Webszolgáltatások kommunikációs overhead-jének becslése

Simon Balázs, sbalazs@iit.bme.hu
Dr. Goldschmidt Balázs, balage@iit.bme.hu
Dr. Kondorosi Károly, kondor@iit.bme.hu
Budapesti Műszaki Egyetem, Irányítástechnika és Informatika Tanszék
1117 Budapest, Magyar tudósok krt. 2

Absztrakt

A webszolgáltatások elosztott kommunikációt valósítanak meg különböző platformok között. A WS-* szabványok címzéssel, megbízható üzenetküldéssel, titkosított adatcserével és egyéb middleware aspektusokkal bővítik a webszolgáltatásokon alapuló kommunikációt. Azonban ezeknek a szabványoknak jelentős befolyása van a kommunikáció válaszidejére. Jelen cikk célja a webszolgáltatások és a WS-* szabványok által okozott kommunikációs overhead kimérése, illetve egy olyan teljesítménymodell definiálása, amely segítségével tetszőleges interfészű webszolgáltatás válaszideje becsülhetővé válik.

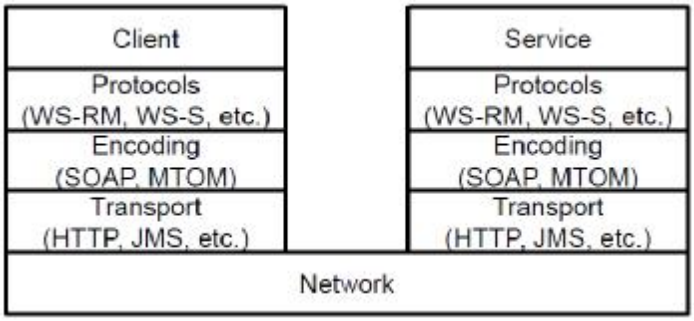
1 Bevezetés

A webszolgáltatások elosztott kommunikációt valósítanak meg oly módon, hogy a kommunikációs protokoll (SOAP), az interfészleíró nyelv (WSDL) és a middleware aspektusok (WS-* szabványok: WS-Addressing, WS-ReliableMessaging, WS-Security, WS-SecureConversation, stb.) mind XML alapúak. Azonban az XML formátum nagy overhead-et ró a kommunikációs csatornára.

Célunk az, hogy a webszolgáltatások által okozott kommunikációs overhead-et kimérjük, és egy olyan teljesítménymodellt alakítsunk ki, amely segítségével ez az overhead előre becsülhetővé válik. Ehhez olyan reprezentatív teszteseteket hoztunk létre, amelyekkel a válaszidő a különböző programnyelvi típusokra és a különböző WS-* protokollokra kimérhetővé válik. A méréseket Java és .NET platformon is elvégeztük.

A cikk a következő módon épül fel. A második szakasz bemutatja a webszolgáltatásokat implementáló keretrendszerek tipikus architektúráját és összegyűjti azokat a tényezőket, amelyek hatással lehetnek a kommunikációs overhead-re. A harmadik szakasz a kialakított teszteseteket ismerteti. A negyedik szakasz a mérési eredményeket tartalmazza, illetve a mérésekből levont tanulságokat veszi számba. Az ötödik szakasz a mérések alapján kialakított teljesítménymodellt írja le, amely segítségével tetszőleges interfészű webszolgáltatás kommunikációs overhead-je becsülhetővé válik a mérésekből kiszámolt együtthatók segítségével. Az eredményeket az ötödik szakasz foglalja össze és egyben ismerteti a továbbfejlesztési lehetőségeket.

2 Webszolgáltatások implementációs architektúrája



1. ábra – Webszolgáltatások implementációs architektúrája

Az 1. ábra a webszolgáltatásokat implementáló eszközök általános architektúráját mutatja. A Windows Communication Foundation [1] (a .NET környezetből) és a Metro [2] (a JAX-WS referencia implementációja) csatornamodellek is ezt a struktúrát követik, és más keretrendszerek is hasonlóan modellezhetők.

A rétegszerkezet legalján a hálózat (**network**) található, amely bájtok átvitelét végzi a kliens és a szolgáltatás között. Webszolgáltatások esetén ez tipikusan a HTTP szokott lenni, azonban általában ez lecserélhető más protokollokra is. A transzport (**transport**) réteg felelős a hálózati protokoll kezeléséért, vagyis a bájtok átküldéséért és fogadásáért. A kódoló (**encoding**) réteg feladata az átküldött bájtok és a felsőbb rétegekben használt üzenetobjektum közötti transzformáció, vagyis a sorosítás. A transzport és kódoló rétegek kötelezően mindig jelen vannak. A felsőbb protokoll (**protocol**) rétegek opcionálisak. Ezek felelnek a különböző WS-* szabványok (WS-ReliableMessaging, WS-Security, stb.) implementálásáért.

Mivel a webszolgáltatások XML-re épülnek, átjárást biztosítanak különböző programnyelvek és keretrendszerek között. Ennek azonban ára van: az XML kezelése, titkosítása, digitális aláírása és sorosítása nagyon nagy overhead-et ró a kommunikációra. Ez az overhead a lokális hívásokhoz és a bináris alapú távoli eljáráshívásokhoz képest jóval nagyobb terhet jelent.

Az egyes rétegek a következőképpen járulnak hozzá a kommunikációs overhead-hez. A transzport réteg független a felsőbb rétegektől, a válaszidőhöz az üzenet bájtban számolt méretével arányosan ad hozzá. A kódoló réteg felelős a sorosításért, így ennek a hozzájárulása elsősorban az üzenetben felhasznált primitív típusoktól (boolean, int, double, string stb.), tömböktől és összetett típusoktól függ. Ezen kívül e réteg válaszidejét befolyásolja a string-ek hossza, a tömbök mérete és a struktúrák mélysége is. A protokoll rétegek a kliens és a szolgáltatás közötti kapcsolat felépítése során úgynevezett bootstrap üzeneteket is cserélhetnek egymással, amelyek jelentősen megnövelhetik a válaszidőt. Titkosító protokollok esetén pedig az XML titkosítás és digitális aláírás további terhet jelent. Jelen cikk az itt felsorolt tényezők hozzájárulásának mérésével és becslésével foglalkozik.

3 Tesztesetek

Az előző szakaszban bemutatott architekturális rétegszerkezetből adódnak azok a tényezők, amelyek befolyásolják a webszolgáltatás hívások válaszidejét. Ezek alapján a tényezők alapján meghatározhatók azok a tesztesetek, amelyek segítségével reprezentatív méréseket lehet végezni az egyes implementációs keretrendszerekkel kapcsolatban. A mérések alapján megbecsülhető az egyes keretrendszerek hozzájárulása a kommunikációs overhead-hez.

A már bemutatott tényezők alapján a következő teszteseteket alakítottuk ki. A legtöbb programnyelv a következő primitív típusokat definiálja: boolean, byte, int, long, float, double, string. Feltehetjük tehát, hogy ezek alkotják az összetettebb típusokat is, amelyek tömbök vagy struktúrák lehetnek. A mérések pontosabbak, ha nagyobb méretű adathalmazon dolgozunk, éppen ezért a primitív típusokat különböző mélységű struktúrákba és különböző hosszúságú tömbökbe csomagoltuk. Ennek megfelelően minden egyes primitív típusra két fajta szolgáltatás készült:

1. Egy szolgáltatás egy operációval, mindegyik az adott primitív típusból álló tömböt használ
2. Egy szolgáltatás egy operációval, mindegyik az adott primitív típusból alkotott láncolt listákból álló tömböt használ

A tömbök használata segíti elő azt, hogy a méréseket nagyobb adatmennyiségeken is el lehessen végezni. Összesen tehát 14 szolgáltatás készült el, mindegyik 1-1 operációt tartalmaz.

A felsorolt tesztesetek elegendőek a transzport és kódoló rétegek overhead-jének becsléséhez. Azonban a WS-* szabványok további jelentős hozzájárulást adnak a válaszidőhöz. Éppen ezért minden egyes fenti szolgáltatást a következő WS-* protokolloknak megfelelően öt-öt külön végponton konfiguráltunk:

1. nincs plusz WS-* protokoll
2. WS-Addressing 1.0
3. WS-Addressing 1.0, WS-ReliableMessaging 1.1
4. WS-Addressing 1.0, WS-Security 1.0, Basic Security Profile 1.0
5. WS-Addressing 1.0, WS-Security 1.0, WS-Trust 1.3, WS-SecureConversation 1.3, Basic Security Profile 1.0

Ezen túlmenően minden egyes végpont a SOAP 1.1 és SOAP 1.2 protokollokkal, valamint az MTOM mehanizmus engedélyezésével és tiltásával is kombinálva volt.

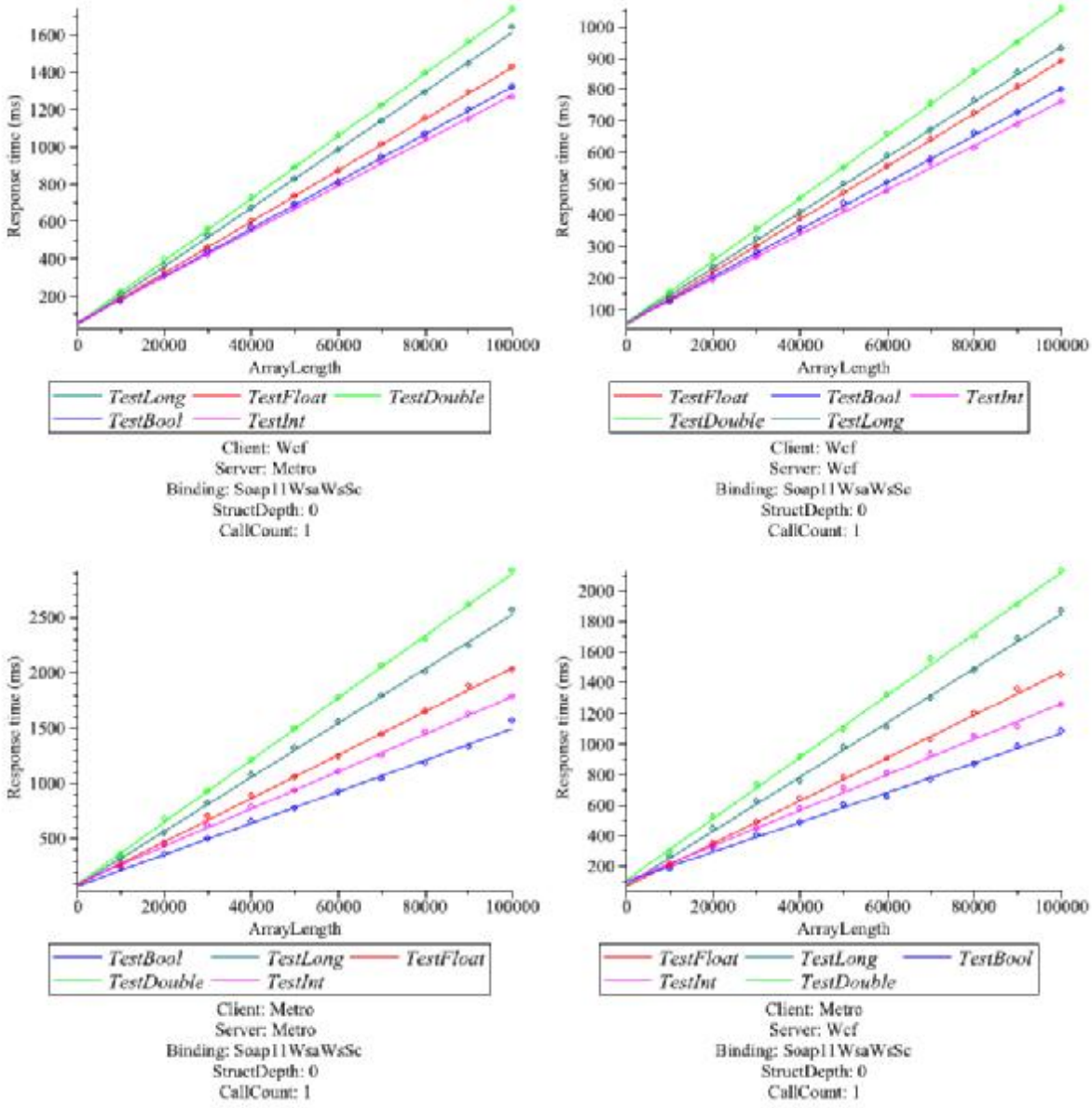
Összesen tehát 14*5*2*2=280 szolgáltatási végpont készült, és mindegyikhez implementáltunk egy-egy klienst is. A méréseket két keretrendszerben (WCF és Metro) végeztük, mindegyikben mind a 280 szolgáltatást és a hozzájuk tartozó 280 klienst is elkészítettük úgy, hogy a kliensek a másik keretrendszer szolgáltatásait is meg tudják hívni.

4 Mérési eredmények

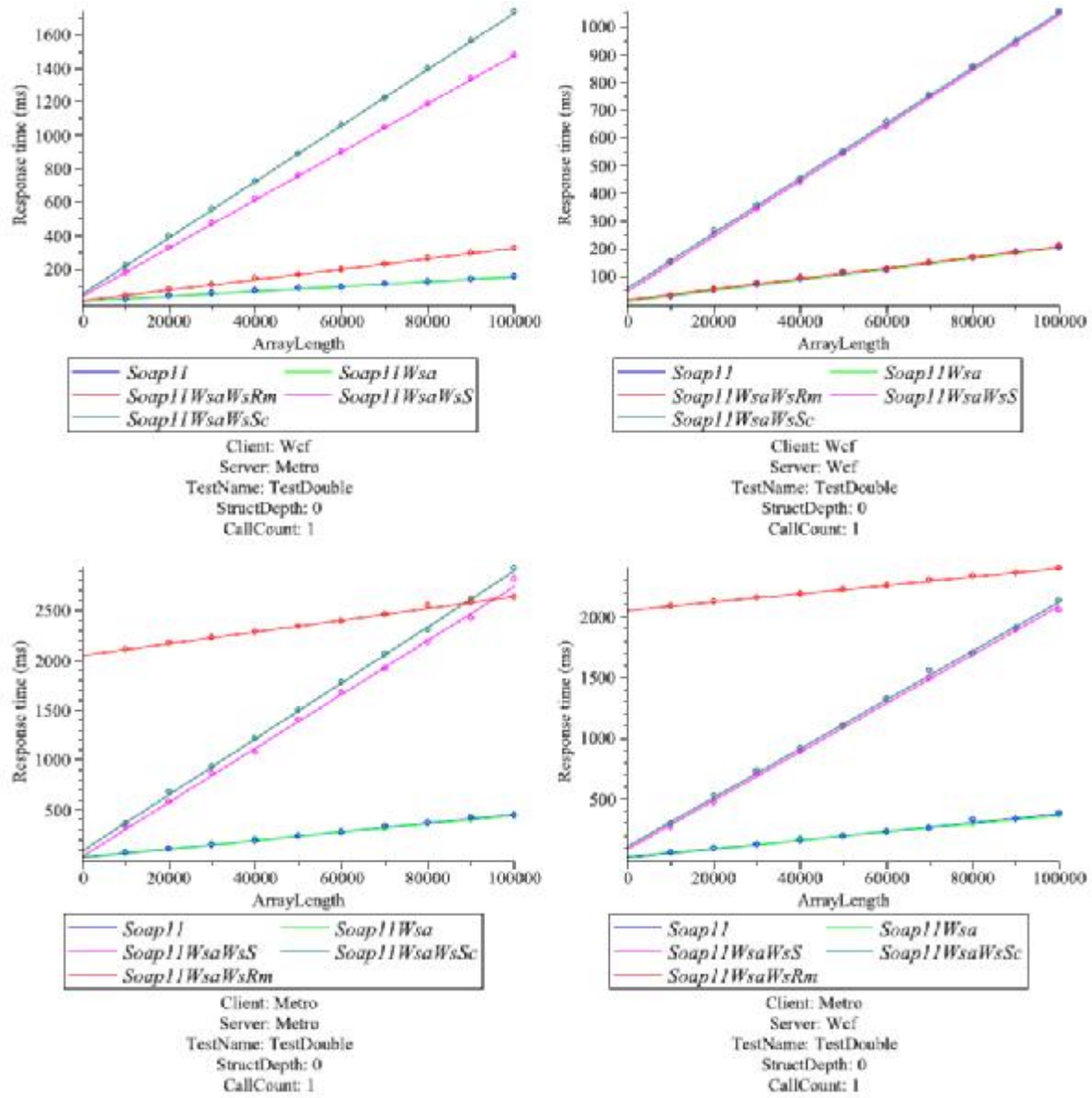
Ez a szakasz a mérési eredményeket foglalja össze. A méréseket a következő konfiguráción végeztük:

- AMD Phenom II X4 955 Black Edition 3.2 GHz CPU
- 12 GB RAM
- Microsoft Windows 7 Professional SP1 64 bit
- Microsoft .NET 4.0, WCF, IIS szerver 7.5
- Oracle JRE 7 és JDK 7, GlassFish szerver 3.1.1 Open Source Edition Full Platform

Terjedelmi okokból a mérési eredményekről csak néhány reprezentatív ábrát (2. és 3. ábra) illesztettünk be.



2. ábra – Válaszidők WS-SecureConversation esetén különböző primitív típusokra



3. ábra – Válaszidők double típus esetén különböző WS-* protokollokra

A mérésekből a következő tanulságok szűrhetők le:

1. A különböző keretrendszerek különböző válaszidőket produkálnak, azonban karakterisztikájuk ugyanolyan az egyes tényezőket tekintve
2. A SOAP verziónak nincs hatása a válaszidőre
3. Ha nincsenek byte tömbök, az MTOM használatának nincs hatása a válaszidőre
4. Az üzenet (nettó vagy bruttó) mérete nem elegendő a válaszidő becsléséhez
5. A primitív típusok befolyásolják a válaszidőt
6. A WS-* protokollok jelentősen hozzájárulnak a válaszidőhöz
7. A WS-Addressing protokollnak nincs számottevő hatása

8. A legnagyobb válaszidőket az elvárásoknak megfelelően a WS-ReliableMessaging, WS-Security és WS-SecureConversation szabványok produkálják
9. A válaszidő a tömb hosszával lineárisan arányos.
10. A válaszidő a struktúrák mélységével lineárisan arányos.
11. A válaszidő a string-ek hosszával lineárisan arányos.
12. A válaszidő a hívások számával lineárisan arányos.

5 Válaszidő becslése

Az előző szakaszban ismertetett mérési eredmények és tapasztalatok alapján kialakítottunk egy olyan teljesítménymodellt, amely segítségével tetszőleges interfészű webszolgáltatás válaszidejének overhead-je becsülhetővé válik.

A teljesítménymodellben az első lépés a mérési eredményeket közelítő függvény definiálása. Legyen ez a **válaszidő függvény**, amelyet a következőképpen definiálunk:

$$r_U(k,l,m) = (a_{U,k}k + b_{U,k})(a_{U,l}l + b_{U,l})(a_{U,m}m + b_{U,m})$$

Ahol:

- *U = (ClientFramework x ServerFramework x Binding x Type)*, vagyis a mérésben szereplő kliens keretrendszer, szerver keretrendszer, kommunikációs protokoll és a felhasznált primitív típus Descartes-szorzata
- *k* a tömb hossza
- *l* a string-ek hossza
- *m* a struktúrák mélysége
- $a_{U,k}$, $b_{U,k}$, $a_{U,l}$, $b_{U,l}$, $a_{U,m}$, $b_{U,m}$ a mérések lineáris regressziójával kapott együtthatók

Ezeket a válaszidő függvényeket kell áttranszformálni olyan függvényekké, amelyek csak egy keretrendszertől függenek. Ezáltal meghatározható az adott keretrendszer karakterisztikája. Ehhez szükség van a **bootstrap overhead függvényre** és a **hívás overhead függvényre**.

Legyen a **bootstrap overhead függvény** a következő (ez valójában egy konstans, mivel nem függ a tömb hosszától, a string-ek hosszától és a stuktúrák mélységétől):

$$b_V$$

A **hívás overhead függvény** pedig a következő:

$$c_W(k,l,m) = (a_{W,k}k + b_{W,k})(a_{W,l}l + b_{W,l})(a_{W,m}m + b_{W,m})$$

Ahol:

- *V = (Framework x Side x Binding)*, vagyis a keretrendszer, az oldal (kliens vagy szerver) és a protokoll Descartes-szorzata
- *W = (Framework x Side x Binding x Type)*, vagyis a keretrendszer, az oldal (kliens vagy szerver), a protokoll és a felhasznált primitív típus Descartes-szorzata

A fentiek alapján a **válaszidő függvény** a következőképpen írható fel:

$$r_U = b_{V_c} + b_{V_s} + \left[c_{W_c}(k,l,m) + c_{W_s}(k,l,m)\right] \cdot n$$

Ahol:

- *U = (ClientFramework x ServerFramework x Binding x Type)*
- V_c *= (ClientFramework x ClientSide x Binding)*
- V_s *= (ServerFramework x ServerSide x Binding)*
- W_c *= (ClientFramework x ClientSide x Binding x Type)*
- W_s *= (ServerFramework x ServerSide x Binding x Type)*

Az egyenleteket felírva a **bootstrap-** és a **hívás overhead függvények** előállíthatók. Ezek a függvények pedig már csak egyetlen keretrendszertől függnek, így segítségükkel az adott keretrendszer karakterisztikája számítható:

$$t_{Framework \times Side \times Binding} = b_{Framework \times Side \times Binding} + \left[\sum_{Types \in PrimitiveTypes} c_{Framework \times Side \times Binding \times Type}\left(k_{Type}, l_{Type}, m_{Type}\right)\right] \cdot n$$

Ennek a függvénynek a segítségével tetszőleges interfészű és protokollú webszolgáltatás válaszideje becsülhetővé válik.

6 Összefoglalás

A webszolgáltatások ugyan platformfüggetlen kommunikációt tesznek lehetővé, azonban az XML-re épülő WS-* szabványok nagy overhead-et rónak a hívásokra. Ez az overhead nagyságrendekkel nagyobb lehet, mint magának az alkalmazáslogikának a futásideje. Éppen ezért fontos, hogy ezt az overhead-et becsülni tudjuk, mivel ezáltal a szolgáltatások interfésze és a kommunikációs protokollok optimálisabban választhatók meg.

Jelen cikkben bemutattuk a webszolgáltatásokat implementáló keretrendszerek tipikus architektúráját, illetve az architektúra elemeiből fakadó tényezőket, amelyek hatással vannak a kommunikációs overhead-re. A tényezők alapján meghatároztuk azokat a reprezentatív teszteseteket, amelyek segítségével kimérhető az egyes keretrendszerek válaszideje. A méréseinket két keretrendszerben (WCF és Metro) illetve a két keretrendszer között is végrehajtottuk, és a cikkben ismertettük a legfontosabb tanulságokat, amelyek a mérésekből adódtak. A tanulságok alapján pedig kidolgoztunk egy teljesítménymodellt. A teljesítménymodell alkalmas az egyes keretrendszerek válaszidejének becslésére.

A továbbiakban más keretrendszerekre (IBM, Oracle, JBoss, Apache CXF) is ki fogjuk terjeszteni a méréseket, és megvizsgáljuk, hogy azok is hasonló karakterisztikát mutatnak-e.

Köszönetnyilvánítás

A munka szakmai tartalma kapcsolódik a "Új tehetséggondozó programok és kutatások a Műegyetem tudományos műhelyeiben" c. projekt szakmai célkitűzéseinek megvalósításához. A projekt megvalósítását a TÁMOP - 4.2.2.B-10/1--2010-0009 program támogatja.

Referenciák

[1] Microsoft: Windows Communication Foundation, http://msdn.microsoft.com/en-us/netframework/aa663324, utolsó letöltés: 2012. március 3.

[2] Oracle: Metro, http://metro.java.net/, utolsó letöltés: 2012. március 3.