# SOA rendszerek felügyelete és vizualizációja

## 1 Bevezetés

A gazdasági, üzleti, pénzügyi élet alapja, hogy különböző érdekeltségben lévő erőforrások együttműködnek egymással. Sok esetben ez az együttműködés nem elég hatékony, és csak korlátozott mértékben jön létre. Mivel az említett erőforrásokat kezelő számítógépes rendszerek általában különböző felépítésűek és más szoftvereket használnak, nincs szabványos módja ezen rendszerek között az adatok megosztásának. Továbbá az erőforrásokon végzett műveletek gyakran több implementációban is léteznek. Ez felesleges redundanciát jelenthet. Egy olyan vállalati rendszer, melyre az előzőek teljesülnek, magas karbantartási költségekkel jár, nem hatékony, ugyanis bizonyos műveletek többször is implementálva vannak, az erőforrások közötti együttműködés pedig kétséges. Különböző erőforrásokat használó valós idejű műveletek elvégzése is nehezen megvalósítható a szabványos kommunikáció hiányában.

A többszörösen megvalósított műveletek helyett célszerű egy implementációt készíteni és mindenhol azt használni. Emellett szem előtt kell tartani azt is, hogy a vállalati rendszer akkor tud hatékonyan működni, ha az általa használt erőforrásokkal teljes mértékben, valós időben együtt tud működni. A Szolgáltatás Orientált Architektúra (SOA) erre az elgondolásra építve vezeti be a szolgáltatásokat. A szolgáltatás az erőforrásokon végezhető műveleteket elérhetővé teszi mások számára egy egységes interfészen keresztül, melynek során a kommunikáció és a szolgáltatás leírása elterjedt szabványokra épül (pl. SOAP, WSDL). Ezáltal egy erőforráson végezhető műveletet elég egyszer implementálni, majd elérhetővé tenni egy szolgáltatás létrehozásával. Ha valaki használni akarja az erőforrást, akkor azt a szolgáltatáson keresztül tudja megtenni. Szolgáltatások segítségével egységesen lehet összekapcsolni különböző rendszereket az elterjedt szabványok használatának köszönhetően. A szolgáltatások újrafelhasználhatóságának köszönhetően csökken a fejlesztési idő, az erőforráson végzett minden művelet a szolgáltatáson keresztül történik (konzisztencia), és a karbantartás is egyszerűbbé válik.

Minél több szolgáltatás van összekapcsolva egymással, annál összetettebbé, átláthatatlanabbá válik a rendszer. Ez nehezíti a rendszer felügyeletét, monitorozását, ugyanis hiányzik egy olyan magas szintű nézet, ahol látható a szolgáltatások hálója, a köztük zajló

kommunikáció. Az optimális erőforrás kihasználás végett szükség van olyan adatokra, mint pl. a szűk keresztmetszetet képező kapcsolatok vagy a túlterhelt szolgáltatások megjelenítése. A felügyeleti képesség hiányában további problémát jelenthet a rendszerben bekövetkező hibásan működő szolgáltatás megtalálása.

Az újrahasználható szolgáltatások nagy flexibilitást nyújtanak az alkalmazások létrehozásához laza csatolásuk miatt, ugyanis több alkalmazásban is fel tudjuk őket használni. Azonban pont ez a laza csatolás nehezíti meg a rendszer monitorozását. 1970-es évek mainframe rendszerei óta jól kiforrott alkalmazási, hálózati és infrastrukturális felügyeleti rendszerek jöttek létre. A SOA lazán csatolt szolgáltatásai esetén ezek a módszerek nem használhatóak, ugyanis ott nem tudják biztosítani a tranzakciók láthatóságát, integritását és helyreállítási lehetőségeit.

# 2 Felügyeleti lehetőségek

A SOA rendszerek felügyeletére két elterjedt nézet jött létre. Az egyik nézet szerint a problémát célszerű aszerint megközelíteni, hogy a SOA bevezetése és használata mit is hivatott kiszolgálni: az üzleti tevékenységeket. Mivel az üzleti célok elérése és a szolgáltatás minőségének fenntartása érdekében a szolgáltatási szint szerződéseket (SLA) kell teljesíteni, ezért előnyös lehet, ha a felügyeleti eszköz is lehetővé teszi ennek monitorozását végponttól végpontig. Az ezzel kapcsolatos irányzatok:

- **Business Service Management**: üzlet- és ügyfél-orientált megközelítése a szolgáltatás menedzsmentnek.
- **Business Transaction Management**: SOA környezetben az egész üzleti folyamatra biztosított tranzakció menedzsment
- **Business Activity Monitoring**: Az egyes komponensek aktivitásának és teljesítményének monitorozása, hogy felismerjük a rendszerben a szűk keresztmetszeteket.
- **Business Process Management**: az üzleti folyamatok hatékonyságáért és azok folyamatos optimalizálásáért, javításáért felel.

A másik megközelítés szerint a SOA lazán kapcsolt szolgáltatásai közötti interakciókat kell megfigyelni: így követhető az is, hogy egy újonnan rendszerbe kapcsolt komponens mennyire terheli le a már létező szolgáltatásokat. Fontos, hogy a monitorozás ténylegesen túlmutasson ezeken a lazán csatolt határokon, ugyanis csak így biztosítható a teljes rendszer menedzselése. E szerint a nézet szerint a SOA rendszerek felügyeletének kihívásait úgy lehet leküzdeni, ha a szolgáltatásokat a monitorozásra és menedzsmentre felkészítve hozzuk létre. Egy olyan

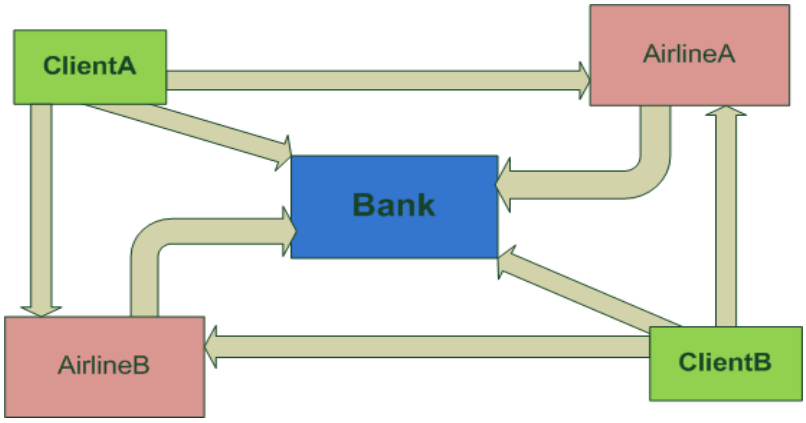
szolgáltatás, ami nem szolgáltat semmilyen információt a saját állapotáról vagy a rajta futó tranzakciókról olyan, mint egy fekete doboz: a szolgáltatás használója illetve a rendszert monitorozó adminisztrátor nem tudják megállapítani, hogy a szolgáltatás megfelelően működik-e. A felügyeleti képesség az architektúrába a kezdetektől be kell legyen építve, ugyanis így valósítható meg, hogy folyamatosan reális képünk legyen a rendszerről.

# 3 Mintaalkalmazás

A vizsgált probléma szemléltetésére egy mintaalkalmazás kerül ismertetésre, a későbbiekben pedig ezen az alkalmazáson kerül bemutatásra a monitorozást biztosító megoldás.

A mintaalkalmazás egy egyszerűsített repülőjegy vásárlást támogató rendszert reprezentál. Vannak légitársaságok és egy bank. A klienseknek számlája van a bankban, aminek lekérdezhetik az egyenlegét, pénzt rakhatnak rá, vagy ki is vehetnek. A légitársaságoktól a kliensek lekérhetik a repülőgép járatok adatait, jegyet foglalhatnak járatokra, lemondhatják a rendelésüket, illetve megtekinthetik a korábbi foglalásaikat. A rendelések foglalásakor/lemondásakor a légitársaság ellenőrzi a bankon keresztül, hogy az adott felhasználónak megfelelő fedezete van-e a foglalásra, és amennyiben igen, akkor levonja a számlájáról a megfelelő összeget (foglalás lemondása esetén pedig visszahelyezi rá). Természetesen a mintaalkalmazás csak a webszolgáltatások használatát, kapcsolataikat kívánja bemutatni, egyszerűségéből adódóan másra gyakorlatilag nem alkalmas.

Az interoperabilitás szem előtt tartása végett a rendszer WCF (.NET) és Apache CXF (Java) webszolgáltatás technológiákban készült el, bemutatva a két különböző szolgáltatásverem közötti együttműködést.

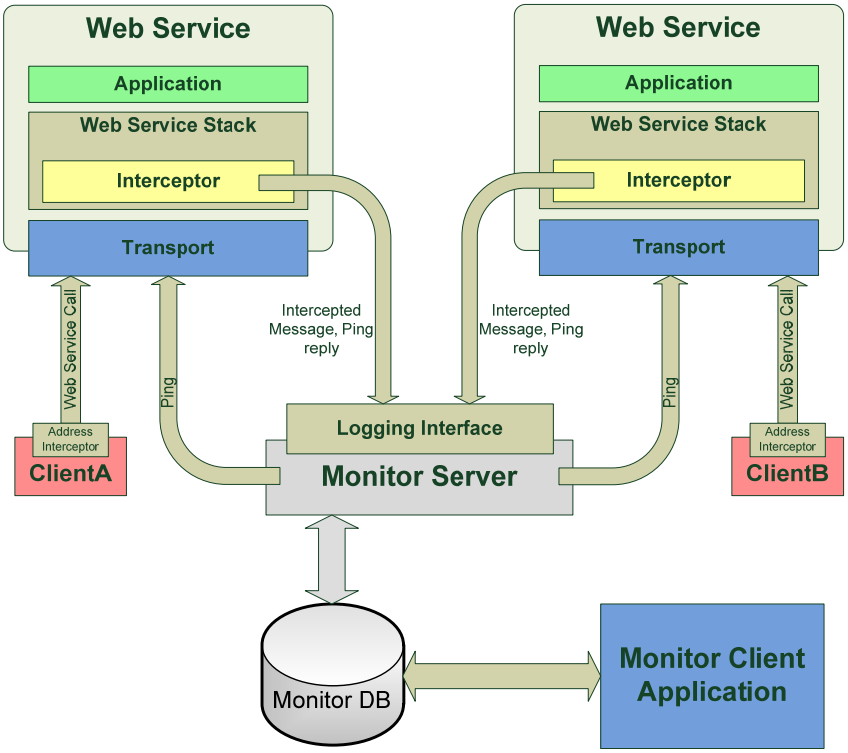



**1. ábra A repülőjegy vásárlást támogató rendszer kommunikációs sémája**

# 4 A létrehozott felügyeleti rendszer

A kidolgozott felügyeleti rendszer a korábban ismertetett felügyeleti lehetőségek közül az utóbbi alapján készült, miszerint a megfigyelési képességet célszerű SOA rendszerünk

architektúrájába a kezdetektől beépíteni, megkönnyítve a rendszer felügyeletét. Ha egy szolgáltatás némi adatot is biztosít a belső működésének állapotáról, máris egyszerűbb a monitorozása. Az előzőekben ismertetett mintaalkalmazás pedig azt a célt fogja szolgálni, hogy rajta keresztül kerül bemutatásra a kifejlesztett monitorozó rendszer működés közben.




**2. ábra A monitorozó rendszer felépítése**

Az ábrán két webszolgáltatás látható, amit két kliens hív meg. A monitorozó rendszer alapját az adja, hogy a szolgáltatásoknak a vermét kibővíti egy Interceptor réteg. Hagyományos esetben egy szolgáltatás úgy épül föl, hogy a szállítási réteg (pl. http, JMS stb.) fölött a webszolgáltatás-verme található, ami az üzenet fogadásáért/küldéséért, kódolásáért/dekódolásáért, a különböző protokollok és WS-* specifikációk implementálásért, valamint az alkalmazással való kommunikációért felel. A webszolgáltatás-verme fölött található maga az alkalmazás, illetve a webszolgáltatás interfészének implementációja.

Az **Interceptor** réteg a webszolgáltatás-vermének aljára épül be, amikor még a protokollok nagy része nem kerül értelmezésre. Azért található ezen a szinten, hogy ne kelljen figyelembe vennie a felsőbb rétegekben lévő protokollokat, illetve a „ping” típusú üzenet is független legyen a használt protokolloktól. Célja, hogy a szolgáltatásokat egy olyan réteggel bővítse ki, ami a felügyelethez szükséges plusz információt fogja szolgáltatni a monitorozó szerver számára. Üzenet érkezésekor értesíti a monitorozó szervert, valamint „ping” üzenetek elkapásakor „ping reply” választ küld a szervernek, és nem engedi az üzenet

feljebb található rétegekbe való eljutását. (Kliens oldalon kiegészítésként szerepel ez a réteg, hogy azonosítani lehessen az üzenet küldőjét).

A **monitorozó szervernek** a fő feladata a webszolgáltatások állapotainak nyilvántartása, valamint a köztük történő fő események adatbázisba történő rögzítése. Amikor egy szolgáltatásnak az Interceptora értesítést küld a monitorozó szervernek, az beírja az adatokat a hozzá tartozó adatbázisba. Így a tábla tartalma alapján bármikor megtekinthető, hogy adott időpontban milyen kommunikáció folyt a szolgáltatások között.

A **megjelenítő alkalmazás** célja, hogy az adatbázis segítségével egy átfogó képet adjon a rendszerről. Legfontosabb funkciója a webszolgáltatások és kapcsolataik megjelenítése. Mivel egy rendszerben akár nagyon sok szolgáltatás és kapcsolat is lehet, ezért a megjelenítés gráf formájában történik: a csúcsok a végpontok, valamint az élek jelzik, hogy mely végpontok között történik kommunikáció. A csúcsok valójában nem csak a végpontok lehetnek, hanem a megfelelő hierarchiaszint kiválasztásával a gépek és szerverek is. Továbbá a kliensek is feltüntetésre kerülnek, hogy pontosabb képet lehessen kapni a rendszerben történő kommunikációról. Mivel az adatbázisból kiolvashatóak a rendszer főbb eseményei, ezért egyszerű lekérdezésekkel fontos forgalmi statisztikákhoz is juthatunk. Ennek fényében a monitorozó alkalmazás diagramokon is megjelenít forgalmi adatokat.

# 5 Az implementáció részletei

## 5.1 Interceptorok

A szolgáltatások kibővítése 3 különböző technológia esetén került megvizsgálásra: Metro, Apache CXF és WCF. A megvalósított funkcionalitás mindhárom technológia esetén hasonló, csak annak módja különbözik. A létrehozott interceptorok is a szolgáltatásvermek ugyanazon szintjére kerülnek beillesztésre.

### 5.1.1 Java: Metro

A Metro szolgáltatásverem esetén csatornának hívják a különböző webszolgáltatás protokollok implementálásáért felelős rétegeket. Az üzenetek elkapására két lehetőség van: alkalmazás szinten handlerek használatával, valamint a csatorna láncba egy új csatorna (tube) beillesztésével. A két módszer között a fő különbség az, hogy alkalmazás szintű elkapás esetén nem kell törődni a különböző szolgáltatás protokollokkal, ugyanis a beérkező hívást a protokollok feldolgozása után, valamint a kimenő választ a protokollok hozzáadása előtt kapjuk meg. Esetünkben ezzel az a probléma, hogy csak azokról az üzenetekről értesülnénk, amik a protokollokon már átjutottak. Ezért az interceptort célszerű a másik módszerrel, saját

csatorna megírásával létrehozni. Ehhez a saját csatornán kívül az azt legyártani képest gyárat (factory) is meg kell írni, valamint a csatornalánc létrehozásáért felelős csatornalánc összerakót. A Metro korábbi verzióiban egy XML konfigurációs fájlban kellett megadni, hogy milyen csatornákkal szeretnénk kibővíteni a szolgáltatásvermet. Az új verzióban található csatornalánc összerakó ezt hivatott helyettesíteni, azonban azt nem sikerült működésre bírni, nem illesztődött be a saját csatorna. Emiatt a felügyeleti rendszerből a Metro-s implementáció kimaradt.

### 5.1.2 Java: Apache CXF

Mivel fontos cél volt a Java platformon is megvalósítani a szolgáltatásverem kibővítését, ezért a következő jelölt az Apache CXF webszolgáltatás keretrendszer volt. Az Apache CXF-ben az interceptorok jelentik a feldolgozás alapvető egységét. Amikor egy szolgáltatás meghívódik, létrejön az interceptor lánc és minden benne szereplő interceptor meghívódik: megkapják az üzenetet, feldolgozzák azt, majd továbbadják a következő interceptornak. A láncok fázisokra (Phase) vannak felosztva. Minden interceptornak a konstruktorában meg kell adnia, hogy melyik fázisban akar lefutni, illetve opcionálisan az interceptorok közötti sorrendjét is. A kibővítéshez saját interceptort kell írni, majd be kell illeszteni a megfelelő fázisba. Sajnos a CXF-ben a SOAP fejlécek kezelése némileg nehézkes, és mindenképp JAXB annotációkra is szükség van a korrekt XML reprezentáció biztosításához. Szerencsére a Metro-val ellentétben nem voltak nehézségek a létrehozott interceptor beillesztésével. A megfelelően felkonfigurált szolgáltatásokban az üzenetek elkapásra kerültek és a monitorozó szerver felé továbbítva lettek a szükséges információk.

A mintaalkalmazás egy légitársasága a CXF-fel lett megvalósítva, beépítve a megfelelő interceptorral mind kliens, mind szerver oldalon.

### 5.1.3 .NET: Windows Communication Foundation

A cross-platform működés bemutatása végett a WCF technológiával is meg lett valósítva az üzenetek elkapása és a csatornamodell kibővítése. Erre két lehetőség van: A szolgáltatás (vagy kliens) és a kötés (binding) között található szerződés (contract) szintjén vagy saját protokoll csatorna írásával. Utóbbi módszer segítségével a kötés kerül kibővítésre, és pontosan lehet vele kontrollálni, hogy milyen protokollok feldolgozása után/előtt történjen az üzenet elkapása. Működését tekintve a két módszer nagyban hasonlít a Metro csatornamodelljénél látottakhoz. Ahhoz hasonlóan, itt is a második módszer került alkalmazásra, ugyanis a szerződés szintjén túl magas szintű üzenet elkapás történik (amikor az üzenet már túljutott a protokollokon), valamint itt is érvényes, hogy nem lenne minden

üzenethez hozzáférésünk. Saját protokoll csatorna írásához számos osztályt kell írni (különböző csatornahasználattal kapcsolatos interfészek implementálása), azonban az interneten fellelhető dokumentációk segítségével ez nem annyira nehéz feladat. A WS-Addressing szabvány népszerűségének köszönhetően a WCF-ben olyan szinten támogatva van, hogy az üzeneteknél közvetlenül megadhatóak, hogy a különböző WS-Addressing fejléc mezőknek mi legyen az értéke. Ebből adódóan a tényleges interceptor funkcionalitás lényegesen egyszerűbb, mint a Metro és CXF esetében. Az elkészült protokoll csatorna beillesztése is gond nélkül történt és az elvártaknak megfelelően elkapta az üzeneteket.

A mintaalkalmazás bankja és egy légitársasága WCF szolgáltatásként lett megvalósítva, beépítve a megfelelő interceptorral mind kliens, mind szerver oldalon.

## 5.2 Monitorozó szerver

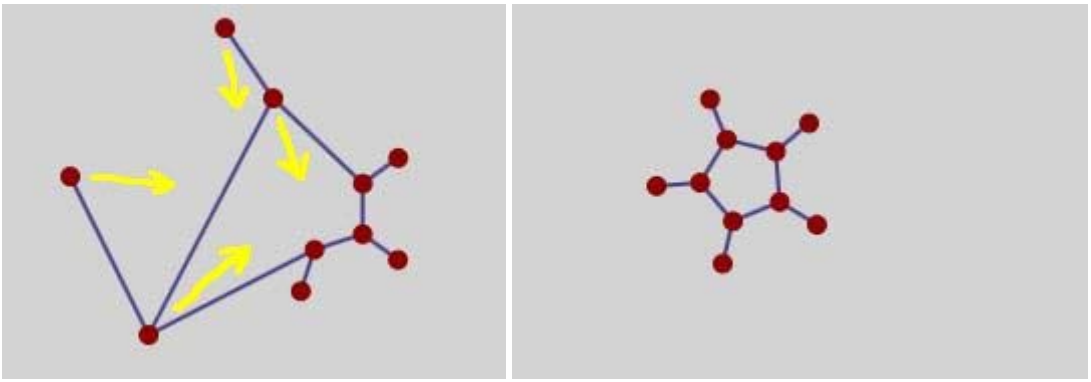
A monitorozó szerver a felügyeleti rendszer központi elemét alkotja. Az interceptorok az elkapott üzeneteket hozzá továbbítják, amiket beír az adatbázis eseményeket rögzítő táblájába. Továbbá számon tartja a szolgáltatások állapotát is (elérhető-e vagy sem) és bizonyos időközönként „ping” üzenetet küld nekik. A válasz (vagy annak hiánya) alapján frissíti az állapotokat és a változásokat az adatbázisban rögzíti. A szervert egy WCF-es szolgáltatás valósítja meg. Ezek után már minden adat rendelkezésre áll a szolgáltatásokat megjelenítő áttekintő ábra létrehozásához, illetve a különböző forgalmi statisztikák elkészítéséhez. A mintaalkalmazás esetében az összes szolgáltatás (légitársaságok, bank) interceptora ennek a szervernek küldi a kommunikáció adatait, illetve ide küldik a „ping reply” üzeneteket.

## 5.3 Megjelenítő alkalmazás

A megjelenítő alkalmazás célja egy átfogó képet adni a rendszerről. Ezt úgy éri el, hogy megmutatja a gépek/szerverek/végpontok kapcsolatait és állapotát egy gráfban. A gráf képes az áramló üzenetek megjelenítésére is. Továbbá az alkalmazás diagramok segítségével forgalmi statisztikák kimutatására is alkalmas. Az implementáció a WPF keretrendszer segítségével történt.

Az alkalmazás fő része a gráf létrehozásáért, megjelenítéséért és kezeléséért felelős logika. A gráf implementációja a Model-View-Controller tervezési mintára épül. Ennek előnye, hogy elkülöníti a megjelenítést, a felhasználói interakció lekezelését és az üzleti logikát. A komponensek közötti laza csatolás létrehozásáért pedig az Observable minta felel. A gráf valójában nem csak egy egyszerű gráf, ami tárolja a csúcsokat és a hozzájuk tartozó

éleket. A gráf dinamikusan átrendezi önmagát a jobb átláthatóság érdekében. Ennek lényege, hogy a csúcsok és élek a fizikából ismert elektronok taszításának analógiájára építve próbálja rendezettebb struktúrába átrendezni a gráfot. A csúcsok (elektronok) taszítják egymást, azonban az élek mentén némi vonzás tapasztalható. Ezen egyszerű jelenségek arra késztetik a gráfot, hogy „kifeszítse magát”: a csúcsok minél távolabb helyezkedjenek el, de az élek mentén ez a távolodás korlátozódik (ez látható az alábbi ábrán).



**3. ábra A gráf dinamikus átrendeződése**

A csúcsok a felhasználó által mozgathatóak, rögzíthetőek, megváltoztatható a töltésük.

A gráf segítségével megtekinthető a rendszer valós idejű helyzete (valójában 1 perc késéssel), valamint korábbi időpontokból vissza is játszható a hálózat működése. Az adatbázisból kiolvasott adatok alapján forgalmi statisztikák megjelenítését is támogatja az alkalmazás diagramok formájában.

# 6 Összegzés

Bár az alkalmazás még gyerekcipőben jár, a szolgáltatásokba beépített interceptor rétegnek köszönhetően alkalmasnak tűnik a rendszerről egy átfogó kép alkotására. Sajnos ez az interceptor réteg jelenti a korlátot is: csak azon szolgáltatások esetén működik, amik valamely támogatott technológiában lettek elkészítve. Továbbfejlesztési lehetőségek is adódnak bőven: további szolgáltatásvermek támogatása, interceptorok hatékonyabb kommunikációja, helyesség vizsgálata, nem módosítható szolgáltatások felügyelete. A létrehozott rendszer természetesen túl egyszerű ahhoz, hogy egy komplett menedzsment megoldásként szolgálhasson, de a jelenlegi termékeket kiegészítheti egy hasznos, más termékekben ily módon nem előforduló funkcionalitással.

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.