabb ipari SOA termékekre. Ezért kezdtem el foglalkozni egy kifejezetten webszolgáltatások leírására szolgáló szakterület-specifikus nyelv kialakításával, amely a megfelelő absztrakciós szinttel rendelkezik, és amely biztosítani tudja a valós életben előforduló projektek számára a megfelelő produktivitást.

Megvizsgáltam az üzleti folyamatok példányainak migrálásával foglalkozó megközelítéseket is, azonban egyik sem volt egyszerre teljes, platformfüggetlen és a folyamatmotorokba való beavatkozást nem igénylő megoldás. Az üzleti folyamatok példányainak hordozhatósága egy e-közigazgatási környezetben alapvető fontosságú, és csak egy folyamatmotorokba való beavatkozást nem igénylő megoldás elfogadható, mert a zárt kódú motorok nem minden esetben testreszabhatók. Ezért kezdtem el egy új, korábban még nem vizsgált irányt feltérképezni, amely a folyamatpéldányok által küldött és kapott üzenetek naplózásán és visszajátszásán alapult. Ez a megközelítés ígéretesnek bizonyult, azonban a migrálható folyamatokkal szembeni pontos követelményeket még ki kellett dolgozni, és bizonyítani kellett a megoldás teljességét.

A webszolgáltatásokkal szembeni kritikák egyik legfontosabb eleme, hogy a kommunikációs overhead túlságosan nagy, és egy üzleti környezetben szigorú QoS előírások is lehetnek a válaszidőre. Azért, hogy ezeket a követelményeket teljesíteni lehessen, szükség van a szolgáltatás interfésze alapján a válaszidő előzetes becslésére még azelőtt, hogy a szolgáltatás implementációjára sor kerülne. Megvizsgáltam a korábbi megoldásokat ezen a területen is, azonban egyetlen korábbi megoldás sem fedte le az összes primitív típust, valamint a WS-* protokollok hatásának becslését még senki nem vizsgálta korábban.

Mivel a felsorolt problémák a gyakorlati életből származnak, a kutatásaimat alkalmazott kutatásként lehet besorolni. Az elméleti eredményeket a gyakorlatba is visszaültettem. A webszolgáltatások leírására általam javasolt szakterület-specifikus modellező keretrendszer a produktivitását több valós projektben is bizonyította, többek között a magyar E-Közigazgatási Keretrendszer kialakítása második fázisában és a MÁV-val egy közös projekt keretében is rendkívül hasznosnak bizonyult a keretrendszer. Ezen túlmenően az elért tudományos eredmények további kutatási irányokat is megnyitottak.

1.4 A disszertáció tudományos eredményei

A disszertáció a webszolgáltatások fejlesztésével kapcsolatos következő tudományos eredmények részletes leírását tartalmazza:

- megalkottam egy szakterület-specifikus nyelvet, amely egy felhasználóbarát platformfüggetlen leírást biztosít WS-* protokollokat is használó webszolgáltatásokból álló elosztott rendszerek leírására
- elkészítettem egy generátort, amely a legfontosabb ipari SOA eszközök számára interoperábilis programkódokat és konfigurációs fájlokat állít elő, és a korábbi megoldásokhoz képest nagyobb produktivitással rendelkezik
- kidolgoztam egy módszert, amely az üzenetek naplózásán és visszajátszásán alapulva alkalmas BPEL folyamatok példányainak különböző folyamatmotorok közötti migrálására
- kifejlesztettem egy teljesítménymodellt, amely már tervezési időben is alkalmas a webszolgáltatások válaszidejének becslésére a szolgáltatások interfészleírója alapján

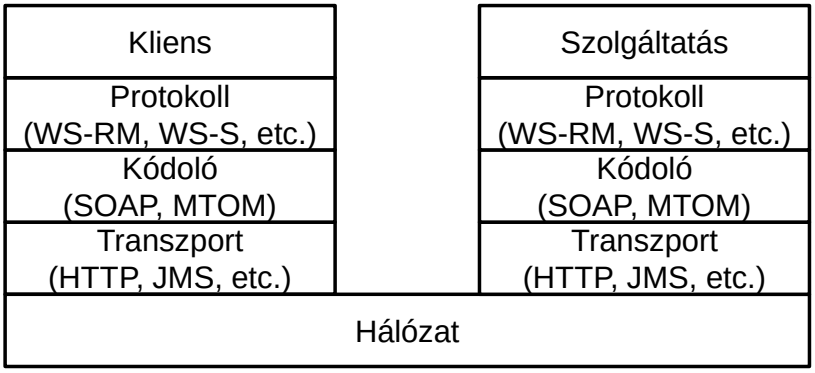
2

Új tudományos eredmények

Ez a fejezet a webszolgáltatás-keretrendszerek általános architektúrájának bemutatása után ismerteti a disszertáció új tudományos eredményeit. Először az elosztott webszolgáltatásokból álló rendszerek platformfüggetlen leírására szolgáló szakterület-specifikus modellező nyelv kerül bemutatásra. A modellező nyelv alkalmas az egyszerű és komplex típusok, a szolgáltatások interfészeinek és köztesrétegbeli aspektusainak megadására, beleértve a címzést, a megbízható üzenetküldést, a tranzakciókat és a titkosítást, valamint leírhatók az elő- és utófeltételeket megfogalmazó szerződések és az attribútum alapú jogosultságkezelés is. A következő szakasz az üzleti folyamatok példányai állapotának migrációjára szolgáló algoritmus alapötletét vázolja fel. Az utolsó szakasz a webszolgáltatások teljesítménybecslésével kapcsolatos módszert ismerteti.

2.1 Webszolgáltatás implementációk architektúrája

A legelterjedtebb WS-* protokollok a WS-Addressing, WS-ReliableMessaging, WS-Security, WS-SecureConversation és WS-AtomicTransaction. A Microsoft .NET keretrendszer, a GlassFish szerver, az Oracle WebLogic szerver, az IBM WebSphere szerver és az Apache CXF keretrendszer is támogatja legtöbbjüket. A keretrendszerek közötti fő különbség abban rejlik, hogy miként lehet ezeket a protokollokat konfigurálni. Sok egyéb WS-* protokoll is létezik még, azonban ezeket az elterjedt SOA eszközök nem támogatják ilyen széles körben, így ezekre a protokollokra a disszertáció nem tér ki.



2.1. ábra. A webszolgáltatás-keretrendszerek implementációjának tipikus architektúrája

A 2.1-es ábra a webszolgáltatás-keretrendszerek implementációjának tipikus architektúráját mutatja. A Windows Communication Foundation (WCF) [Mic14] (a Microsoft .NET keretrendszerből) és a Metro [Ora14] (a JAX-WS referencia implementációja) keretrendszerek ezt az architektúrát követik, és a többi implementáció is hasonlóan modellezhető.

A hálózat található a legalsó szinten, itt kerülnek átküldésre a bájtok. Webszolgáltatásoknál tipikusan a HTTP protokoll használatos, azonban ez más protokollra is lecserélhető.

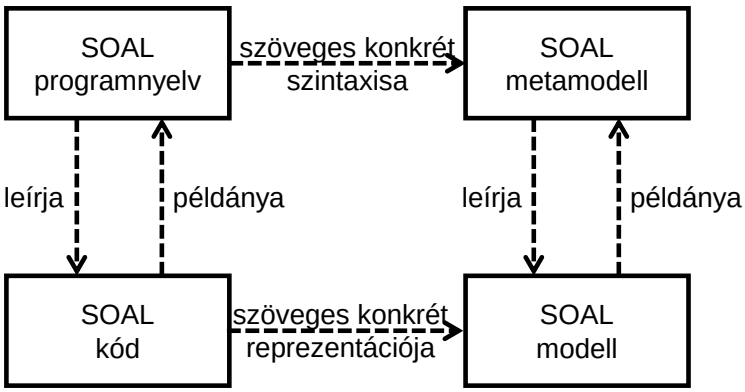
A szállító réteg felelős a hálózati protokoll kezeléséért. A szerver oldalon várja a kliensek csatlakozását, kliens oldalon pedig csatlakozik a szolgáltatásokhoz. Ez a réteg felelős a bájtok átküldéséért a szereplők között.

A kódoló réteg biztosítja a transzformációt a bájtsorozatként vándorló üzenet és annak keretrendszerbeli objektumreprezentációja között, vagyis ez a réteg felelős a sorosításért és visszasorosításért (pl. SOAP-ra MTOM-mal vagy anélkül). A szállító és kódoló rétegek mindig kötelezőek.

A protokoll rétegek opcionálisak. Ezek implementálják a különböző WS-* szabványokat (pl. WS-ReliableMessaging, WS-Security, stb.). A protokoll rétegek általában plusz inicializáló és lezáró üzeneteket iktatnak a kommunikációba vagy plusz fejléceket adnak az üzenetekhez. Például a WS-ReliableMessaging protokoll a megbízható kapcsolat felépítésénél és lezárásánál alkalmaz ilyen plusz üzeneteket, valamint az üzenetekhez plusz fejléceket is ad, amelyek a már megérkezett üzenetek visszaigazolását végzik. A WS-SecureConversation is plusz üzeneteket használ a biztonsági kulcs megbeszéléséhez. A WS-Security és WS-SecureConversation protokollok is további fejléceket adnak az üzenetekhez, többek között időbélyeget, digitális aláírásokat és titkosított kulcsokat.

2.2 Metamodell webszolgáltatások leírására

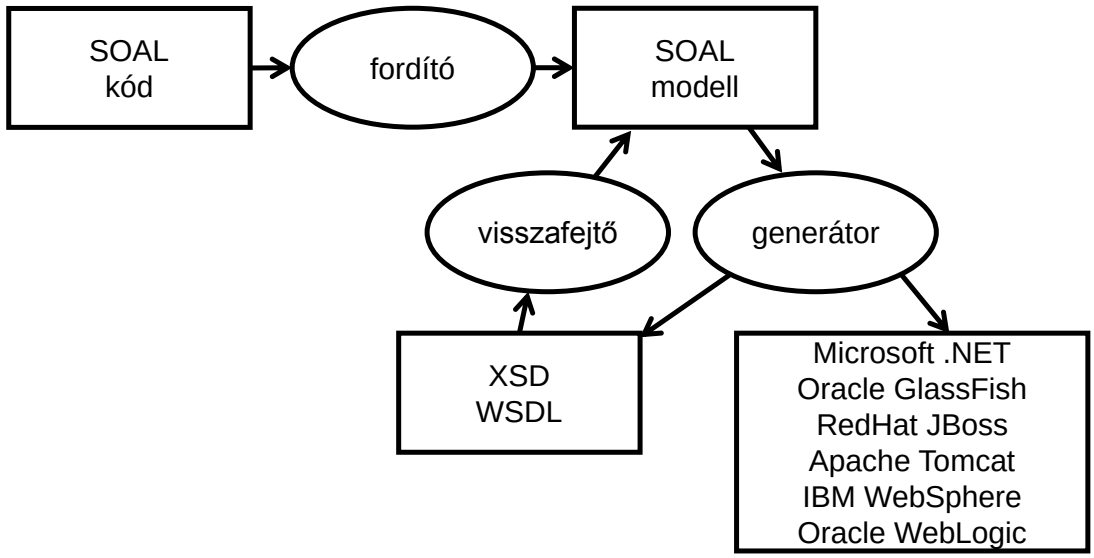
Mivel a webszolgáltatás-keretrendszerek eltérő konfigurációs módokat biztosítanak, ezeket a konfigurációkat nehéz összeegyeztetni egymással. A probléma megoldására megfogalmazható kutatási célként egy felhasználóbarát, platformfüggetlen szakterület-specifikus nyelv (Domain Specific Language, DSL) kialakítása a WS-* szabványok számára, hogy az elosztott webszolgáltatásokból álló rendszereket modellezni lehessen, a modellből pedig előállíthatóak legyenek a platformfüggő konfigurációk a különböző keretrendszerek számára. A disszertáció keretein belül platform alatt a webszolgáltatás-keretrendszerek értendők. A disszertációban bevezetett szakterület-specifikus nyelv a SOAL (Service-Oriented Architecture Language).



2.2. ábra. SOAL metamodell és programnyelv

A SOAL egy technikai DSL, mivel a célközönség a webszolgáltatás-keretrendszerekkel dolgozó fejlesztők. A fejlesztők jobban kedvelik a szöveges szintaxist, mint a grafikus szintaxist [ZS09] [Voe08], éppen ezért a SOAL egy szöveges konkrét szintaxissal rendelkezik, amelyet a SOAL programnyelv ír le. Az absztrakt szintaxist a SOAL metamodell definiálja. Egy SOAL programkód a SOAL programnyelv egy példánya, és SOAL modellé fordul, amelyet a SOAL metamodell ír le (lásd a 2.2-es ábrát).

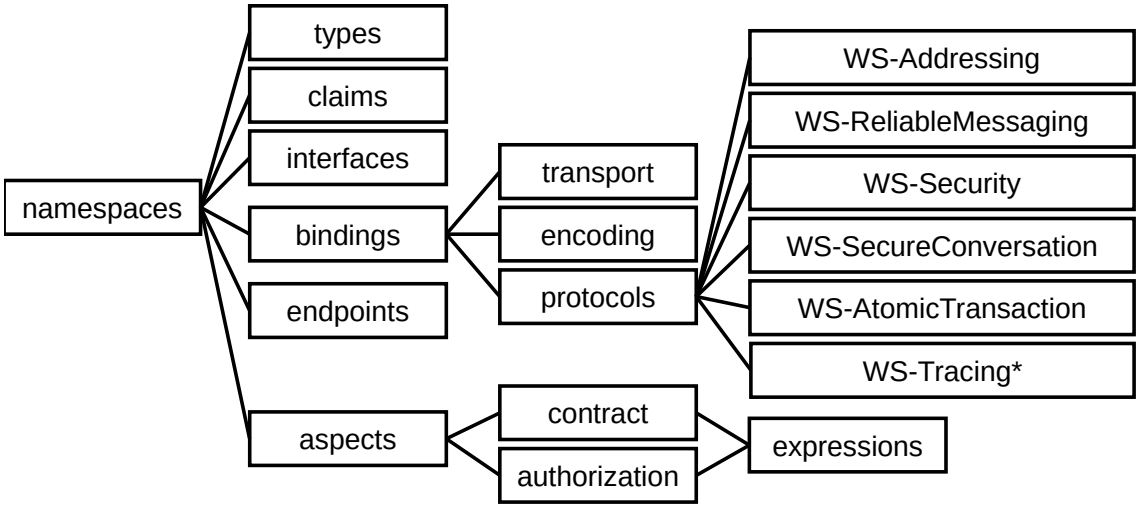
A SOAL keretrendszer a SOAL metamodell és programnyelv köré épül. A keretrendszer része egy fordító és egy kódgenerátor (lásd 2.3-as ábra). A fordító egy SOAL programkódot SOAL modellé fordít. A kódgenerátor szabványos XSD és WSDL fájlokat, valamint forráskódokat, konfigurációs fájlokat és projekteket készít a különböző webszolgáltatás-keretrendszerek számára. Jelenleg a Windows Communication Foundation a Microsoft .NET-ben Visual Studio-val és IIS-sel, a Metro keretrendszer GlassFish szerverrel és Netbeans-zel, az Apache CXF keretrendszer JBoss Application Server-rel és Eclipse-szel, az Apache CXF keretrendszer Tomcat Server-rel és Eclipse-szel, az IBM WebSphere szerver a Rational Application Developer-rel, és az Oracle WebLogic szerver JDeveloper-rel támogatottak. A SOAL keretrendszer egyik fő célja a



2.3. ábra. A SOAL keretrendszer architektúrája

fentről lefelé (WSDL-ből kiinduló) fejlesztés produktivitásának növelése, ezért a SOAL modellből az XSD és WSDL fájlok is előállnak.

A keretrendszer támogatja XSD és WSDL fájlok visszafejtését SOAL modellé. Ennek köszönhetően a már létező szolgáltatásokhoz kapcsolódó fejlesztések termelékenysége is javítható, mert ezek a szolgáltatások a különböző JAX-WS implementációk között könnyen hordozhatóvá válnak.



2.4. ábra. A SOAL metamodell architektúrája

A 2.4-es ábra a SOAL metamodell architektúráját mutatja. A WSDL legfontosabb részei megjelennek a SOAL metamodellben is, azonban a redundáns message rész nincs modellezve. Minden más elem (types, portType, binding és endpoint) szerepel a SOAL metamodellben, és további deklarációk is elérhetők például az attribútum

alapú jogosultságkezelés támogatására. A binding-ok definiálják a transzport (transport), kódoló (encoding) és protokoll (protocol) rétegeket is. A támogatott protokollok a WS-Addressing, WS-ReliableMessaging, WS-Security, WS-SecureConversation, WS-AtomicTransaction. A WS-Tracing nem egy szabványos protokoll. Ezt a protokollt az üzleti folyamatok példányai migrálásának támogatásához hoztam létre. A SOAL ezek mellett kétfajta aspektus leírására is alkalmas: az egyik az elő- és utófeltételeket megfogalmazó szerződések (contracts), a másik pedig az attribútum alapú hozzáférést (authorization) írja le. Mindkét aspektusban kifejezések (expressions) segítségével lehet megadni a kívánt feltételeket.

Az új tudományos eredményeket a következő tézis foglalja össze:

1. Tézis *Definiáltam egy új szakterület-specifikus metamodellt SOAL néven, amely elosztott, WS-* protokollokat is használó webszolgáltatásokból álló rendszerek platformfüggetlen leírására szolgál. Definiáltam egy új programnyelvet, amely szöveges konkrét szintaxisként szolgál a metamodell számára. A SOAL metamodell és SOAL programnyelv jobb absztrakciós szintet adnak, és ezáltal egyszerűbb, felhasználóbarátabb és produktívabb leírást biztosítanak a webszolgáltatásokra, mint a korábbi megoldások. Specifikáltam egy transzformációt, amely SOAL modelleket interoperábilis programkódokra és konfigurációs fájlokra képez le a legfontosabb ipari webszolgáltatás-keretrendszerek számára. A transzformáció helyességét interoperabilitási tesztekkel verifikáltam. A metamodellt, a programnyelv fordítóját és a transzformációt .NET környezetben implementáltam.*

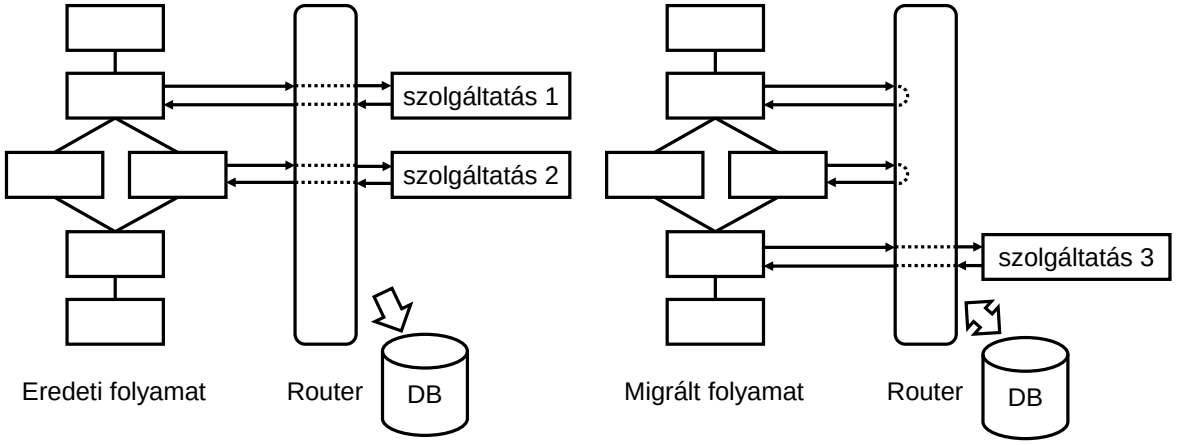
A tézishez kapcsolódó publikációk: [1] [2] [4] [5] [6] [7] [8] [9] [10] [11] [15]

2.3 Üzleti folyamatok példányainak migrálása

Sok üzleti folyamat hosszú lefutású, mert általában emberi beavatkozást is igényelnek. Ez egy e-közigazgatási környezetben különösen igaz, ahol az egyes folyamatok akár napokig vagy hetekig is futhatnak. Amikor a folyamatmotornak egy új verziójára kéne áttérni, akkor nem biztos, hogy a már futó folyamatpéldányok állapotának migrálását támogatja a motor gyártója. Még nagyobb kockázatot jelent a szállítóhoz való kötöttség (vendor lock-in). Jelenleg nem léteznek olyan konverterek, amelyek különböző gyártók folyamatmotorjai között biztosítanák az átjárást.

A BPEL folyamatok gyártóspecifikus belső állapotreprezentációinak migrálása bonyolult és nehezen fenntartható feladat, mivel minden egyes folyamatmotornak az összes szükséges verzióját figyelembe kéne venni.

A fentiek miatt egy olyan megoldást javaslok a folyamatpéldányok belső állapotának migrálására, amely kívülről, a folyamatmotorokba való beavatkozás nélkül is működik. A folyamatmotort fekete dobozként kezeli a módszer, azonban a helyes működéshez a folyamatokat speciális szabályok betartásával kell elkészíteni. Ez abból áll, hogy a folyamatpéldányokat egyedi azonosítókkal kell ellátni, amelyeknek szerepelniük kell az általuk küldött illetve kapott üzetekben is, és a nemdeterminisztikus viselkedést is kezelni kell. Ha a javasolt mintákat követjük, akkor a folyamatpéldányok a következőképpen migrálhatók. Először az eredeti példány által küldött és kapott üzeneteket egy router naplózza egy adatbázisban (lásd 2.5a ábra). A router normál esetben nem avatkozik be a kommunikációba.



(a) Eredeti folyamatpéldány (b) Migrált folyamatpéldány

2.5. ábra. Az eredeti és a migrált folyamatpéldány futása

Amikor a futó folyamatpéldányok migrálása szükségessé válik, a folyamatok leírását az új motorra kell telepíteni, és minden egyes eredeti folyamatpéldánynak egy új folyamatpéldányt kell indítani az új motoron. Az új folyamatpéldányok addig nem kommunikálnak a külvilággal, amíg el nem érték ugyanazt a belső állapotot, amit az eredeti változatuk. Egészen eddig a router elkapja az üzeneteiket, és az eredeti válaszokat visszajátssza nekik anélkül, hogy erről a külvilág értesülne (lásd 2.5b ábra). Ezáltal az új folyamatpéldányok pontosan úgy viselkednek, mint az eredeti példányok és képesek felvenni ugyanazt az állapotot. Ez a módszer ugyanazon folyamatmotor különböző verziói és akár különböző gyártók folyamatmotorjai között is működik. A megoldás egy beavatkozás nélküli megoldás, hiszen a folyamatmotorokon nem kell módosítani.

A módszer egyedül a folyamatok leírásával szemben támaszt néhány nem túl szigorú követelményt. A disszertációban megadom ezeket a követelményeket és javaslok egy keretrendszert, amely a SOAP üzenetek naplózásán és visszajátszásán alapuló migrálást tesz lehetővé BPEL folyamatok számára.

Az új tudományos eredményeket a következő tézis foglalja össze:

2. Tézis *Definiáltam egy új BPEL folyamatpéldányok migrációjára alkalmas keretrendszer architektúráját, amely a SOAP üzenetek naplózásán és visszajátszásán alapul. A korábbi megoldásokhoz képest a keretrendszer nem igényli a folyamatmotorok módosítását és minden folyamatmotorral használható. Definiáltam a migráció-kompatibilitás fogalmát a folyamatpéldányok helyes migrálásához. Definiáltam egy új protokollt, a WS-Tracing protokollt, amely a SOAP üzenetek és a BPEL folyamatpéldányok korrelációjához szükséges. Definiáltam egy új útvonalválasztási algoritmust, amely a keretrendszeren belül a folyamatpéldányok migrálását elvégzi. Megmutattam, hogy ha a folyamatok leírásai betartják a migráció kompatibilitási előírásokat, akkor az útvonalválasztási algoritmus helyes migrációt eredményez. A migrációs keretrendszert .NET környezetben implementáltam, az implementáció helyességét valós életbeli példákon keresztül verifikáltam.*

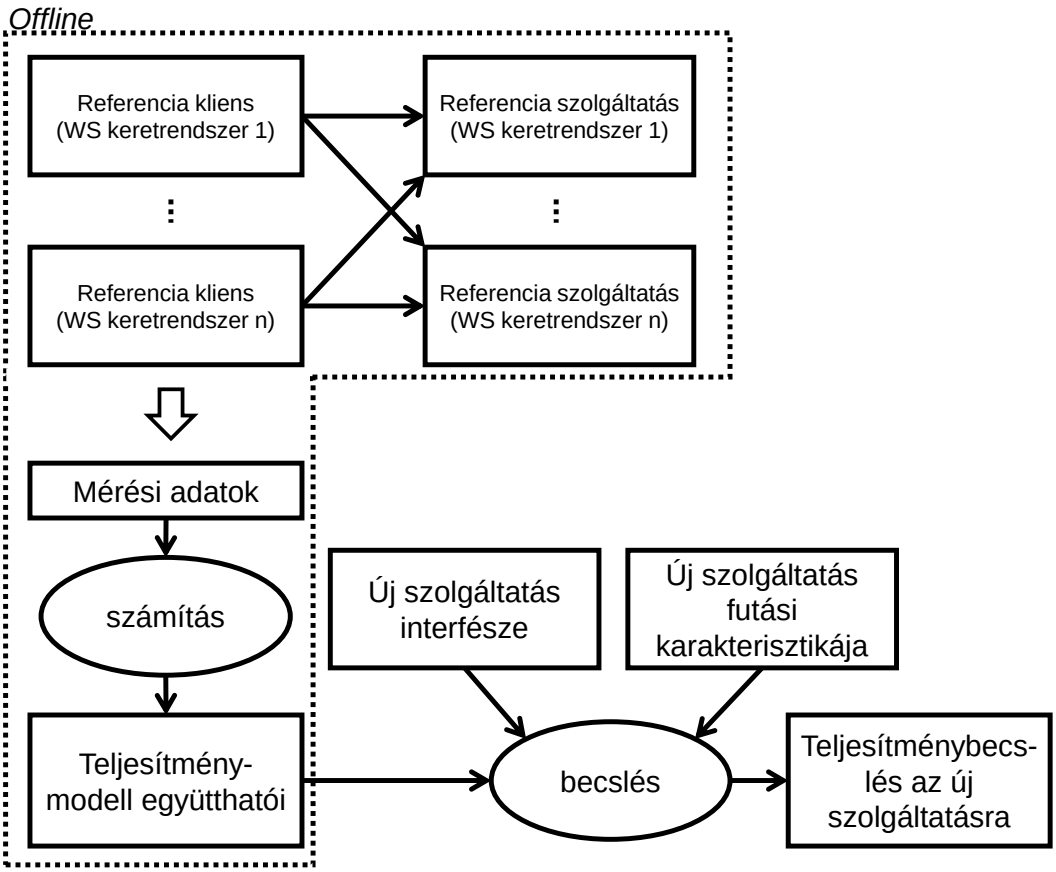
A tézishez kapcsolódó publikációk: [12] [13] [14]

2.4 Webszolgáltatások válaszidejének becslése

A webszolgáltatások kommunikációjának overhead-je elsősorban az XML sorosításból következik. A WS-* protokollok plusz fejlécekkel és üzenetekkel bővítik ki az egyszerű SOAP üzeneteket, és az XML titkosítás és digitális aláírás is csak lassítja a kommunikációt. A disszertációban egy olyan teljesítménymodellt javaslok, amely segítségével a webszolgáltatások kommunikációs overhead-je megbecsülhetővé válik a szolgáltatás interfésze alapján. A becslés működéséhez azonban a teljesítménymodell együtthatóit előre ki kell számolni.

Az együtthatók kiszámítására is adok egy módszert a disszertációban. A módszer lényege, hogy egy előre definiált ún. referencia kliens és szolgáltatás halmazzal méréseket kell végezni a különböző keretrendszerek között. A referencia szolgáltatások interfésze olyan operációkat definiál, amelyek be- és kimenő paraméterei többféle primitív típusú mezőkből álló struktúrák tömbjei. A primitív típusok a programnyelvekben használt leggyakoribb primitív típusok. A referencia szolgáltatások csak az engedélyezett WS-* protokollokban különböznek. Minden referencia klienst és szolgáltatást minden webszolgáltatás-keretrendszerben implementálni kell. Ezután minden kliensnek meg kell hívnia az összes nekik megfelelő szolgáltatást még webszolgáltatás-keretrendszerek között is. A méréseket különböző tömbhosszakkal és különböző darabszámú hívásokkal is meg kell tenni.

Mérések alapján megállapítható, hogy a különböző webszolgáltatás-keretrendsze-

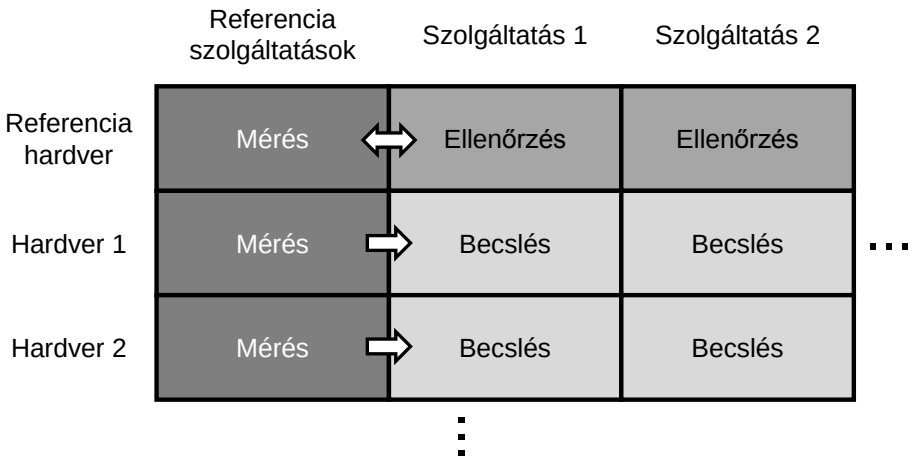


2.6. ábra. A teljesítménybecslési folyamat áttekintése

rek ugyanolyan karakterisztikával rendelkeznek: a válaszidő lineáris korrelációban áll a tömbhosszakkal és a hívások darabszámával is. Ennek köszönhetően egy közös teljesítménymodell készíthető a különböző környezetekre. Ha ezt a teljesítménymodellt használjuk, akkor a környezetek csak a modell együtthatóinak értékeiben különböznek. Ezen együtthatók értékeit a referencia szolgáltatások és kliensek közötti mérésekből lehet kiszámolni.

Egy tetszőleges interfészű szolgáltatás válaszideje megbecsülhető, ha a teljesítménymodell együtthatói valamint a szolgáltatás futásidejű tulajdonságai (pl. hívások száma, tömbök hossza) ismertek.

A 2.7-es ábra a teljesítménymodell kifejlesztésének, ellenőrzésének és felhasználásának módját mutatja. Először a referencia hardveren végeztem méréseket a referencia szolgáltatásokkal és kliensekkel. Ezeknek a referenciaméréseknek az eredményeiből alkottam meg a teljesítménymodellt. Ezt a modellt a referencia hardveren ellenőriztem korábban még nem mért szolgáltatásokra tett becslések és a valódi válaszidők összehasonlításával. Más hardvereken és más szolgáltatásokra a következőképpen lehet becsléseket tenni. Először a rederenciaméréseket kell elvégezni az új hardveren a



2.7. ábra. Teljesítménybecslés különböző fajta hardverekre és különböző szolgáltatásokra

referencia szolgáltatások és kliensek segítségével. Ezután a teljesítménymodell új hardverre specifikus együtthatóit kell kiszámolni a mérések eredményei alapján. Végül tetszőleges szolgáltatás válaszideje becsülhető az új hardveren, ha már az együtthatók rendelkezésre állnak. Ezt a javasolt módszert is ellenőriztem egy referencia hardvertől különböző hardveren.

Az új tudományos eredményeket a következő tézis foglalja össze:

3. Tézis *Kidolgoztam egy új teljesítménymodellt a webszolgáltatások válaszidő overheadjének a szolgáltatások interfésze alapján történő becslésére. A becsléshez ismerni kell a teljesítménymodell futtatási környezetre specifikus együtthatóit. Definiáltam egy referencia méréshalmazt ezen együtthatók kiszámítására. Kiszámoltam egy hardver környezetre az együtthatókat két különböző webszolgáltatás-keretrendszerre végrehajtott referencia mérésekkel. A teljesítménymodell helyességét független mérésekkel verifikáltam egy másik hardverkörnyezetben és korábban még nem mért webszolgáltatásokkal is. A teljesítménymodell a WS-* protokollok elméleti megfontolásaival is korrelál. A korábbi megoldásokhoz képest az általam javasolt teljesítménymodell lefedi a leggyakoribb primitív és komplex típusokat és a legelterjedtebb WS-* protokollokat, továbbá a tervezési fázisban is használható, amikor a szolgáltatások implementációja még nem áll rendelkezésre.*

A tézishez kapcsolódó publikációk: [3] [16]

3

Az új tudományos eredmények alkalmazása

A disszertáció tudományos eredményei valós projektekben is hasznosultak. A SOAL keretrendszer bizonyult a leghasznosabbnak, ezt három különböző projektben is sikerült alkalmazni.

A magyar kormányok egyik fő célja, hogy a közigazgatási folyamatokat elektronizálják. Ezért született meg a magyar E-Közigazgatási Keretrendszer kialakítására szolgáló projekt. A magyar közigazgatásban a különböző szakrendszerek különböző operációs rendszereket és különböző szoftvergyártóktól származó fejlesztőkörnyezeteket használnak, és igyekeznek megőrizni autonómiájukat. Ezeknek a szakrendszereknek az integrációja egyszerűbbé válik, ha rendelkezésre áll egy olyan keretrendszer, amely a különböző SOA eszközök számára interoperábilis komponenseket tud készíteni. A SOAL keretrendszert először az E-Közigazgatási Keretrendszer pilot projektjében alkalmaztuk, ahol egy valós életbeli közigazgatási folyamatot (egyéni vállalkozás alapítása) kellett webszolgáltatások és BPEL segítségével megvalósítani. A pilot rendszer négy szakrendszer szimulációját is tartalmazta. Összesen hét webszolgáltatásból, egy BPEL folyamatból és egy weboldalból állt a pilot rendszer. A szolgáltatások és a folyamat implementációját először kézzel készítettük el, és ez körül-belül egy hónapnyi időt vett igénybe. A pilot rendszer részleteit és a tapasztalatokat publikáltuk is [9]. A SOAL keretrendszer elkészülte után újra implementáltuk a szolgáltatásokat úgy, hogy először SOAL nyelven specifikáltuk őket, majd előállítottuk a WSDL fájlokat, programkódokat és konfigurációs fájlokat a különböző SOA eszközök számára. Ebben a második fázisban a valódi Ügyfélkapu rendszer tesztrendszerét is beépítettük a pilot rendszerbe. Ez a fázis csak három napot vett igénybe. Ez is mutatja, hogy a SOAL keretrendszer nagyban meg tudja növelni az elosztott SOA rendszerek fejlesztésének

produktivitását.

Ebben a második fázisban a üzleti folyamatok példányainak migrálására javasolt megoldás prototípusát is teszteltük két valódi közigazgatási folyamat segítségével: egyéni vállalkozás alapítása és anyasági támogatás igénylése. A folyamatpéldányok migrációja sikeres volt az ActiveVOS, az Oracle BPEL Process Manager és az IBM WebSphere Process Server között.

A SOAL keretrendszer a magyar rendőrséggel végzett közös projektben is hasznosnak bizonyult. Ebben a projektben egy sportrendészeti nyilvántartást kellett kialakítani, amely sportrendezvényekről illetve sportlétesítményekből el- illetve kitiltott emberek adatait kezelte. Az egyes rendezvények szervezőinek beléptetéskor le kell ellenőrizniük minden egyes embert, hogy jogosult-e az adott rendezvényre illetve létesítménybe bejutni. Ezeket a lekérdezéseket online egy webszolgáltatáson keresztül kell a szervezőknek megtenniük. Ennek a webszolgáltatásnak az interfészét és szkeletonját, valamint a megfelelő konfigurációs fájlokat a SOAL keretrendszer segítségével generáltuk.

A harmadik és egyben legfontosabb alkalmazása a SOAL keretrendszernek egy a MÁV-val közösen végzett projekt volt. Egy új jegykiadó automata szoftverét kellett elkészítenünk. Az automata a háttérben egy jegyeladó rendszerrel és egy karbantartó rendszerrel kommunikál. Mindkét rendszer webszolgáltatásainak interfészét SOAL nyelven terveztük, és az implementáció szkeletonját, valamint a .NET és Java konfigurációkat is a SOAL leírásból generáltuk. A tervezési fázisban a jegyeladó rendszer webszolgáltatásának interfésze három hónapon keresztül hetente változott. A SOAL keretrendszer segítségével ezeket a változásokat könnyen és gyorsan lehetett követni, és a szolgáltatás interfészeinek különböző verzióit is könnyen lehetett karbantartani. Az elő- és utófeltételeket leíró szerződés aspektus is hasznosnak bizonyult a jegyeladó rendszer által küldött eredmények ellenőrzésére, ami folyamatos fejlesztés alatt állt ebben az időszakban. A SOAL keretrendszer nagyon produktívnak bizonyult, és rengeteg munkát megspórolt nekünk ebben a projektben.

A javasolt teljesítménymodell a webszolgáltatások válaszidejének becslésére egy friss eredmény, így valós projektekben még nem került felhasználásra.

A SOAL keretrendszer segítségével az egyetemi előadásaimhoz WSDL fájlokat és programkódokat is generáltam demonstrációs célokra. Ezeken kívül webszolgáltatásokkal és BPEL folyamatokkal kapcsolatos laboratóriumi feladatokat is készítettem a keretrendszer segítségével. A SOAL keretrendszer így az oktatásban is hasznosnak bizonyult.

Publikációk

Folyóiratcikkek (3)

[1] B. Simon, B. Goldschmidt, and K. Kondorosi, "A metamodel for the web services standards," *Journal of Grid Computing*, vol. 11, no. 4, pp. 735–752, 2013.

[2] B. Simon, B. Goldschmidt, and K. Kondorosi, "A platform independent access control metamodel for web services," *Periodica Polytechnica Electrical Engineering and Computer Science*, vol. 58, no. 3, pp. 93–108, 2014.

[3] B. Simon, B. Goldschmidt, and K. Kondorosi, "A performance model for the web service protocol stacks (accepted paper)," *Services Computing, IEEE Transactions on*, vol. PP, pp. –, 2014.

Nemzetközi konferenciacikkek (11)

[4] B. Simon, B. Goldschmidt, P. Budai, I. Hartung, K. Kondorosi, Z. Laszlo, and P. Risztics, "Generic contract descriptions for web services implementations," in *ICDS 2011, The Fifth International Conference on Digital Society*, pp. 51–56, 2011.

[5] B. Simon, B. Goldschmidt, P. Budai, I. Hartung, K. Kondorosi, Z. Laszlo, and P. Risztics, "A metamodel of the ws-policy standard family," in *ICDS 2011, The Fifth International Conference on Digital Society*, pp. 57–62, 2011.

[6] B. Simon, Z. László, B. Goldschmidt, K. Kondorosi, and P. Risztics, "Evaluation of ws-* standards based interoperability of soa products for the hungarian e-government infrastructure," in *Digital Society, 2010. ICDS'10. Fourth International Conference on*, pp. 118–123, IEEE, 2010.

Független idézők száma: 7

[7] B. Simon and B. Goldschmidt, "A human readable platform independent domain specific language for wsdl," in *Networked Digital Technologies*, pp. 529–536, Springer Berlin Heidelberg, 2010.

[8] B. Simon, B. Goldschmidt, and K. Kondorosi, "A human readable platform independent domain specific language for bpel," in *Networked Digital Technologies*, pp. 537–544, Springer Berlin Heidelberg, 2010.

Független idézők száma: 1

[9] B. Simon, Z. László, and B. Goldschmidt, "Soa interoperability, a case study," *Proceedings of the IADIS International Conference, Informatics*, pp. 131–138, 2008.

Független idézők száma: 1

[10] B. Simon and Z. László, "A framework for traversing models using loops," in *Proceedings of the IADIS International Conference on Applied Computing 2007*, pp. 17–20, 2007.

[11] Z. László and B. Simon, "Model-based code generation with tree construction," *Proceedings on the IADIS International Conference on Applied Computing*, pp. 498–502, 2006.

[12] B. Simon, B. Goldschmidt, and S. Imre, "Engineering instance-migratable bpel business processes," in *Proceedings of the 15th International Multiconference INFORMATION SOCIETY – IS 2012*, pp. 233–236, Narodna in Univerzitetna Knjiznica, Ljubljana, Slovenia, 2012.

[13] B. Simon, B. Goldschmidt, and Z. László, "Bpel movie framework: Replaying bpel processes from logs," *Proceedings of the 13th IASTED International Conference, Software Engineering and Applications*, pp. 242–249, 2009.

[14] V. Müller, B. Simon, Z. László, and B. Goldschmidt, "Business monitoring solution for an e-government framework based on soa architecture," *Proceedings of the IASTED International Conference*, vol. 669, no. 039, p. 234, 2009.

Független idézők száma: 3

Magyar nyelvű konferenciacikkek (2)

[15] B. Simon, Z. Laszlo, and B. Goldschmidt, "Soa mintarendszer kialakításának tapasztalatai," in *Networkshop 2008, Dunaújváros, Magyarország*, pp. 1–8, 2008.

[16] B. Simon, B. Goldschmidt, and K. Kondorosi, "Webszolgáltatások kommunikációs overhead-jének becslése," in *Networkshop 2012, Veszprém, Magyarország*, pp. 1–8, 2012.

Hivatkozások

[AdL08] Rubén González Antolín and Juan de Lara. A Model-Driven Approach to Service Performance Prediction and Analysis. In *Proceedings of 6th Workshop on System Testing and Validation*, STV08. Fraunhofer IRB, 2008.

[AGM08] Danilo Ardagna, Carlo Ghezzi, and Raffaela Mirandola. Model Driven QoS Analyses of Composed Web Services. In *Proceedings of the 1st European Conference on Towards a Service-Based Internet*, ServiceWave '08, pages 299–311, Berlin, Heidelberg, 2008. Springer-Verlag.

[CAB08] Mohamed Amine Chaâbane, Eric Andonoff, and Rafik Bouaziz. Dealing with business process evolution using versions. In *ICE-B*, pages 267–278, 2008.

[CBZ+04] Fei Cao, Barrett R. Bryant, Wei Zhao, Carol C. Burt, Rajeev R. Raje, Andrew M. Olson, and Mikhail Auguston. A meta-modeling approach to web services. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '04, pages 796–, Washington, DC, USA, 2004. IEEE Computer Society.

[CMV04] V. De Castro, E. Marcos, and B. Vela. Representing wsdl with extended uml, 2004.

[DNsmGW08] C. Dumez, A. Nait-sidi moh, J. Gaber, and M. Wack. Modeling and specification of web services composition using uml-s. In *Next Generation Web Services Practices, 2008. NWESP '08. 4th International Conference on*, 2008.

[DVRS03] Paulo Dias, Pedro Vieira, and António Rito-Silva. Dynamic evolution in workflow management systems. In *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, pages 254–260. IEEE, 2003.

[EES10] A. Elgammal and M. El-Sharkawi. Using uml to model web services for automatic composition. Open Access publications from Tilburg University urn:nbn:nl:ui:12-4561873, Tilburg University, 2010.

[EKW+09] Thomas Erl, Anish Karmarkar, Priscilla Walmsley, Hugo Haas, L Umit Yalcinalp, Kevin Liu, David Orchard, Andre Tost, and James Pasley. *Web service contract design and versioning for SOA*. Prentice Hall, 2009.

[ESBS07] Julian Eckert, Stefan Schulte, Rainer Berbner, and Ralf Steinmetz. An Approach for Capacity Planning of Web Service Workflows. In *13th Americas Conference on Information Systems*, AMCIS 2007, 2007.

[ESS12] ESSI WSMO working group. *Web Service Modeling Ontology (WSMO)*. `http://www.wsmo.org/`, accessed: 29.01.2012.

[FM14] Greg Flurry and Manish Modh. *Web services development patterns*. `http://www.ibm.com/developerworks/websphere/library/techarticles/0511_flurry/0511_flurry.html`, accessed: July 13, 2014.

[GSSO04] Roy Gronmo, David Skogan, Ida Solheim, and Jon Oldevik. Model-driven web services development. In *The 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-04)*, 2004.

[IKLC10] G. Imre, M. Kaszó, T. Levendovszky, and H. Charaf. A Novel Cost Model of XML Serialization. *Electron. Notes Theor. Comput. Sci.*, 261:147–162, February 2010.

[JB08] Harshavardhan Jegadeesan and Sundar Balasubramaniam. An mof2-based services metamodel. *Journal of Object Technology*, 7(8):71–96, 2008.

[JRB+06] Matjaz B. Juric, Ivan Rozman, Bostjan Brumen, Matjaz Colnaric, and Marjan Hericko. Comparison of performance of Web services, WS-Security, RMI, and RMI-SSL. *J. Syst. Softw.*, 79(5):689–700, May 2006.

[JRH00] Matjaz B Juric, Ivan Rozman, and Marjan Hericko. Performance comparison of CORBA and RMI. *Information and Software Technology*, 42(13):915 – 933, 2000.

[KG99] Markus Kradolfer and Andreas Geppert. Dynamic workflow schema evolution based on workflow type versioning and workflow migration. In *Cooperative Information Systems, 1999. CoopIS'99. Proceedings. 1999 IFCIS International Conference on*, pages 104–114. IEEE, 1999.

[KKK08] Jong Myoung Ko, Chang Ouk Kim, and Ick-Hyun Kwon. Quality-of-service oriented web service composition algorithm and planning architecture. *Journal of Systems and Software*, 81(11):2079 – 2090, 2008.

[Klu08] M. Klusch. *CASCOM - Intelligent Service Coordination in the Semantic Web*, chapter 3. Birkhäuser Verlag, Springer, 2008.

[KPKH05] Vladimir Kolovski, Bijan Parsia, Yarden Katz, and James Hendler. Representing web service policies in owl-dl. In *In International Semantic Web Conference (ISWC*, pages 6–10, 2005.

[Mic14] Microsoft. *Windows Communication Foundation*. `http://msdn.microsoft.com/en-us/library/dd456779%28v=vs.110%29.aspx`, accessed: May 11, 2014.

[MM07] Moreno Marzolla and Raffaela Mirandola. Performance prediction of web service workflows. In *Proceedings of the Quality of software architectures 3rd international conference on Software architectures, components, and applications*, QoSA'07, pages 127–144, Berlin, Heidelberg, 2007. Springer-Verlag.

[NSP+10] Marc Novakouski, Soumya Simanta, Gunnar Peterson, Ed Morris, and Grace Lewis. *Performance Analysis of WS-Security Mechanisms in SOAP-Based Web Services*. http://www.sei.cmu.edu/reports/10tr023.pdf, November 2010.

[OMG09] OMG. *Service oriented architecture Modeling Language (SoaML)*. `http://www.omg.org/spec/SoaML/Current`, 2009.

[Ora14] Oracle. *Metro*. `http://metro.java.net/`, accessed: May 11, 2014.

[PSSvdA07] M. Pesic, M.H. Schonenberg, N. Sidorova, and W.M.P. van der Aalst. Constraint-based workflow models: Change made easy. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, volume 4803 of *Lecture Notes in Computer Science*, pages 77–94. Springer Berlin Heidelberg, 2007.

[PWH08] Baba Piprani, Chong Wang, and Keqing He. A metamodel for enabling a service oriented architecture. In *Proceedings of the OTM Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems: 2008 Workshops: ADI, AWeSoMe, COMBEK, EI2N, IWSSA, MONET, OnToContent + QSI, ORM, PerSys, RDDS, SEMELS, and SWWS*, pages 668–677, Berlin, Heidelberg, 2008. Springer-Verlag.

[QN10] Xhevi Qafmolla and Viet Cuong Nguyen. Automation of web services development using model driven techniques. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 3, pages 190 –194, feb. 2010.

[REB11] Douglas Rodrigues, Julio C. Estrella, and Kalinka R. L. J. C. Branco. Analysis of Security and Performance Aspects in Service-Oriented Architectures. *International Journal of Security and Its Applications*, 5(1):13–30, January 2011.

[RMR10] S. Rinderle-Ma and M. Reichert. Advanced migration strategies for adaptive process management systems. In *Commerce and Enterprise Computing (CEC), 2010 IEEE 12th Conference on*, pages 56–63, Nov 2010.

[RRKD05] MU Reichert, SB Rinderle, Ulrich Kreher, and Peter Dadam. Adaptive process management with adept2 (tool demo). 2005.

[Som11] Ian Sommerville. *Software Engineering*. International Computer Science Series. Pearson, 2011.

[SSFG04] Satoshi Shirasuna, Aleksander Slominski, Liang Fang, and Dennis Gannon. Performance Comparison of Security Mechanisms for Grid Services. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 360–364, Washington, DC, USA, 2004. IEEE Computer Society.

[SSVS04] Natenapa Sriharee, Twittie Senivongse, Kunal Verma, and Amit Sheth. On using ws-policy, ontology, and rule reasoning to discover web services. In Finn Arve Aagesen, Chutiporn Anutariya, and Vilas Wuwongse, editors, *Intelligence in Communication Systems*, volume 3283 of *Lecture Notes in Computer Science*, pages 246–255. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-30179-0_22.

[VAG05] Kunal Verma, Rama Akkiraju, and Richard Goodwin. R.: Semantic matching of web service policies. In *Proceedings of the Second Workshop on SDWP, 2005*, pages 79–90, 2005.

[vEZ08] Robert A. van Engelen and Wei Zhang. An Overview and Evaluation of Web Services Security Performance Optimizations. In *Proceedings of the 2008 IEEE International Conference on Web Services*, ICWS '08, pages 137–144, Washington, DC, USA, 2008. IEEE Computer Society.

[Voe08] Markus Voelter. A family of languages for architecture description. In *in 8th OOPSLA Workshop on Domain-Specific Modeling (DSM'08)*, 2008.

[W3C12a] W3C. *OWL-S: Semantic Markup for Web Services*. `http://www.w3.org/Submission/OWL-S/`, accessed: 29.01.2012.

[W3C12b] W3C. *Semantic Annotations for WSDL and XML Schema (SAWSDL)*. `http://www.w3.org/TR/sawsdl/`, accessed: 29.01.2012.

[WSO06] Hiroshi Wada, Junichi Suzuki, and Katsuya Oba. A model-driven development framework for non-functional aspects in service oriented grids. In *Proceedings of the International Conference on Autonomic and Autonomous Systems*, ICAS '06, pages 30–, Washington, DC, USA, 2006. IEEE Computer Society.

[WSO08] Hiroshi Wada, Junichi Suzuki, and Katsuya Oba. A model-driven development framework for non-functional aspects in service oriented architecture. *International Journal of Web Services Research (IJWSR)*, 5(4):1–31, 2008.

[ZHKL10] Sonja Zaplata, Kristof Hamann, Kristian Kottke, and Winfried Lamersdorf. Flexible execution of distributed business processes based on process instance migration. *Journal of Systems Integration*, 1(3):3–16, 2010.

[ZS09] Uwe Zdun and Mark Strembeck. Reusable architectural decisions for dsl design. In *Proceedings of the 14th Annual European Conference on Pattern Languages of Programming (EuroPLoP)*, Aachen, Germany, 2009.